



APUNTE ELECTRÓNICO

Arquitectura de Computadoras

Licenciatura en Informática





COLABORADORES

DIRECTOR DE LA FCA

Mtro. Tomás Humberto Rubio Pérez

SECRETARIO GENERAL

Dr. Armando Tomé González

COORDINACIÓN GENERAL

Mtra. Gabriela Montero Montiel

Jefa del Centro de Educación a Distancia y Gestión del
Conocimiento

COORDINACIÓN ACADÉMICA

Mtro. Francisco Hernández Mendoza
FCA-UNAM

COORDINACIÓN DE MULTIMEDIOS

L. A. Heber Javier Méndez Grajeda
FCA-UNAM

AUTORES

Ing. Tomás García González
Ing. Antonio Martín Garcés Madrigal

REVISIÓN PEDAGÓGICA

Lic. Dayanira Granados Pérez
Mtro. Joel Guzmán Mosqueda

CORRECCIÓN DE ESTILO

Mtro. Carlos Rodolfo Rodríguez de Alba

DISEÑO DE PORTADAS

L.CG. Ricardo Alberto Báez Caballero

DISEÑO EDITORIAL

L.D. y C.V. Verónica Martínez Pérez



Dr. Enrique Luis Graue Wiechers
Rector

Dr. Leonardo Lomelí Vanegas
Secretario General



Mtro. Tomás Humberto Rubio Pérez
Director

Dr. Armando Tomé González
Secretario General



Mtra. Gabriela Montero Montiel
Jefa del Centro de Educación a Distancia
y Gestión del Conocimiento

Arquitectura de computadoras

Apunte electrónico

Edición: 23 octubre 2018.

D.R. © 2018 UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Ciudad Universitaria, Delegación Coyoacán, C.P. 04510, México, Ciudad de México.

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

ISBN:
Plan de estudios 2012, actualizado 2016.

“Prohibida la reproducción total o parcial por cualquier medio sin la autorización escrita del titular de los derechos patrimoniales”

“Reservados todos los derechos bajo las normas internacionales. Se le otorga el acceso no exclusivo y no transferible para leer el texto de esta edición electrónica en la pantalla. Puede ser reproducido con fines no lucrativos, siempre y cuando no se mutile, se cite la fuente completa y su dirección electrónica; de otra forma, se requiere la autorización escrita del titular de los derechos patrimoniales.”

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 **primera** unidad es introductoria, 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.

La unidad **dos** trata de los sistemas numéricos, del uso del sistema de numeración binario y sus representaciones octal y hexadecimal; el sistema binario es el soporte matemático utilizado por los circuitos electrónicos de una computadora y 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 2 a 4, constituyen la parte conceptual, el soporte matemático de la materia.

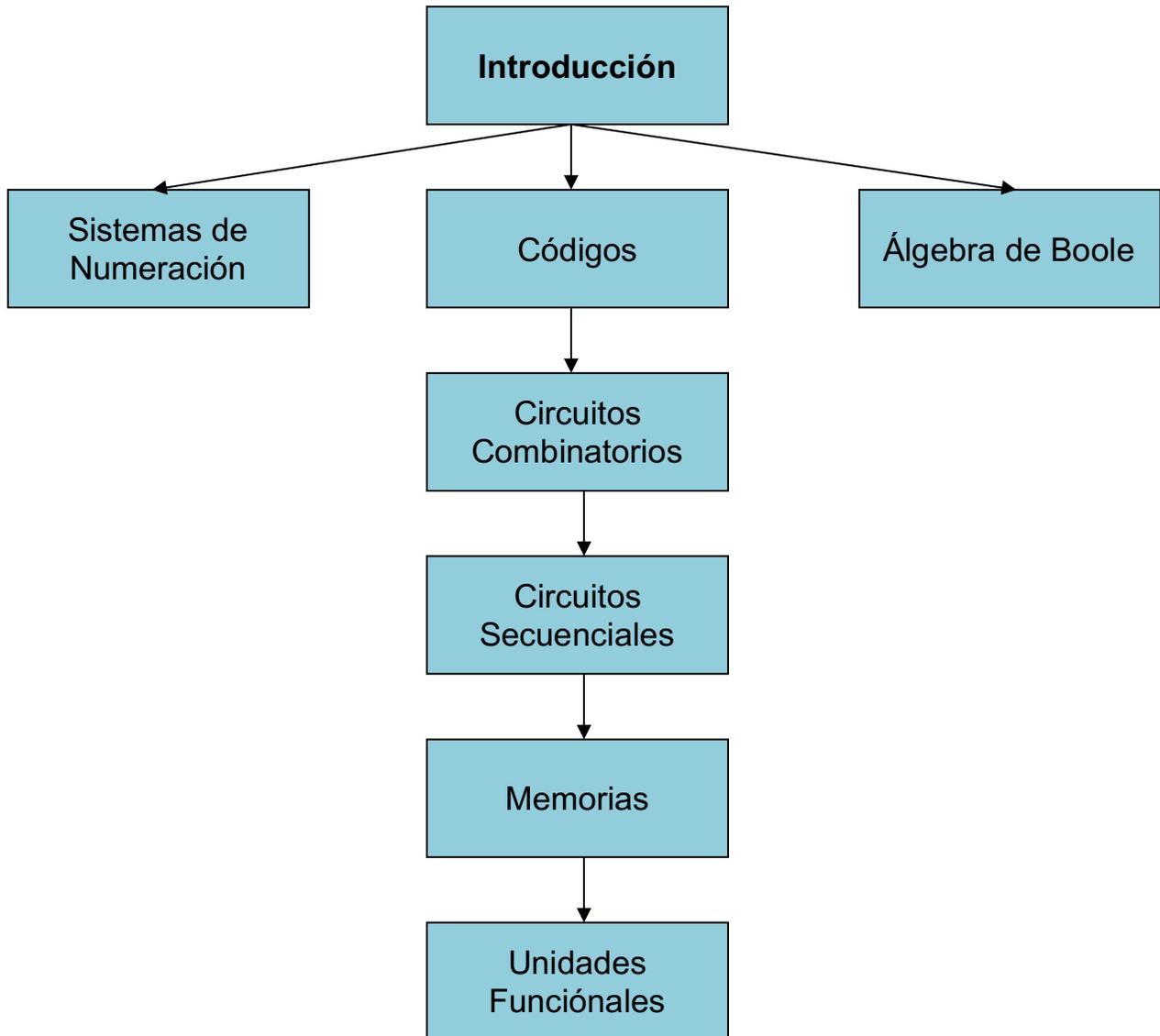
En la unidad **cinco** se presentan los diversos circuitos combinatorios, que funcionan exclusivamente a partir de los valores de sus entradas como eventos de ocurrencia y que son independientes de la variable tiempo para su funcionamiento y a partir de los cuales se crean las partes reconocibles de una computadora.

La unidad **seis** es donde estudiaremos los circuitos lógicos secuenciales y, además, se revisarán otros parámetros de interés para entender conceptos como el tiempo de proceso y las señales de sincronía, así como los elementos básicos de memoria creados a partir de celdas digitales denominadas *flip flops*.

La unidad **siete** presenta los principales tipos de memoria, así como su funcionamiento, tiempos de acceso y direccionamiento y control de las mismas.

Finalmente, en la unidad **ocho** se relacionan los elementos vistos hasta la unidad siete para integrarlos al modelo de computadora actual, en sus unidades funcionales y subsistemas lógicos.

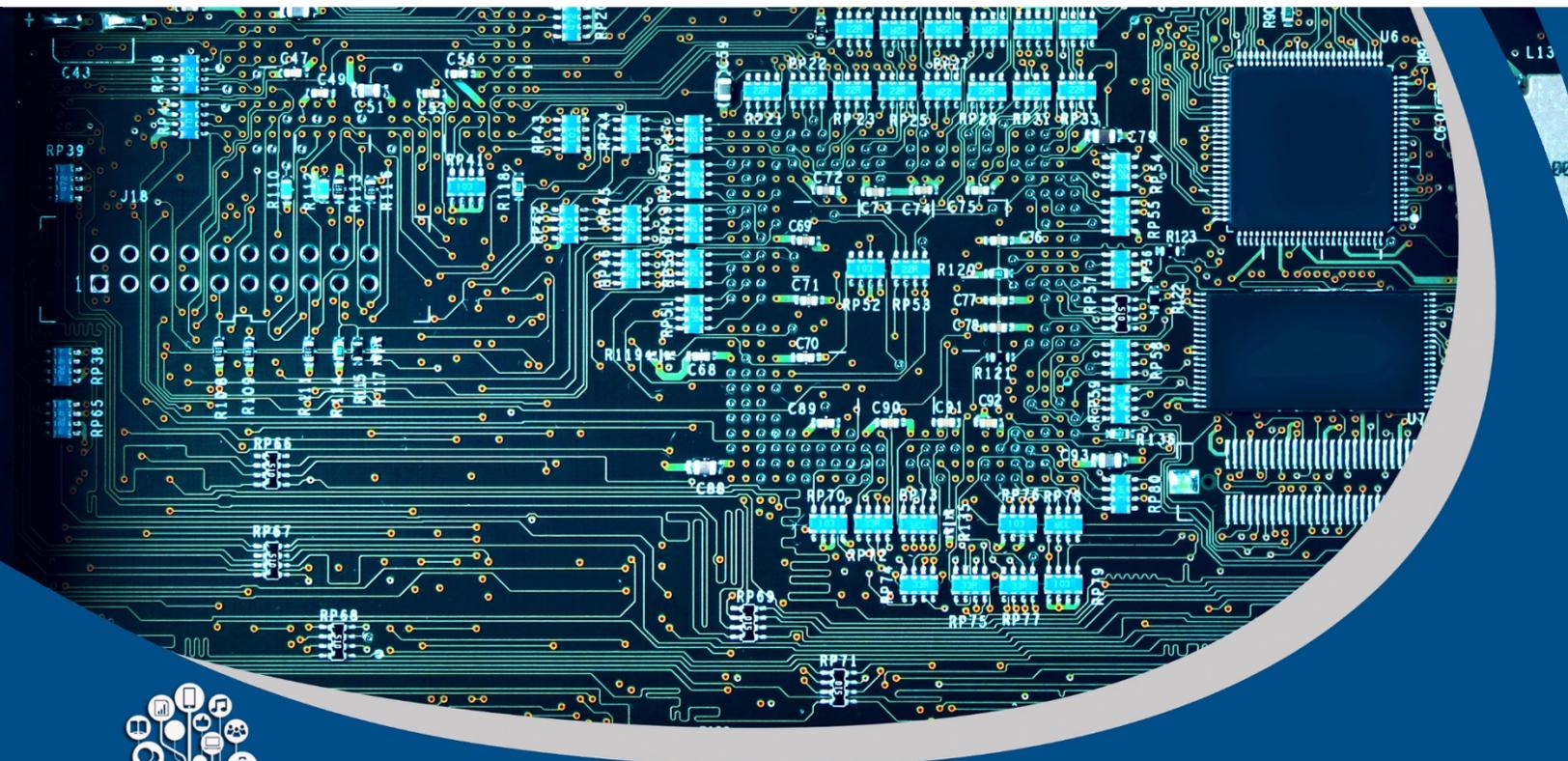
ESTRUCTURA CONCEPTUAL





UNIDAD 1

INTRODUCCIÓN





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

El deseo de controlar las máquinas es un afán que inquieto a la humanidad desde la antigüedad y lo podemos ubicar en diferentes épocas, sin embargo, desde un enfoque en la historia más reciente, nos podemos centrar en los griegos y sus desarrollos mecánicos, de navegación y de control asociados. Tuvo representantes destacados como Arquímedes, Filón de Bizancio y Herón de Alejandría. En una etapa más actual podemos destacar los siglos XVII al XIX donde los desarrollos de Leibniz, Charles Babbage, Joseph Marie Jacquard y Ada Lovelace con sus creaciones respectivas: la rueda de Leibniz, la máquina diferencial, las tarjetas perforadas y equipos como el Aritmómetro, que llevan a la mente humana al concepto de programabilidad de las máquinas, situación que se consolida en el siglo XX con las teorías de grandes científicos como Claude Shanon, Konrad Suze, Alan Turing y John Von Neumann.

En la actualidad, la arquitectura (organización interna) de las *computadoras digitales* constituye un importante tema de estudio para la práctica cotidiana de nuestra profesión. En esta unidad explicaremos la arquitectura básica de una computadora digital, el funcionamiento de cada uno de sus componentes y la interrelación que existe entre ellos.

1.1. Estructura básica de las computadoras

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

Sin embargo, es conveniente hacer una breve revisión de eventos en una línea de tiempo y desarrollo científico y tecnológico que enmarcan el surgimiento de la computadora como la conocemos hoy.

En 1801 Joseph Marie Jacquard desarrolla un telar en el que su operación era controlada por tarjetas perforadas, estas controlaban el equipo *sin necesidad de modificar su estructura mecánica*, el desarrollo tecnológico lleva al estudio de conceptos sobre la “*programabilidad*” de las máquinas.

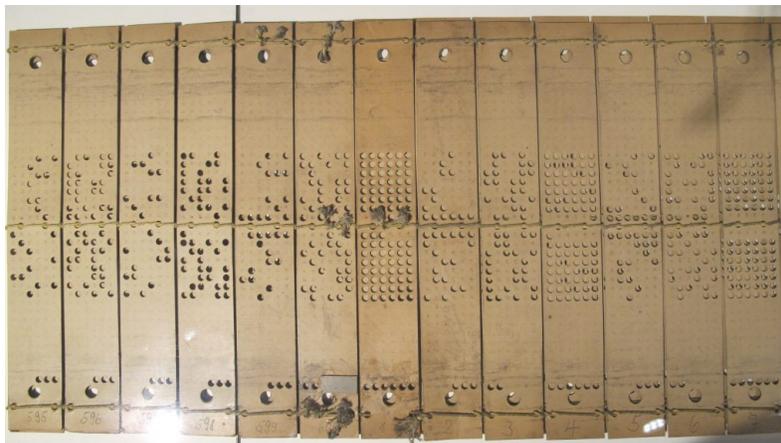


Imagen 1.1 Tarjetas perforadas del telar de Jacquard
Trilnick, Carlos (2018) “*Telar de Jacquard*”. Recuperado el 8 de octubre de 2018, disponible en:
<http://proyectoidis.org/telar-de-jacquard/>



Posteriormente, en los años previos a la segunda guerra mundial (1930-1940) se desarrollan las *computadoras análogas*, que son equipos que permiten resolver problemas científicos y técnicos considerando *esquemas secuenciales*; estos equipos funcionan a partir de conjuntos de tubos de vacío y relevadores electromecánicos y pueden realizar algunas operaciones matemáticas por medio de señales eléctricas; sin embargo, esta forma de operación ocasiona que su tamaño y consumo de energía sean muy grandes; además, estos equipos presentan el inconveniente de tener que ser reconfigurados (programados) manualmente a través de cables eléctricos en sus interconexiones en cada nueva actividad a desarrollar, por lo que una vez alambrados solo pueden realizar una actividad específica, además esta programación podía tardar semanas en ser realizada.

En el año de 1936, el matemático inglés Alan Turing realiza trabajos teóricos acerca de la posibilidad de resolver todo tipo de problemas de forma secuencial, demostrando que, en general, esto no era posible y propone el modelo de *máquina universal de Turing*, que permite establecer máquinas de propósito general, en contraparte de las computadoras análogas de propósito específico. Se considera que Turing es el padre del *procesador* (CPU) actual.

En 1937 Claude Shannon desarrolla un trabajo en el que implica el *álgebra Booleana* en la operación de circuitos de tipo binario (0/1, sí/no, prendido/apagado), cuyo comportamiento físico se asociaba a los relevadores electromecánicos y a una forma de operación de los tubos de vacío (bulbos) en su operación como interruptores, avance que es que es fundamental para la operación de las computadoras digitales.

En 1941, el científico alemán Konrad Zuse fabrica las primeras *computadoras digitales operativas de propósito general controladas por programación*, estos equipos (Z1-Z3), como los similares de la época, funcionaban a partir de relevadores electromecánicos y bulbos.



En 1945 el científico húngaro John von Neumann desarrolla un modelo de computadora en la que se establece *la estructura* actual de los equipos de cómputo; de manera paralela es desarrollada otra estructura computacional denominada arquitectura Harvard.

En 1948 se inventa *el transistor* electrónico por los científicos estadounidenses John Bardeen, Walter Brattain y William Shockley. Este dispositivo tiene propiedades operativas y lógicas similares a los relevadores electromecánicos y a los tubos de vacío, pero físicamente es mucho más pequeño, rápido y comparativamente consume una fracción mínima de energía eléctrica, basado en esto se empiezan a construir computadoras transistorizadas, lo que permite la difusión de las computadoras a otros ámbitos, pues los modelos previos eran demasiado costosos, grandes y consumían enormes cantidades de energía.

En 1971 la compañía Intel presenta el primer *procesador miniaturizado* (uCPU o comúnmente conocido como uP), dando inicio a la tercera generación de computadoras. La primera generación de computadoras (con miles de partes electromecánicas, relevadores y tubos de vacío) ocupaba habitaciones enteras (17600 bulbos, 160 m² y 30 toneladas de peso) para su despliegue y la más representativa de ella, ENIAC, realizaba 5000 sumas por segundo, en tanto que la segunda generación (transistores) eran de menor tamaño (de un armario a un escritorio) pero requerían de miles de dispositivos transistorizados para su implementación, lo que generaba problemas para su construcción, operación y mantenimiento. Así, con el moderno desarrollo diseñado de forma conjunta por Federico Faggin, Ted Hoff y Stanley Mazor, se fabrica el microprocesador Intel 4004, que era un dispositivo de aproximadamente 5 cm², 2300 transistores, operaba con una longitud de palabra de 4 bits en un encapsulado físico (*chip*) que cuenta con 16 contactos (*pines*) para establecer sus conexiones y que operaba a

una frecuencia de reloj de 740 KHz, consumía ocho ciclos de reloj por instrucción de programa, lo que significa que el chip era capaz de ejecutar hasta 92,600 instrucciones por segundo.

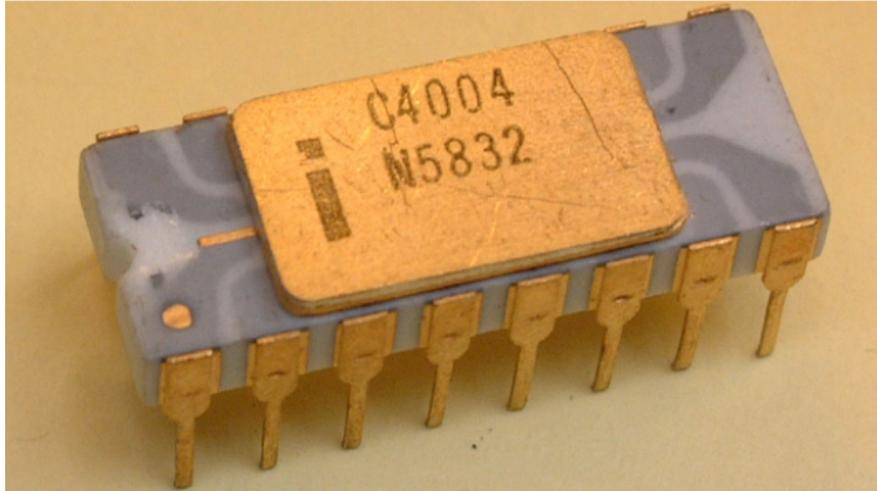


Imagen 1.2. Vista exterior del encapsulado del microprocesador Intel 4004, se observan claramente sus 16 pines (8 de cada lado). Recuperado el 7 de febrero de 2019 de: <https://www.profesionalreview.com/2018/04/21/cual-fue-el-primer-microprocesador-de-la-historia-y-quien-lo-invento/>

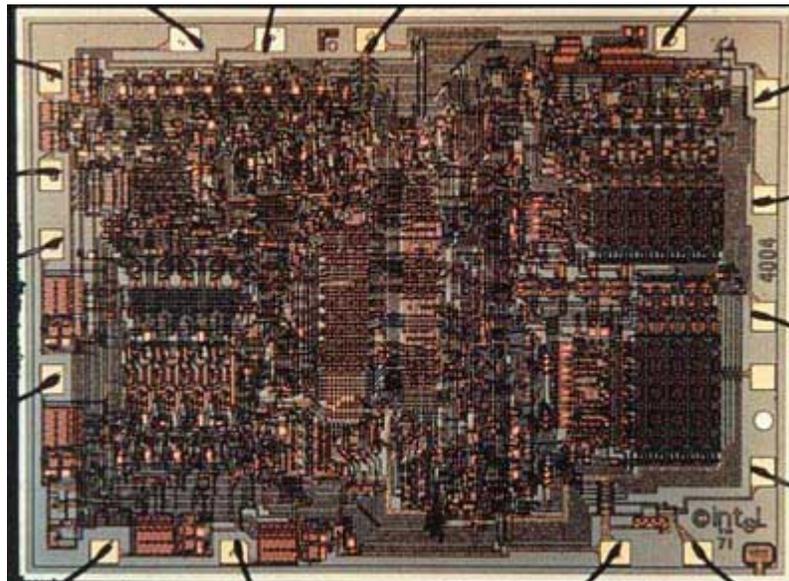


Imagen. 1.3 Interior del microprocesador 4004. Recuperado el 7 de febrero de 2019 de: <http://www.tugurium.com/gti/termino.php?Tr=4004> (Compartido bajo licencia Creative Commons BY NC SA.)



El interior del microprocesador 4004, consta de 2300 transistores que conforman todas las unidades funcionales de este dispositivo. La miniaturización lograda es evidente, la computadora Eniac (1940) ocupaba una superficie de 160 m², este procesador solamente ocupa 5 cm².

Esta revisión tiene en sí dos propósitos iniciales, uno es mostrar la forma como se va dando la secuencia de descubrimientos científicos y tecnológicos que permiten ir desarrollando el equipamiento o *hardware* de las computadoras, su programación y modelado y, otro, es puntualizar los elementos que constituyen una computadora. De esta forma podemos observar que están involucradas operaciones secuenciales, el álgebra binaria y los dispositivos que pueden ser modelados a través de ella (relevadores, bulbos y transistores), los programas que controlan el funcionamiento de los equipos, la interrelación matemática que hay entre estos elementos (máquina de Turing) y la estructuración física en la que pueden operar todos ellos (arquitecturas de Von Neumann y Harvard). Este conjunto de elementos, aparentemente inconexos, puede ser revisado a través de una analogía simple, la forma en cómo se construye un edificio: en él hay cimientos, columnas, techos, ventanas, etc. y la forma en la que se estructuran o interconectan se denomina arquitectura de las edificaciones o simplemente arquitectura. En nuestro caso, a partir de este símil, podemos observar el sentido de nuestra materia, es decir la arquitectura de las computadoras, implica qué elementos y el cómo y deben interconectarse para lograr establecer una estructura funcional. A partir de esto procederemos a presentar los dos paradigmas de arquitecturas computacionales y los elementos que los constituyen y así facilitar el entendimiento de los conceptos y disciplinas que deben de ser estudiados en nuestra materia.

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 (Boole, Shanon y Turing) y el concepto de “programa almacenado” (Jaqcquard y Zuse). La estructura von Neumann se ilustra en la siguiente figura y está formada por:

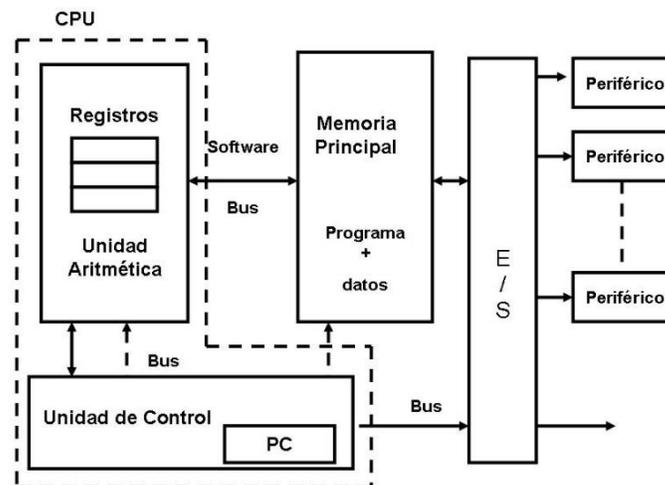


Figura 1.1 Arquitectura von Neumann

Fuente: Lynxblack. (2016). *¿Qué es la arquitectura von Neumann?* Recuperado el 26 noviembre de 2018, de: <https://frikosfera.wordpress.com/2015/02/27/que-es-la-arquitectura-von-neumann/>

Las computadoras se componen por las siguientes unidades:

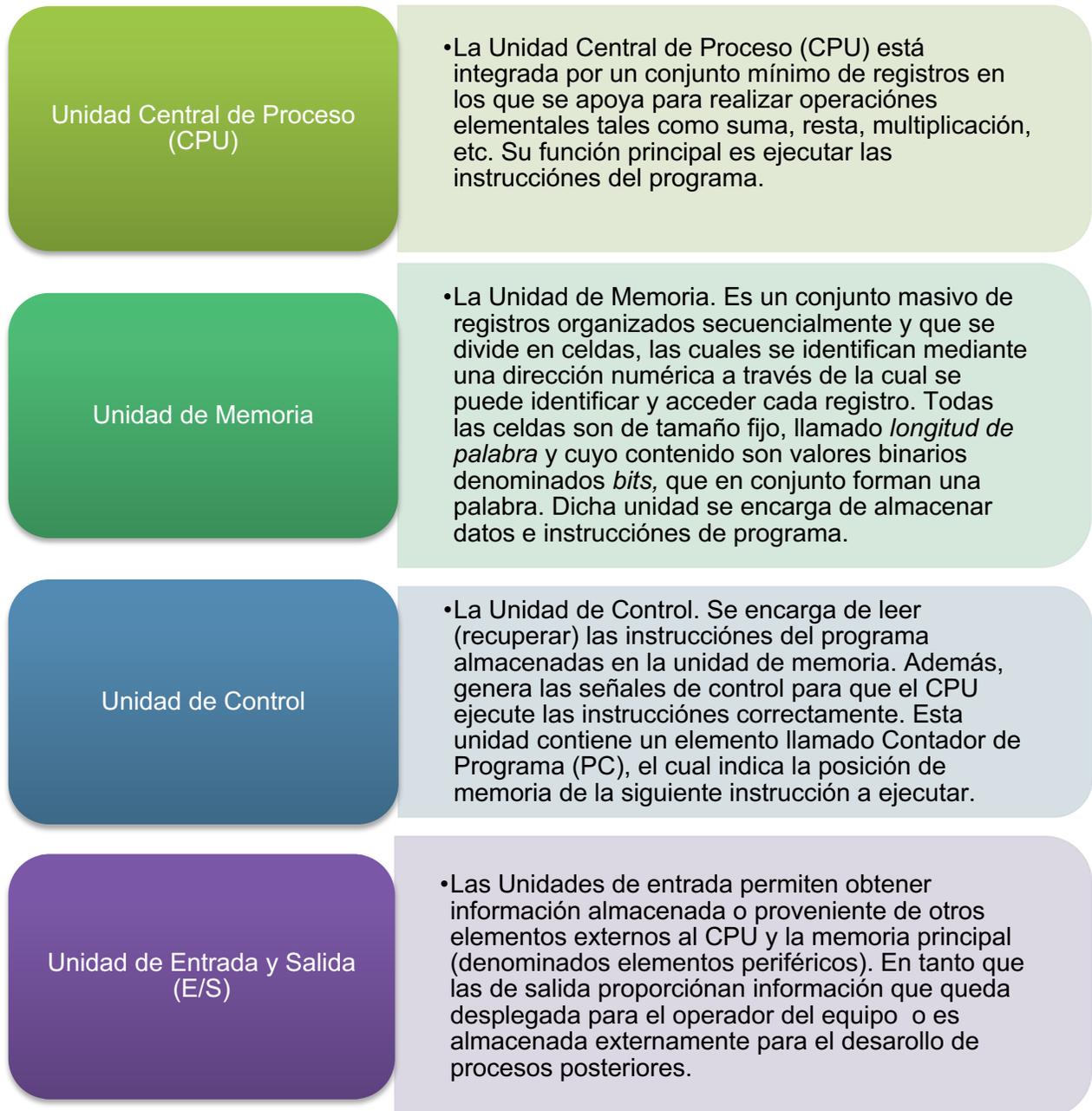


Figura 1.2 Elementos básicos de una computadora. Elaboración propia.

Todos estos bloques o unidades funcionales están interconectados por medio de un conjunto de líneas y/o alambres de cobre, por las cuales se transfieren eléctricamente los bits de información de cualquier unidad funcional a otra, este conjunto de líneas se denomina *bus de datos* (o *canal*). Este puede transmitir la información de manera unidireccional (en un solo sentido, “hacia”) o de manera bidireccional (en ambos sentidos, “hacia” o “desde”), su objetivo es conducir a través de señales eléctricas las instrucciones, datos y parámetros de control entre las unidades funcionales de la computadora (CPU, memoria, etc.).

La estructura de von Neumann utiliza el modelo de “programa almacenado” y dicho modelo presenta las siguientes ventajas:



Figura 1.3 Ventajas del programa Von Neumann

Una de las principales aportaciones de la estructura de Von Neumann es el concepto de “programa almacenado” el cual se explica a continuación:

Las computadoras con esta arquitectura resuelven un problema en dos fases: *compilación* y *ejecución*. Durante la fase de compilación se lee una serie de instrucciones introducidas (programa fuente, codificado y suministrado por el



operador o el programador del equipo) y éstas se traducen a lo que se denomina *lenguaje de máquina* (recordemos que el CPU opera con instrucciones binarias constituidas de ceros y unos o *bits*, como lo propuso Shannon) y que se almacenan en la memoria principal. Cada instrucción puede estar formada por una o varias *palabras* en una instrucción única. Durante la fase de ejecución, cada instrucción se recupera de la memoria principal en secuencia indicada por la unidad de control, desde la unidad de almacenamiento y se retiene temporalmente en un registro de instrucción (en el CPU) en tanto se ejecuta. Esta operación en dos fases, en la cual el programa fuente se traduce y se almacena (compilación) y luego se ejecuta (ejecución) de manera automática, se conoce como el concepto de *programa almacenado*.

El concepto de programa almacenado hace posible la ejecución de un programa dentro del CPU de la computadora sin tener que volverlo a escribir. Una ventaja de esta estructura computacional es que puede ser utilizada para varias aplicaciones (*computadora de propósito general, Jacquard*), tan solo cargando y ejecutando el programa correspondiente.

En la sección anterior, mencionamos que, además de la arquitectura de von Neumann, surge casi de forma paralela otra arquitectura denominada Harvard. Esta última difiere principalmente de la anteriormente descrita en que la memoria principal se divide en dos secciones: memoria de datos y memoria de instrucciones. Esta forma de operación tiene la particularidad de poder operar más rápidamente, ya que la arquitectura von Neumann al pasar los datos y las instrucciones por un solo bus afecta el desempeño global del equipo; sin embargo, para la fecha en que surge, la complejidad (y el costo) del *hardware* de la estructura Harvard es más elevado, y debido a la madurez tecnológica de la época, se optó por un mayor uso del modelo de von Neumann. Otra virtud del modelo de von Neumann es que incorpora y sistematiza los avances desarrollados por Turing, Boole, Shannon, Zuse y Jacquard, los que proveen elementos sumamente importantes pero que aislados

no llegan a ser definitivos y que al ser integrados en este esquema se generaliza la forma en cómo debe de operar una computadora digital.

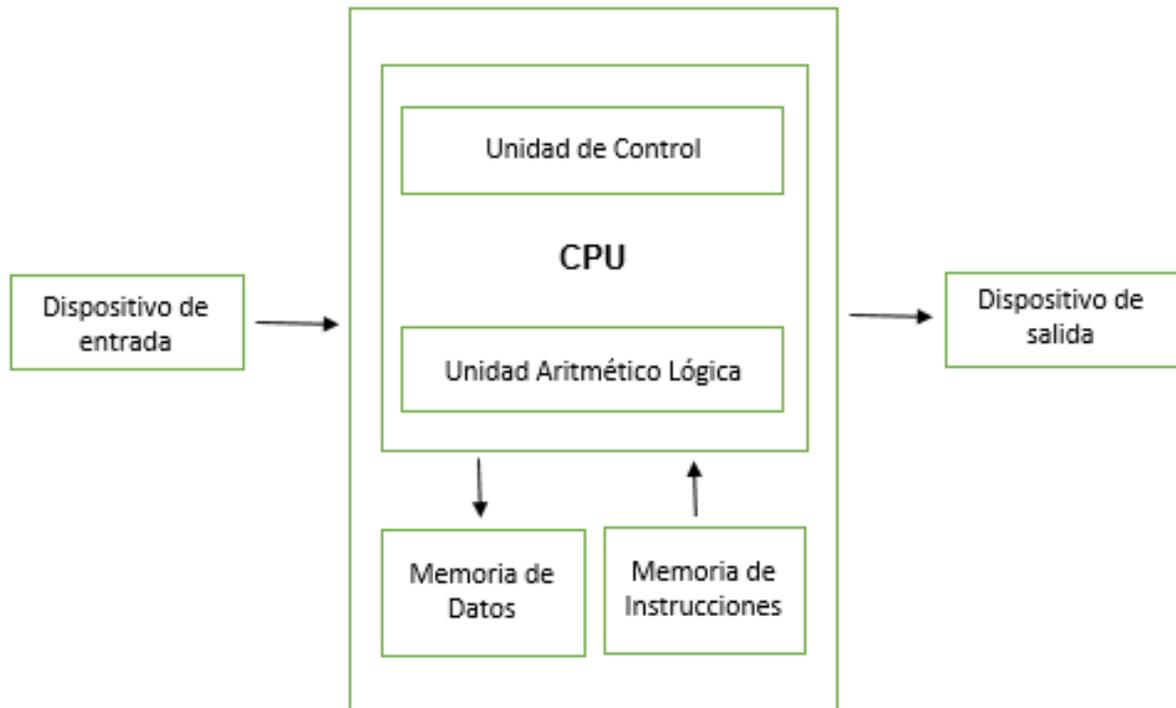
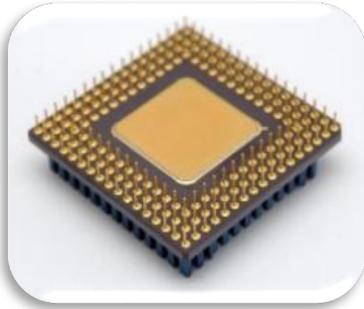


Figura 1.4 Arquitectura Harvard. Fuente:Arquitectura Von Neumann y arquitectura Harvard. (s.f.). Recuperado 26 noviembre, 2018, de <http://rcmcomputointegrado.blogspot.com/2012/04/arquitectura-von-neumann.html>

1.3. El microprocesador



Como se observa en el modelo de von Neumann, este elemento juega un papel central en la estructura de una computadora y como vemos en su esquema, en él se ejecutan las instrucciones del programa (equivale al “cerebro” de la computadora) y además se realizan las operaciones lógicas y matemáticas necesarias para determinar el *comportamiento y flujo* de éste. Su estructura en un inicio estaba constituida por elementos de suma, resta y operadores lógicos, implementados con cientos de relevadores electromagnéticos, bulbos o transistores, sin embargo, con el avance de la electrónica este último se pudo miniaturizar y llegar a la tecnología que hizo posible la fabricación los circuitos integrados o *microchips*, como se ilustra en la figura superior izquierda. El *microprocesador* (sus siglas son uP), como revisamos, fue implementado inicialmente con la tecnología PMOS (semiconductor oxido-metal de canal positivo), sin embargo, a lo largo de los años, estas tecnologías en circuitos integrados han evolucionado por diversas etapas (NMOS, CMOS, etc.) y en cada una de ellas se busca reducir aún más el tamaño de los transistores, su consumo de energía y la velocidad con la que pueden operar, de esta forma lograr uP más poderosos, pequeños y rápidos.

Mano M. Morris dice lo siguiente: “Un CPU encapsulado en una pastilla de circuito integrado se denomina *microprocesador*” (Mano, 2006: 4)

Tal como hemos insistido, el parámetro indicativo está dado por el tamaño del transistor. Mientras que en 1971 el uP 4004 contaba con 2300 transistores y fue construido con tamaño de transistor de 10 micras (0.00001 m), con longitud de palabra de 4 bits en un empaquetado de 16 contactos (pines), actualmente el microprocesador Intel Icore 7 cuenta con 382 millones de transistores, está



construido con la tecnología de 32 nanómetros (0.000000032 m), con longitud de palabra de 64 bits en un chip de 1366 pines.

Modelo	Año	Tran.	Tran./mm ²	W.
Procesadores de 8 bits, era previa al PC				
4004	1971	2.300	192	0,5
8008	1972	3.500	219	1,0
8080	1974	6.000	300	0,8
Procesadores de PC de 16 bits				
8086	1979	29.000	879	1,7
80286	1982	134.000	2.851	3,3
Procesadores de PC de 32 bits				
80386DX	1986	275.000	2.670	1,7
80486DX	1989	1.200.000	14.815	3,0
Pentium	1993	3.200.000	10.884	13,0
Pentium II	1997	7.500.000	36.946	12,0
Pentium III	1999	28.000.000	264.151	16,0
Pentium 4	2000	42.000.000	193.548	52,0

Figura 1.5 Evolución de la integración y miniaturización de los transistores en los microprocesadores Intel del 4004 al Core i7. Tran: número de transistores; Tran/mm²: transistores por milímetro cuadrado; W: potencia consumida en Watts por microcircuito. Note como el consumo de potencia no es proporcional al número de transistores, esto es debido al menor consumo energético de las nuevas tecnologías.

Fuente: Monti. (s.f.). Así han evolucionado los microprocesadores. Teleobjetivo. Recuperado: 26 noviembre, 2018, de <https://www.teleobjetivo.org/blog/asi-han-evolucionado-los-microprocesadores.html>



Modelo	Año	Tran	Tran/mm ²	W
4004	1971	2,300	192	0.5
8008	1972	3,500	219	1.0
8080	1974	6,000	300	0.8
8086	1979	29,000	879	1.7
80286	1982	134,000	2,851	3.3
80386DX	1986	275,000	2,670	1.7
80486DX	1989	1,200,000	14,815	3.0
Pentium	1993	3,200,000	10,884	13.0
Pentium II	1997	7,500,000	36,946	12.0
Pentium III	1999	28,000,000	264,151	16.0
Pentium 4	2000	42,000,000	193,548	52.0
Core i7	2011	995,000,000	4,406,481	95.0

Tabla 1.1. Evolución de procesadores. Elaboración propia.

La microcomputadora

La microcomputadora es un equipo de cómputo que está constituido por un conjunto de circuitos electrónicos digitales (o microchips), integrados en una arquitectura similar a la propuesta por von Neumann. Este conjunto de circuitos que constituyen un equipo se denomina *chipset*, y proveen de todos los elementos funcionales necesarios para la operación de la computadora: microprocesador, unidad de control, bus de direcciones y datos, memoria principal, control de periféricos (de entrada y salida), entre otros. Es importante remarcar esta última indicación debido a que la computadora ha evolucionado sustancialmente desde su propuesta original en cuanto al tipo y variedad de dispositivos, pero en esencia mantiene alguna de las dos arquitecturas propuestas a mediados del siglo pasado. Ha sido costumbre de los diversos fabricantes de estos circuitos el etiquetar estos dispositivos electrónicos

en una secuencia que permite identificar su relación funcional para el usuario; así el uP 4004 contaba con en su *chipset* MCS-4 en los circuitos 4001 (memoria de arranque de sistema o ROM), 4002 (memoria principal o RAM) y 4003 (unidad de registros de corrimiento). Actualmente, el procesador Intel Icore 7 tiene asociado (entre otras versiones del fabricante) el *chipset* H310, que incluye en un circuito integrado todos los elementos necesarios para aprovechar las prestaciones de los dispositivos actuales.

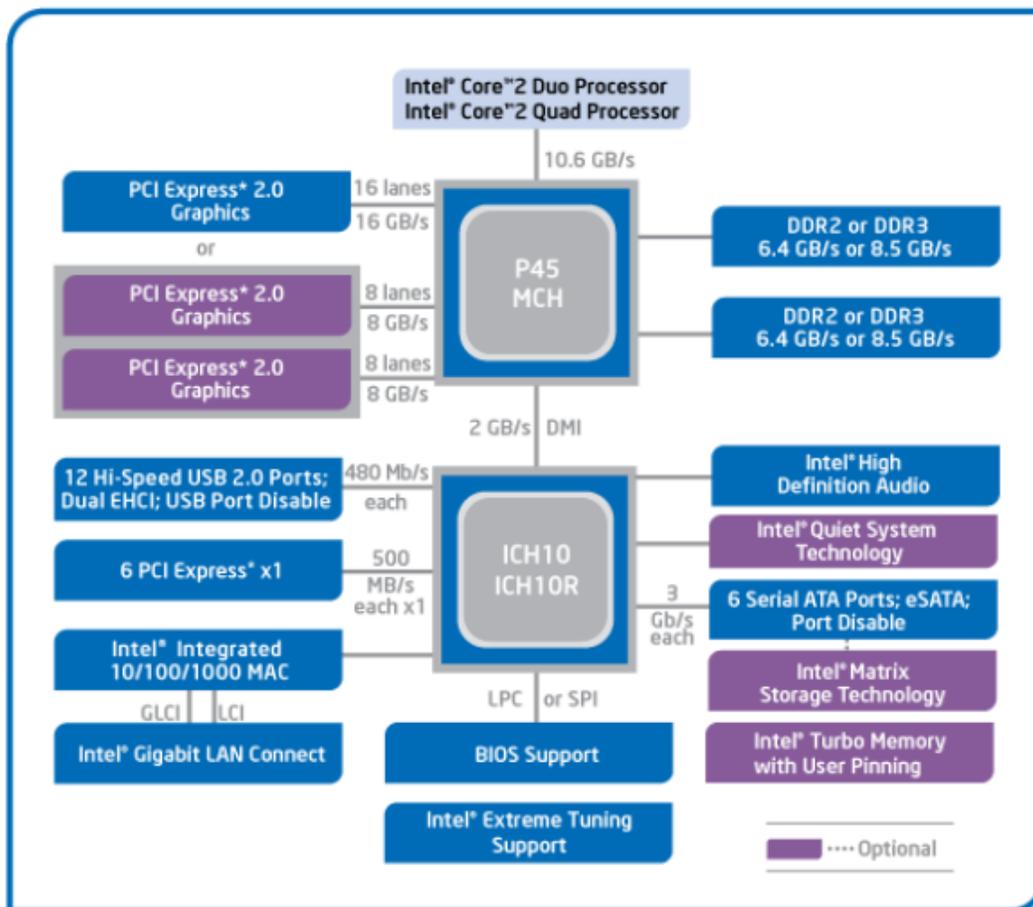


Figura 1.6 Diagrama de bloques de un sistema de cómputo moderno.¹

¹ Se observa el *chipset* indicado en los bloques denominados “Northbridge” que controla los dispositivos de alta velocidad (memoria y gráficos), y “Southbridge” que controla los dispositivos de entrada/salida más lentos, ranuras de expansión (*slots*) y dispositivos heredados antiguos (lista inferior derecha, algunos en desuso). Fuente: IXBT Labs. (s.f.). IXBT Labs - Intel P43 and P45 Chipsets (Eaglelake). Recuperado el 26 noviembre, 2018, de: <http://ixbtlabs.com/articles3/mainboard/i43p-i45p-chipsets.html>

Dentro de la amplísima gama de uP desarrollados a partir de 1971, podemos desatacar con fines ilustrativos el microprocesador 8085 de Intel y su *chipset* MCS-85 en una arquitectura mínima de 4 chips (8156, 8355, 8755, 8085), aunque en las especificaciones del fabricante ésta podía llegar a estar integrada hasta por 41 circuitos (1-mcs-85 family). Hemos elegido esta configuración con dos propósitos iniciales: a) ilustrar y enfatizar como en esencia la arquitectura von Neumann se ha mantenido y b) la facilidad para poder revisar a un amplio nivel de detalle esta arquitectura por medio de este uP y su *chipset* asociado, ya que hoy incluso podemos encontrar sistemas totalmente embebidos en un solo chip (como se observa en la computadoras, dispositivos móviles y teléfonos celulares de la actualidad). 8156 ram+3puertos i/o+timer. 8355 2kx8 ROM+i/o. 8755 EPROM+2 i/o ports.

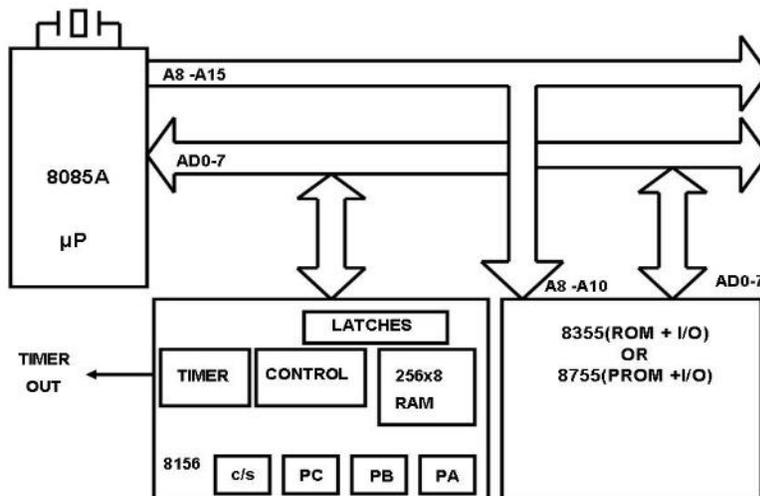


Figura 1.7 Esquema mínimo de computadora con base en el microprocesador Intel 8085^a
 Imagen del manual del circuito disponible en:
<https://www.jameco.com/Jameco/Products/ProdDS/52062.pdf>, figura 7

Como observamos del esquema básico anterior, podemos identificar el microprocesador (uP) 8085, unidad de control (8156), que en este caso también incluye la memoria principal (256x8 RAM) y tres unidades de entrada y salida (PA, PB y PC), adicionalmente tenemos el circuito 8355 que cuenta con un tipo de

memoria principal denominada ROM (Read Only Memory, memoria de solo lectura) y su función es permitir que al encender la computadora, esta se encuentre en estado operativo sin la intervención de un operador, además unidades de entrada salida. Chip 8755, que es un circuito con memoria programable (Programmable Read Only Memory) con la que se puede incrementar las posibilidades de inicio del equipo, además dos puertos de entrada salida (i/o ports). Con flechas de doble sentido y flechas de un solo sentido se indican el bus de datos y el bus de direcciones, que permiten acceder los datos y las instrucciones y los puertos de entrada salida (i/o). De esta descripción fácilmente podemos identificar la arquitectura de von Neumann, como se ha reiterado.

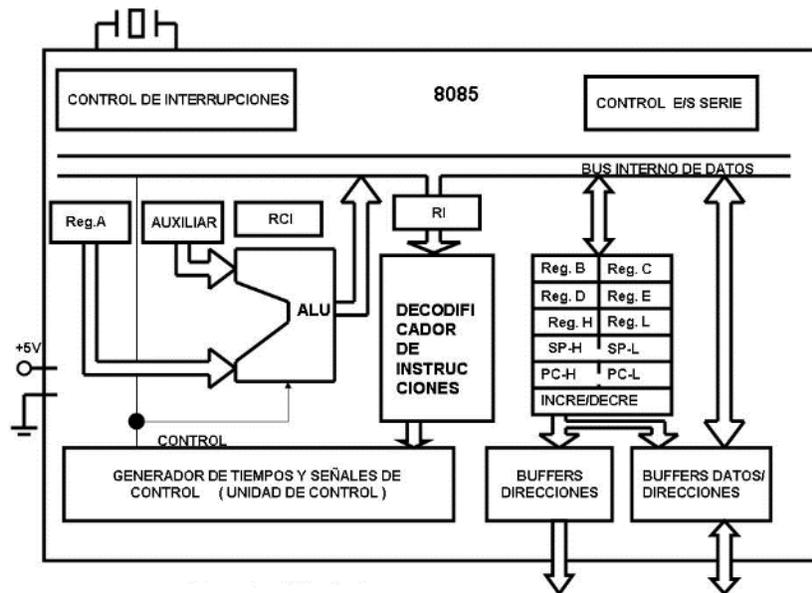


Figura 1.8 Arquitectura de μ P 8085
 Imagen obtenida del manual del circuito
 Fuente: Recuperado el 25 de enero de 2019 de:
<https://www.jameco.com/Jameco/Products/ProdDS/52062> .pdf, figura 1.



Imagen. 1.4 Chip Intel 8085, su encapsulado cuenta con 40 pines (contactos o patillas), 20 de cada lado. Imagen obtenida del manual del circuito
Fuente: Recuperado el 25 de enero de 2019 de:
<https://www.jameco.com/Jameco/Products/ProdDS/52062> .pdf, figura 1.

1.3.1 Bus de direcciones

Como hemos mencionado, el uP actúa como el “cerebro” del equipo de cómputo y a través de la unidad de control puede acceder a las diferentes unidades funcionales por medio de direcciones indicadas numéricamente (sabemos que el equipo opera por valores binarios (ceros y unos) de un sistema de numeración digital, Shanon); este canal es el medio físico por el que se conducen las señales de estas direcciones, en este caso, es necesario indicar que este bus es de tipo *unidireccional*, pues solamente selecciona (indica con qué chip se comunica), sin que los dispositivos seleccionados “le respondan” por este canal, así vemos que no es necesaria una comunicación de dos vías. Dependiendo del chiptset se establece un protocolo de comunicación por medio de señales eléctricas, que son codificadas y decodificadas internamente por los diferentes chips. Así, de la descripción asociada al manejo de las direcciones en este *bus*, debemos destacar la necesidad de conocer el *sistema de numeración binario*, que abordaremos en el siguiente capítulo.

1.3.2 Bus de datos

El bus de datos es el canal físico por el cual el procesador accede información de la memoria y los dispositivos externos. En una memoria el uP puede *escribir* información procesada en su interior, o *leer* de ella instrucciones o datos de resultados de procesos de los periféricos. Así podemos observar que la comunicación en este caso puede ser “leer” (desde) o “escribir” (hacia), por esta razón el bus de datos es de tipo bidireccional, a diferencia del canal de direcciones.

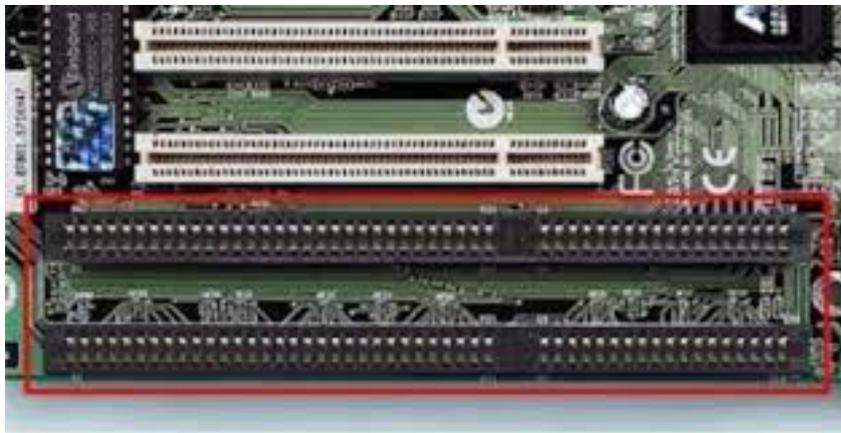


Imagen. 1.5 Bus de datos y direcciones en los conectores de expansión de una computadora personal.²

² Se nota su separación física, derecha e izquierda en ambos tipos, los de color negro son conectores tipo ISA (Industry Standard Architecture), los de color blanco son del tipo PCI (Peripheral Component Interconnect). Fuente: getestet, TOP. (s.f.). bundle retro vintage ATX Motherboard Mainboard ASUS P2B-D, getestet, TOP • EUR 79,00. Recuperado 26 noviembre, 2018, de <https://picclick.de/bundle-retro-vintage-ATX-Motherboard-Mainboard-ASUS-P2B-D-123122464521.html>



1.3.3. Bus de control

Este es el canal físico que conduce las señales eléctricas provenientes de la unidad de control y puede contener líneas de comunicación unidireccional o bidireccional, dependiendo del protocolo y características del chipset de que se trate.

1.3.4. Unidad de control

La unidad de control genera las señales de “semaforización” y de direcciones para que el uP pueda seleccionar y acceder a los diferentes circuitos de la microcomputadora (memorias y periféricos); esto implica aislar a un dispositivo direccionado de los no direccionados (observemos en el esquema mínimo del uP 8085 que todos los microcircuitos están conectados concurrentemente a los buses de datos y direcciones), de indicar si la operación es de lectura o escritura y de generar una secuencia de pulsos o *reloj del sistema*, que es la frecuencia a la que opera una computadora (por ejemplo el uP 4004 operaba a 740 KHz). Este conjunto de elementos es lo que se denomina el *protocolo de comunicación interno* (siguiendo el símil biológico del cerebro por parte del uP, así esta unidad puede ser identificada con el sistema nervioso autónomo) y permite que los diferentes chips se comuniquen entre sí. Es importante recordar que todas las acciones dentro de una computadora deben estar sincronizadas, para poder seguir correctamente las instrucciones de un programa (Turing, Zuse, computadoras analógicas), de otra forma cada dispositivo podría operar de manera descoordinada (desordenada) y no podrían lograr el propósito del programa. Dada la descripción de funcionamiento de esta unidad podemos destacar la necesidad de estudiar los circuitos generadores de las diferentes secuencias numéricas que hacen posible el funcionamiento de los microcircuitos (direcciones: contadores o secuenciadores; protocolo interno: codificaciones y combinaciones) y en cuanto a la frecuencia interna de operación de las computadoras, deberemos estudiar los circuitos electrónicos osciladores o

de reloj, elementos que por las razones descritas también serán revisados en capítulos posteriores.

1.3.5. Unidad lógica aritmética

Dentro del esquema de von Neumann esta unidad funcional está incluida al interior del CPU, y en el uP 8085 no es la excepción (pese a que en los procesadores Intel como el 8088, 8086, 80286, 80386 y 80486 era de tipo externo), pero la destacaremos de forma separada. Esta unidad, como su nombre lo indica, es la que ejecuta realmente el trabajo de procesamiento matemático y lógico. La ALU (*Arithmetic and Logic Unit*) puede recibir datos y efectuar con ellos diferentes operaciones aritméticas (dependiendo del uP), lógicas y de comparación y desplazamiento, entre otras. Para realizar las operaciones, las ALU de los uP, cuentan con diferentes registros o celdas de memoria interna. En el 8085 la ALU tiene dos registros asociados a ella para almacenar los datos, estos son el registro acumulador y el registro auxiliar y cada uno de ellos es de 8 bits. El registro principal del 8085 es el registro Acumulador y al comienzo de una operación contiene uno de los operandos y al final de la operación realizada almacena el resultado. Así de esta revisión, podemos identificar fácilmente la necesidad de conocer los números con los que opera la computadora (sistema de numeración binario) y como realizar operaciones aritméticas y lógicas en este sistema (aritmética binaria y álgebra de Boole), motivos por los que deberemos abordar esa temática en capítulos posteriores.



Imagen 1.6. Procesador Intel Exón³

³ Utilizado comúnmente en equipos servidores de cómputo. Fernando González. (2017, 13 abril). Intel comercializa los nuevos procesadores Xeon E3-1200 v6 - Movilidad Profesional HP. Recuperado 26 noviembre, 2018, de <https://www.muycomputerpro.com/movilidad-profesional/2017/04/14/xeon-e3-1200-v6-intel>



1.3.6. Registros

En general, un 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 específicas.

Registros de propósito general

En el uP 8085, 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 longitud de palabra de 16 bits llamados: BC, DE y HL. Existen además los registros W-Z (o INCRE/DECRE) que no son accesibles por programa y tan solo se utilizan 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 se indican a continuación:

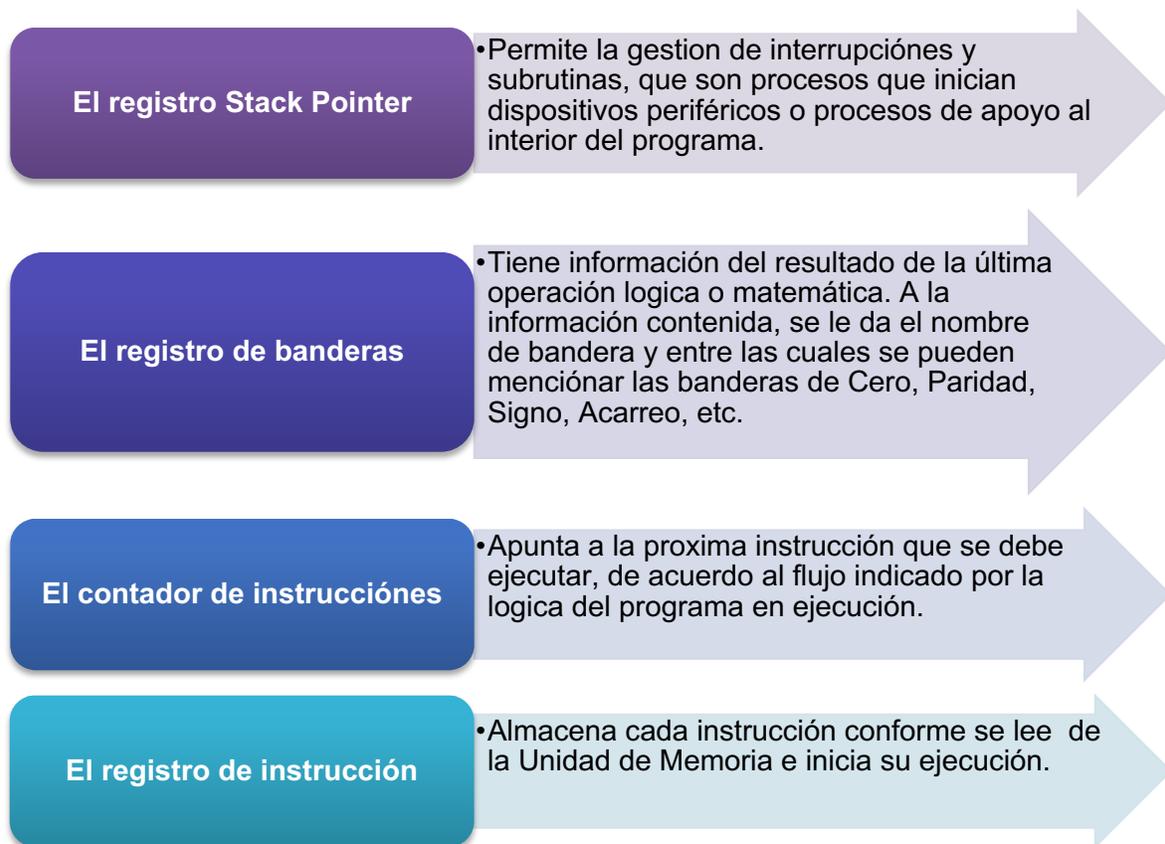


Figura 1.9 Registros de propósito especial. Elaboración propia.

Después de esta revisión podemos destacar una arquitectura resumida de la computadora actual y que de manera común utilizaremos a lo largo de estos apuntes: Unidad de Central de Proceso, Unidad de Memoria, Unidad de Control y unidades de entrada y salida interconectadas por buses unidireccionales y bidireccionales.

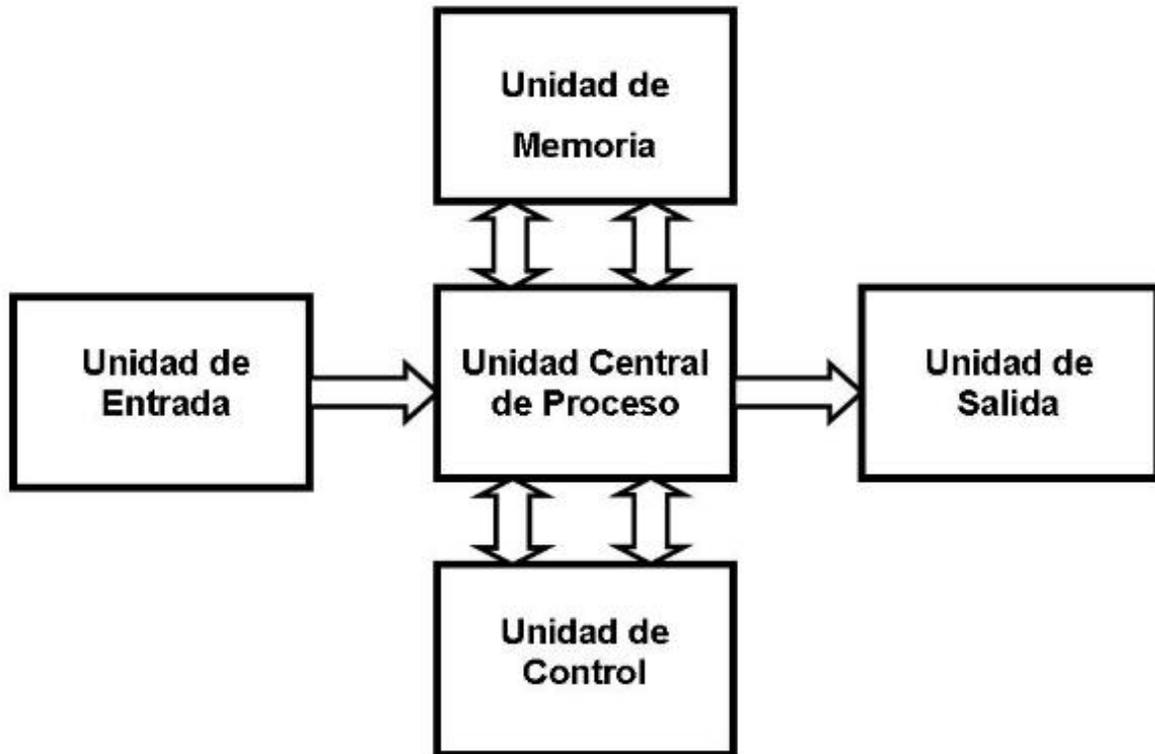


Figura 1.10 Arquitectura resumida de una computadora, según el modelo de von Neumann.
Elaboración propia.

Finalmente ilustraremos la asociación que hay entre equipos actuales y esta arquitectura computacional.

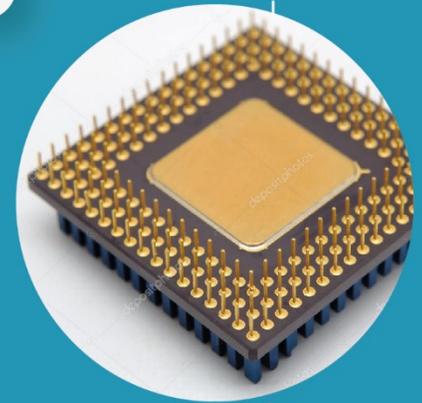
LA UNIDAD DE ENTRADA



A través de la unidad de entrada se suministran 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 programa. Aquí podemos destacar los teclados, pantallas táctiles y asistentes de voz como Siri, Alexa, Cortana, etc. (smartphones, tabletas, computadoras).

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, operaciones lógicas y operaciones de comparación entre otras. El uP se ubica en el interior del gabinete del equipo que se trate: PC, Tablet, Notebook, Smartphone, etc.



CYE. (S.F.). CPU Y SUS ELEMENTOS



La unidad Central de Proceso opera a velocidades más altas que la unidad de memoria. Cuando las cantidades de información son muy grandes la información se transfiere a una unidad de salida, que puede ser una unidad de disco duro externo, un SSD o un arreglo de discos duros tipo RAID, memorias USB, impresora, monitor, etc.

UNIDAD DE MEMORIA

Generalmente, las instrucciones (programa) y los datos se almacenan en la memoria interna de la computadora (Memoria principal), la cual se diferencia del dispositivo de entrada y periféricos (que también pueden ser una memoria) por su velocidad de operación. La memoria interna se diseña para almacenar y manipular cantidades relativamente pequeñas de datos (típicamente 1 a 8 GB, vs cientos a miles de GB), pero internamente opera a velocidades muy rápidas. La velocidad de la computadora está limitada en una proporción importante por la velocidad y cantidad 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 donde han de obtener los datos, donde han de suministrarlos y qué operaciones han de realizar. Esta unidad, al igual que el microprocesador se encuentra al interior del gabinete de los equipos.



UNIDAD DE SALIDA

La unidad de salida muestra información y resultados de manera visible (en una pantalla plana, táctil o no, en una impresora o plotter), auditiva (audífonos y bocinas) o en forma de acciones (control de dispositivos: IoT, sistemas de domótica, seguridad procesos industriales y de control). Hay situaciones en la operación de computadora en la que conforme el usuario introduce algún programa vía el teclado, cada carácter se visualiza simultáneamente en la pantalla.



Figura 1.11. Asociación que hay entre equipos actuales la arquitectura computacional de von Neumann. Elaboración propia.

Tipos de Computadoras

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



Figura 1.12 Tipos de computadoras. Elaboración propia.

Una *computadora analógica* trabaja internamente con señales eléctricas continuas provenientes de procesos físicos, actuadores y transductores, de tal manera que es posible medir las magnitudes de las cantidades involucradas en un circuito eléctrico que se comporta de manera *análoga* (o similar, de ahí su nombre) al proceso analizado y que se puede modelar en estos equipos por medio de la ecuación del fenómeno físico investigado; las señales de entrada (por procesar) y de salida (resultados) mantienen su naturaleza física en todo el proceso (no sufran alguna conversión, por ejemplo digital), su salida puede ser a una pantalla, un actuador electromecánico, eléctrico o electrónico. Por su naturaleza operativa, no son tan versátiles en su programación y, por lo tanto, se enfocan en propósitos específicos de tipo fijo o semifijo.



Imagen 1.8 Computadora analógica transistorizada estadounidense⁴,

⁴ Desarrollada para resolver ecuaciones diferenciales; como las que se revisan en nuestra carrera en la materia de cálculo diferencial, matemáticas III.

Fuente: Jurvetson Steve. (2005, 12 enero). File: What's That (30).jpg - Wikimedia Commons. Recuperado el 26 noviembre de: 2018, de:

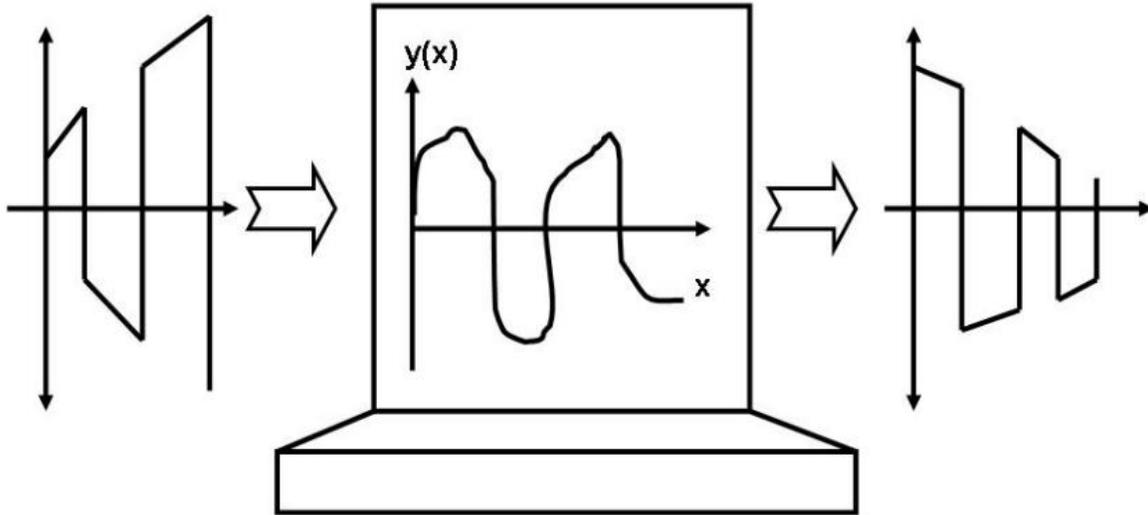


Figura 1.13 Esquema de una computadora analogica. Elaboración propia.

Una *computadora digital* trabaja internamente con señales electrónicas digitales o discretas, sin embargo, la señal de entrada (por procesar) puede ser: una señal analógica, una señal digital o un mensaje de texto, pero para ser procesadas por la microcomputadora sugieren un proceso de conversión denominado *digitalización* (a/d: análogo a digital) y la señal de salida puede ser una señal digital, una señal analógica (previa reconversión analógica o proceso d/a (digital a análogo)) o un mensaje de texto modificado (codificado) por un programa específico previamente almacenado el equipo de cómputo.

[https://commons.wikimedia.org/wiki/File:What%27s_That_\(30\).jpg](https://commons.wikimedia.org/wiki/File:What%27s_That_(30).jpg)

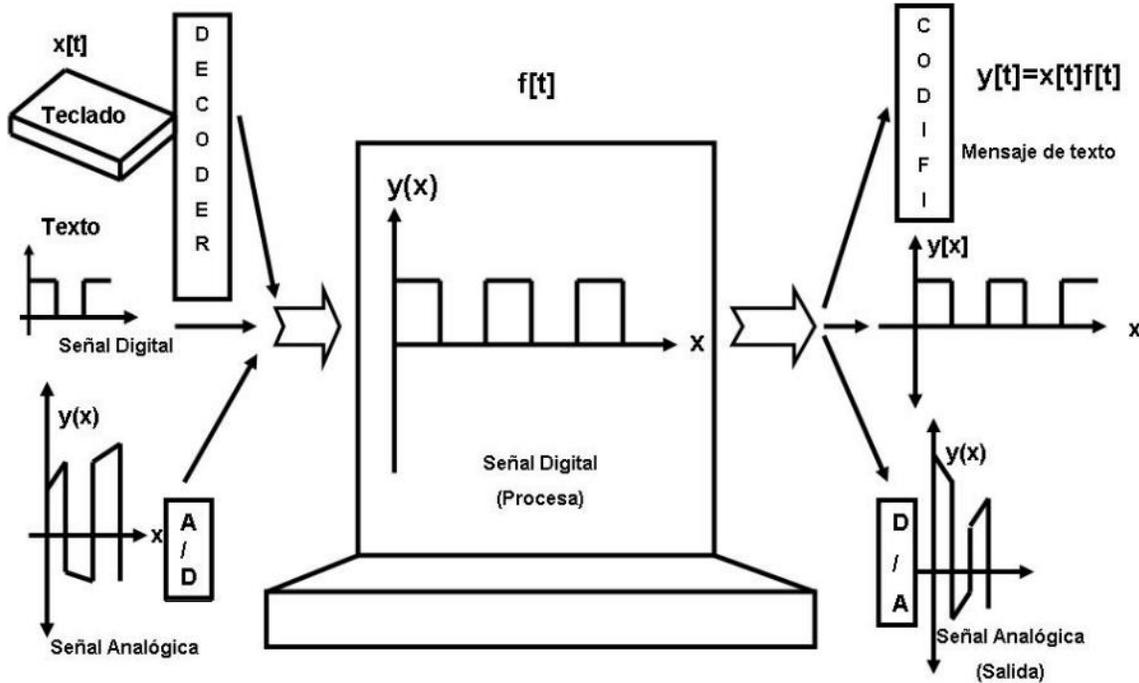
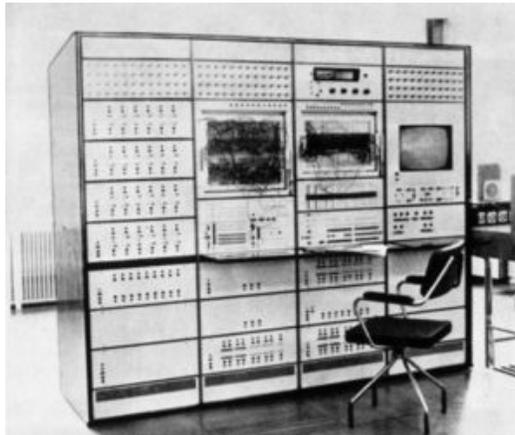


Figura 1.14 Esquema de una computadora digital. Elaboración propia.

Las computadoras digitales son más compactas, más baratas, ocupan menos espacio, consumen menos energía, su costo de mantenimiento es muy bajo y presentan características de versatilidad (propósito general), programabilidad y una excelente velocidad de operación interna. Estas ventajas son proporcionadas por su arquitectura ya que operan a partir de componentes electrónicos con alto nivel de integración (miniaturización).

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.

Imagen 1.8 Computadora híbrida EL WAT 1001⁵

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. Estos equipos pueden resolver muy rápidamente, por los algoritmos de sus programas, una gran variedad de problemas; ya sea de medicina, hacer una contabilidad, una proyección financiera, estudiar el diseño de un proyecto de ingeniería o de jugar ajedrez con el operador (el usuario) del equipo.

Computadoras para propósitos específicos

Una *computadora de propósito específico* está diseñada en torno a un problema particular y es optimizada para tratar solamente con ese tipo de problemas. Generalmente este tipo de computadora es de menor tamaño, más económica y más eficaz para la realización de esa tarea específica. Ejemplos típicos pueden ser el punto de venta de la tienda de conveniencia, el control de un avión, procesadores de los dispositivos de comunicación móviles o el control de una videocámara.

Figura 1.15. Propósitos de las computadoras. Elaboración propia.

⁵ Desarrollada en Polonia con bulbos, utilizada para resolver ecuaciones diferenciales (parte analógica) y realizar toma de decisiones (parte digital).

Fuente: Computadora híbrida - dominio público, computadora polaca. (2018, 29 octubre).

Recuperado: 26 noviembre 2018, de https://es.wikipedia.org/wiki/Computadora_h%C3%ADbrida



Ambos tipos de computadoras (de propósito general y de propósito específico) tienen básicamente la misma arquitectura propuesta por Von Neumann. La diferencia reside en las unidades específicas empleadas para introducir los datos en la computadora y para proporcionar la información de salida. A partir de este momento, cuando hablemos de computadoras, haremos referencia a las computadoras digitales de propósito general.

RESUMEN

En esta unidad, revisamos la arquitectura básica de las computadoras digitales. El modelo más utilizado es el de von Neumann que incorpora y sistematiza los avances desarrollados por Turing, Boole, Shanon, Zuse y Jacquard, científicos y tecnólogos que proveen elementos sumamente importantes para la computación, pero que aislados no llegan a ser definitivos y que solamente al ser integrados y sistematizados en este esquema establecen la forma en cómo debe de operar una computadora digital y a partir del cual se da la revolución tecnológica de la computación digital. Este desarrollo también es un parteaguas a cerca del cual se aceleran desarrollos en la electrónica por medio del transistor, la miniaturización, el desarrollo de los microprocesadores y los circuitos integrados que son los bloques y elementos funcionales que permitieron el surgimiento de la computación hoy.

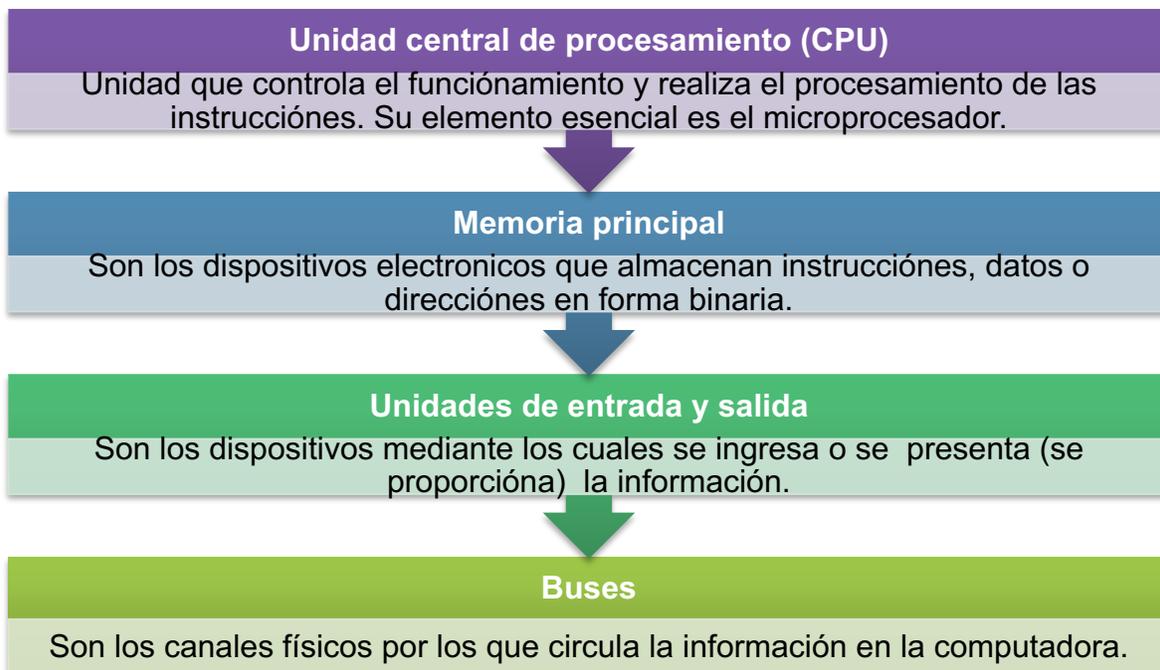


Figura 1.16 Arquitectura resumida de una computadora, según el modelo de von Neumann.
Elaboración propia.

BIBLIOGRAFÍA DE LA UNIDAD



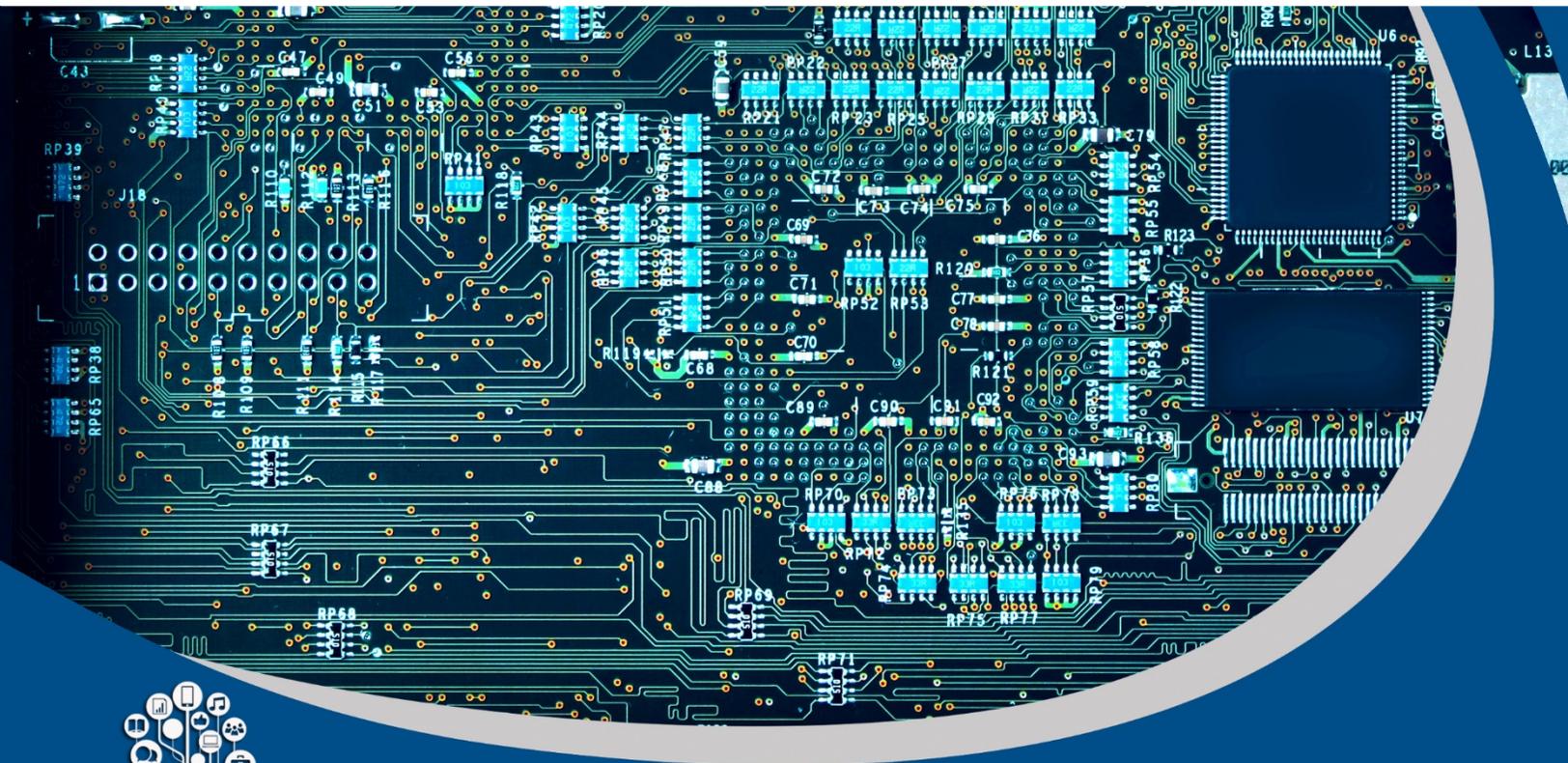
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 2

SISTEMAS DE NUMERACIÓN





OBJETIVO PARTICULAR

El alumno reconocerá los fundamentos teóricos de los sistemas numéricos, las conversiones entre bases numéricas y los algoritmos aritméticos en estos sistemas de numeración.

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 a^r-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 (basado en los dígitos 0-9) es algo “natural”; sin embargo, si se establecen reglas de construcción numérica a partir de otros conjuntos finitos de dígitos (por ejemplo 0-4, 0-1, etc.), la posibilidad de contar en esos sistemas es posible. Un ejemplo práctico lo proporcionan las computadoras, éstas utilizan el sistema de numeración binario. A diferencia del



sistema decimal, el sistema binario solo utiliza dos dígitos: 0 y 1. Entender cómo se realizan las operaciones básicas aritméticas en el sistema binario nos ayudará a entender como las computadoras procesan uno de los tipos de datos con los que las alimentamos, en este caso los datos numéricos.

El adquirir habilidad para el manejo de los sistemas binario, octal (del 0-7) y hexadecimal (del 0-15), al igual que cuando se aprende a hablar otra lengua, no solo 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 solo 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 distinguen 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 (de *digitus*: dedo, en latín). 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. Dado que los humanos utilizamos el sistema numérico decimal, entonces necesitamos comprender las reglas del sistema binario y de este estudio desprenderemos también la relación que existe con los sistemas octal y 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.

Establecida una generalización matemática para un sistema numérico de base “n”, podemos afirmar que el sistema más usado para el diseño y construcción de sistemas digitales de cualquier tamaño es el sistema binario; sin embargo, para realizar las actividades de programación (la parte realizada por los humanos) de los sistemas digitales se utilizan además los sistemas numéricos 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

$$N = d_{p-1}n^{p-1} + d_{p-2}n^{p-2} + \dots + d_0n^0 + d_{-(q-1)}n^{-(q-1)} + d_{-(q-2)}n^{-(q-2)}$$

o en su forma compacta

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

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**” (de menor valor relativo) y el que se encuentre más a la izquierda el de “**más significativo**” (de mayor valor relativo). En los dígitos decimales 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

Tabla 2.1 Equivalencias entre diferentes sistemas de numeración.
Fuente: UNAM. (s.f.). *Sistemas numéricos*. Recuperado: 26 noviembre, 2018, de https://www.unadmexico.mx/sitios/aplicaciones-107/LITE_36/Un_151_OperacionesAritmeticasBasicas/escenas/2_Inicio_1.html

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, los números 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*, investiga en algún sitio de internet el tema de **Sistemas numéricos**.

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 debido a que la operación física de los dispositivos a partir de los que se construyen estos equipos está relacionada directamente con la matemática binaria (del latín *binarius*: compuesto por dos elementos), ya que en su funcionamiento solo puede haber dos estados en su operación: prendido/apagado, asociados matemáticamente en el funcionamiento de las computadoras a los valores 0/1



(Shanon, capítulo 1). El sistema de numeración binario fue desarrollado formalmente por el matemático alemán Gottfried Wilhelm Leibniz en el siglo XVII, en su trabajo "*Explication de l'Arithmétique Binaire*"; este tipo de sistema numérico fue también estudiado por matemáticos de la India y China desde el siglo III antes de nuestra era.



TABLE 86 MEMOIRES DE L'ACADEMIE ROYALE DES NOMBRES.

bres entiers au-dessous du double du plus haut degré. Car icy, c'est comme si on disoit, par exemple, que 111 ou 7 est la somme de quatre, de deux & d'un. Et que 1101 ou 13 est la somme de huit, quatre & un. Cette propriété sert aux Essayeurs pour peser toutes fortes de masses avec peu de poids, & pourroit servir dans les monnoyes pour donner plusieurs valeurs avec peu de pieces.

Cette expression des Nombres étant établie, sert à faire tres-facilement toutes fortes d'operations.

Pour l'Addition ☉

110	6	101	5	1110	14
111	7	1011	11	10001	17
1101	13	10000	16	11111	31

Pour la Soustraction.

1101	13	10000	16	11111	31
111	7	1011	11	10001	17
110	6	101	5	1110	14

Pour la Multiplication.

11	3	101	5	101	5
11	3	11	3	101	5
11	3	101	5	101	5
11	3	101	5	1010	10
1001	9	1111	15	11001	25

Pour la Division.

15	3	5	101	5
3	3	1	101	5
3	3	1	101	5
3	3	1	101	5

Et toutes ces operations sont si aisées, qu'on n'a jamais besoin de rien essayer ni deviner, comme il faut faire dans la division ordinaire. On n'a point besoin non-plus de rien apprendre par cœur icy, comme il faut faire dans le calcul ordinaire, où il faut sçavoir, par exemple, que 6 & 7 pris ensemble font 13; & que 5 multiplié par 3 donne 15, suivant la Table d'une fois un est un, qu'on appelle Pythagorique. Mais icy tout cela se trouve & se prouve de source, comme l'on voit dans les exemples précédens sous les signes ☉ & ☉.

Figura 2.1 Texto original de la obra "Explication de l'Arithmétique Binaire", de Gottfried Wilhelm Leibniz.

Fuente: Real academia de Inglaterra. (s.f.). Escaneo del sistema binario de Leibniz. Recuperado 26 noviembre, 2018, de https://commons.wikimedia.org/wiki/File:Leibniz_binary_system_1703.png

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 siguiente sitio: <http://platea.pntic.mec.es/~lgonzale/tic/binarios/aritmetica.html>. En él encontrarás mayor información sobre aritmética binaria y 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 a^r-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 $N \neq 0$ 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

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

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 sustracción. 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 siguiente sustracción 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:

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

Ignorar el
acarreo



Finalmente, obtenemos

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

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

$$10^3 - 1 - 327 = 672 \text{ (con } p = 3), \text{ y}$$
$$10^{-m} = 10^0 = 1 \text{ (con } q = 0)$$

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

$$1 - 10^3 - 0.173 = 0.826 \text{ (con } q = 3), \text{ y}$$
$$10^p = 10^0 = 1 \text{ (con } p = 0)$$

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

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

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

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

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

Complemento a la base 2

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



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

Solución.

$$\begin{array}{r} 0000\ 0101 \\ \text{Complemento a 1} \longrightarrow \\ + \qquad\qquad\qquad 1 \\ \hline \text{Complemento a 2} \longrightarrow 1111\ 1011 \end{array}$$

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

Solución

$$\begin{array}{r} \text{Número original} \qquad = \qquad 01111\ 1111 \\ \\ \text{Complemento a uno} \qquad = \qquad 1000\ 0000 \\ \qquad\qquad\qquad + \qquad\qquad\qquad 1 \\ \qquad\qquad\qquad \hline \text{Complemento a dos} \qquad = \qquad 1000\ 0001 = -127 \end{array}$$

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

Solución

$$\begin{array}{r} \text{Número original} \quad = \quad 1000\ 0010 \\ \\ \text{Complemento a uno} = \quad 0111\ 1101 \\ \qquad\qquad\qquad + \qquad\qquad\qquad 1 \\ \qquad\qquad\qquad \hline \text{Complemento a dos} = \quad 0111\ 1110 = (+126)_{10} \end{array}$$



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

Solución.

$$\begin{array}{rcl} \text{Número original} & = & 101001 \\ \\ \text{Complemento a uno} & = & 010110 \\ & + & 1 \\ & \hline \text{Complemento a dos} & = & 010111 \end{array}$$

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.

$$\begin{array}{r} 377 \\ - 027 \\ \hline 350 \text{ --complemento a 7} \end{array} \qquad \begin{array}{r} 350 \\ + 1 \\ \hline 351 \text{ -----complemento a 8} \end{array}$$

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

$$\begin{array}{r}
 377 \\
 - 256 \\
 \hline
 121 \text{ --complemento a 7}
 \end{array}
 \qquad
 \begin{array}{r}
 121 \\
 + 1 \\
 \hline
 122 \text{ -----complemento a 8}
 \end{array}$$

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 $(27)_{16}$ (Obtener su negativo).

Solución

$$\begin{array}{r}
 FF \\
 - 27 \\
 \hline
 D8 \text{ Complemento a 15}
 \end{array}
 \qquad
 \begin{array}{r}
 D8 \\
 + 1 \\
 \hline
 D9 \text{ - Complemento a 16}
 \end{array}$$

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



Ejemplo. Obtener el complemento a 16 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:

$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$	$\begin{array}{r} 110 \\ + 1 \\ \hline 111 \end{array}$
$\begin{array}{r} 11 \\ + 1 \\ \hline 100 \end{array}$	$\begin{array}{r} 1000 \\ + 1 \\ \hline 1001 \end{array}$
$\begin{array}{r} 101 \\ + 1 \\ \hline 110 \end{array}$	$\begin{array}{r} 1010 \\ + 1 \\ \hline 1011 \end{array}$

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



$$0 - 0 = 0.$$

$$0 - 1 = 1 \text{ (con acarreo negativo de 1)}$$

$$1 - 0 = 1.$$

$$1 - 1 = 0.$$

Veamos un ejemplo de resta

Restamos $17 - 10 = 7$

10001

-01010

———

01111

Fuente:

García Álvarez José Antonio E. “*Así funciona el sistema numérico binario*” Recuperado el 11 de octubre de 2018 de:

http://www.asifunciona.com/informatica/af_binario/af_binario_5.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 $(1111)_2$.



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

Tabla 2.2 Interpretación del signo.

Fuente: Villa Martínez Héctor Antonio (2016) *Representación de números en binario*. Universidad de Sonora (Unison). Recuperado el 26 noviembre de 2018, de:

https://www.researchgate.net/publication/287646222_Representación_de_numeros_en_binario

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 $(0000)_2$ o $(0)_{10}$ 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 $(0000)_2$ o $(0)_{10}$ tendremos ahora el número $(1111)_2$ o $(-1)_{10}$ y arriba del $(0000)_2$ el número $(0001)_2$ o $(+1)_{10}$.

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 2.3. Conversión - Decimal, Hexadecimal, Octal, Binario. (s/a), (s.f.). Recuperado 26 noviembre, 2018, de <https://ascii.cl/es/conversión.htm>

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

$$\begin{array}{r} 0\ 0\ 0\ 1\ 0 \\ + \quad \quad 1 \\ \hline 0\ 0\ 0\ 1\ 1 \end{array}$$



Finalmente realizamos la operación de suma

Comprobación

$$\begin{array}{r}
 0011 \\
 \hline
 10001 \\
 + \\
 00011 \\
 \hline
 10100
 \end{array}
 \qquad
 \begin{array}{r}
 (17)_{10} \\
 + \\
 (-29)_{10} \\
 \hline
 (-12)_{10}
 \end{array}$$

↑
Bit de Acarreo

(Caso b) Sumando mayor que el sustraendo

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

Solución.

Comprobación

$$\begin{array}{r}
 0000\ 1111 \\
 + \\
 1111\ 0101 \\
 \hline
 10000\ 0100
 \end{array}
 \qquad
 \begin{array}{r}
 (+15)_{10} \\
 + \\
 (-11)_{10} \\
 \hline
 (+04)_{10}
 \end{array}$$



Para el caso de números fraccionarios

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

Solución

$$\begin{array}{r}
 0101.1000 \\
 + \quad 0011.0100 \\
 \hline
 1000.1100
 \end{array}
 \qquad
 \begin{array}{r}
 (5.5)_{10} \\
 + (3.25)_{10} \\
 \hline
 (8.75)_{10}
 \end{array}$$

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

Solución

$$\begin{array}{r}
 0011.0100 \\
 \hline
 1000.1100
 \end{array}
 \qquad
 \begin{array}{r}
 (3.25)_{10} \\
 \hline
 (8.75)_{10}
 \end{array}$$

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

Solución.

$$\begin{array}{r}
 (3.50)_{10} = 0011.1000 \\
 \quad 1100.0111 \text{ ----- Complemento a 1} \\
 + \quad \quad 1 \text{ ----- Complemento a 2} \\
 \hline
 (-3.50)_{10} = 1100.1000
 \end{array}$$

Luego realizamos la suma

Decimal	Binario
$(5.75)_{10}$ $+ (-3.50)_{10}$ <hr style="width: 100px; margin: 0 auto;"/> $(2.25)_{10}$	0101.1100 $+ 1100.0100$ <hr style="width: 100px; margin: 0 auto;"/> $1 \ 0010.0100$ \uparrow 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 0 0000 000 00 0000 00 000 0101 0	120	78

	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.



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

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

Sacamos el complemento a 15

$$\begin{array}{r} F \ F \\ - \\ 0 \ D \\ \hline F \ 2 \\ + \ 1 \\ \hline F \ 3 \end{array}$$

Sacamos el complemento a 15

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

Finalmente, realizamos la suma

$$\begin{array}{r} F \ 5 \\ + \\ F \ 3 \\ \hline 1 \ E \ 8 \end{array}$$

Bit de signo



Comprobación

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

$$\begin{array}{r}
 F\ 3 \\
 - \\
 E\ 8 \\
 \hline
 1\ 7 \\
 + \\
 1 \\
 \hline
 1\ 8
 \end{array}$$

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

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

Solución.

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

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

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

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



$$\begin{array}{r}
 \text{F E E E} \\
 - \\
 \text{4 D F 5} \\
 \hline
 \text{B 2 0 A} \\
 + \\
 \quad \quad \quad 1 \\
 \hline
 \text{B 2 0 B}
 \end{array}
 \qquad
 \begin{array}{r}
 \text{F E E E} \\
 - \\
 \text{2 A F 7} \\
 \hline
 \text{D 5 0 8} \\
 + \\
 \quad \quad \quad 1 \\
 \hline
 \text{D 5 0 9}
 \end{array}$$

Realizamos la suma de los complementos

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

↑
Bit de signo

Comprobación:

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

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



Ejemplo. Realizar la suma.

Solución.

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

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

complemento a 15
del número 4DF5

complemento a 15
del número 2AF7

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

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

Realizamos la suma de los complementos

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

Bit de signo



Finalmente la suma de

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

$$\begin{array}{r}
 B\ 2\ 0\ B_{16} \\
 + \\
 D\ 5\ 0\ 9_{16} \\
 \hline
 8\ 7\ 1\ 4_{16}
 \end{array}$$

Comprobación

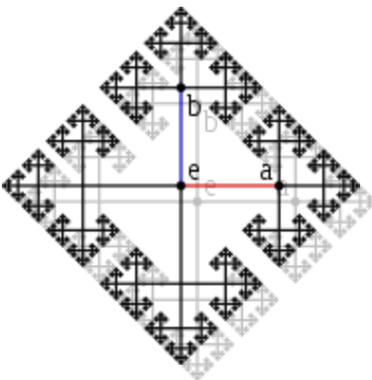
Para comprobar el resultado, obtenemos el valor absoluto del resultado de la suma de la manera siguiente

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

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 en las computadoras hasta los años cuarenta 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. La 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, como 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 DE LA UNIDAD

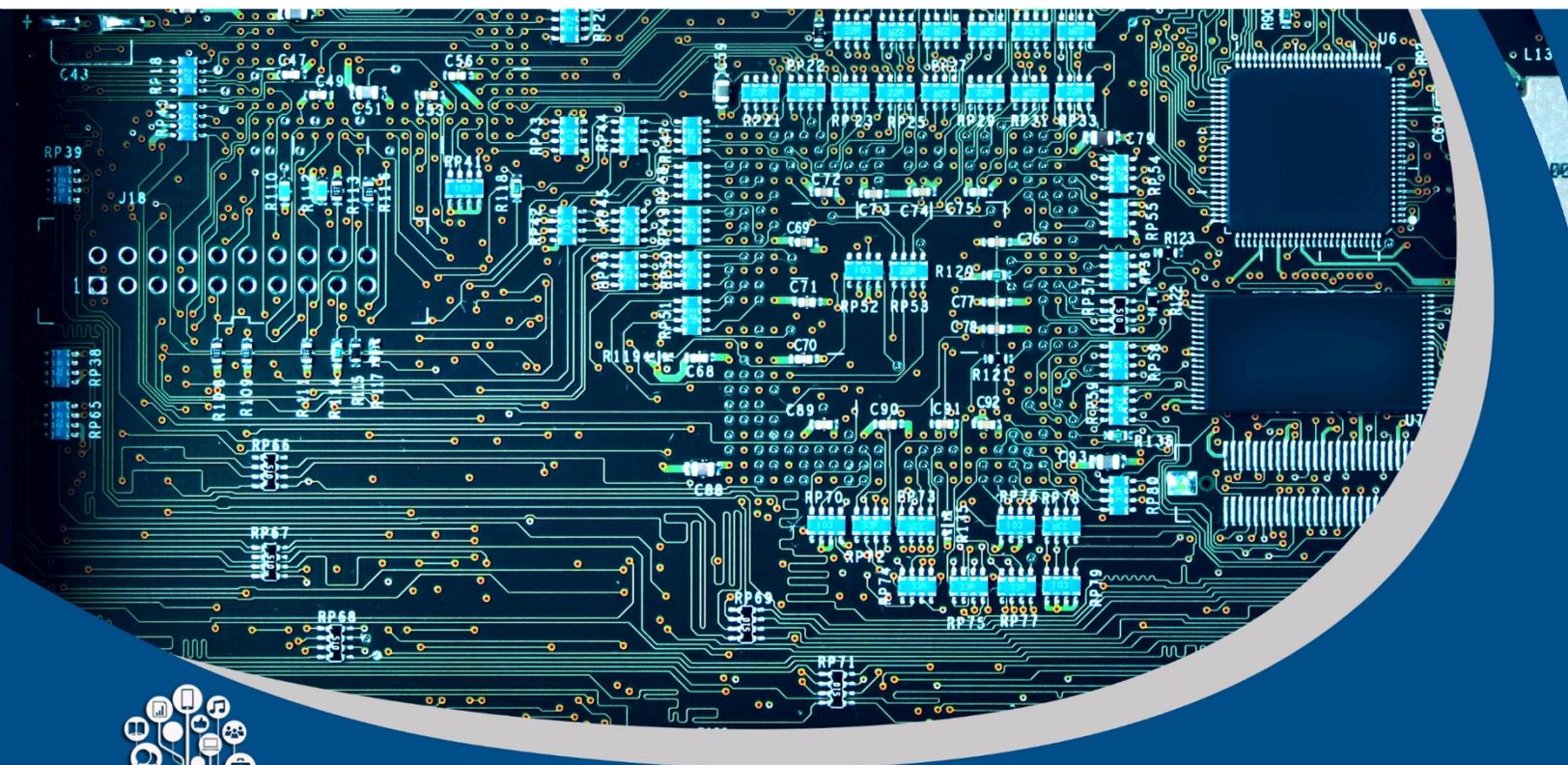


BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 3 CÓDIGOS





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 físicas que se manejan en el mundo real, conocidas como señales analógicas, tienen que ser representadas en los sistemas informáticos empleando solamente los símbolos uno y cero. Se han desarrollado diferentes formas de codificar estas señales, sin embargo, podemos generalizar que solo utilizamos dos tipos de caracteres en las computadoras, numéricos y alfanuméricos. El primer tipo ya se ha visto en la unidad dos, la forma de realizar operaciones con ellos. 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 (números enteros, signados), generalmente las unidades de Entrada/Salida (a través de sus periféricos) reciben o envían información a la computadora en forma decimal o alfanumérica. 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 forma como operan los transistores de los circuitos integrados; el sistema binario puede ser fácilmente implementado en cualquier sistema que considere solo dos modos de acción, así 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 solo 2 estados o condiciones, hace posible la codificación de caracteres numéricos utilizando el código binario.

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

Tabla 3.1 Equivalencia de los números decimales en el sistema binario.



Fuente: (s/a) (s/f). Tabla de conversión - Decimal, Hexadecimal, Octal, Binario.
Recuperado el 26 noviembre de 2018, de: <https://ascii.cl/es/conversión.htm>

3.1.2. BCD (Binary Coded Decimal)

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

Tabla 3.2: Código BCD (Mano, 1986: 17)

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

Tabla 3.3 Elaboración propia.

Puede verse que cada cifra decimal requiere un conjunto binario equivalente de cuatro bits o *nibble*. Para especificar un número, el código BCD requiere más posiciones que el sistema decimal. 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.

Cabe aclarar que esta codificación en BCD no es lo 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 los números en el sistema octal y 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

Tabla 3.4 Elaboración propia.

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).

$$\begin{array}{r}
 8 \\
 + 7 \\
 \hline
 15
 \end{array}
 \qquad
 \begin{array}{r}
 1000 \\
 + 0111 \\
 \hline
 1111
 \end{array}
 \qquad
 \begin{array}{r}
 1000 \\
 + 0111 \\
 \hline
 1111
 \end{array}
 \rightarrow \text{No es un carácter aceptable en BCD (15 es 0001 001)}$$

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 razones del empleo de este código son:

- a) Facilita la representación de 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

Tabla 3.5 Codificación decimal en binario y en Exceso en tres (Mano, 1986: 17)



Como su nombre lo indica, cada carácter codificado en exceso en tres es tres unidades mayores que en BCD. Así, seis o 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,

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

Se lee 0100 0101 o 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 físicos (reales) 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 solo tiene carácter ilustrativo, es decir no es ponderada.

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

Tabla 3.6. Código Gray (Mano, 1986:20)

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 solo 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

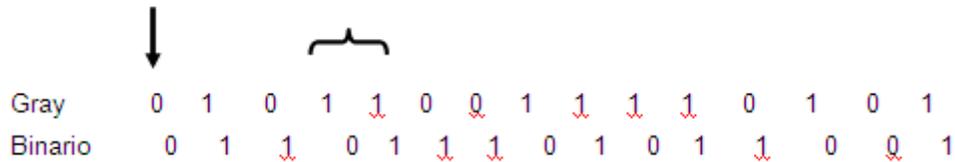
Para hacer la conversión, se repite en la forma binaria el mismo bit que aparece en la forma Gray hasta alcanzar el primer 1, que se repite también. En nuestro ejemplo, la forma Gray empieza por 1, el cual se repite como primer bit (BMS) de la forma binaria. Se sigue repitiendo este bit, en el código 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:

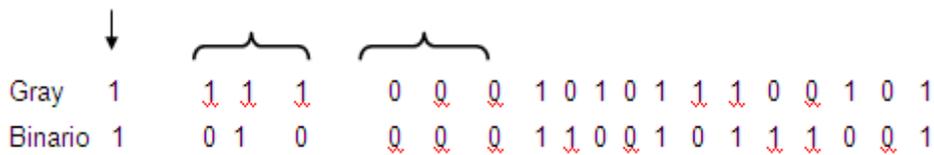


No cambia el bit del binario Cambia el bit del binario



Primer 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 consiste 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 modulo 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 puede estar 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 estandarizado. El 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 (*Extended Binary Coded Decimal Interchange Code*) 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.



Carácter	Código ASCII	Código EBCDIC	Carácter	Código ASCII	Código EBCDIC
Blanco	P010 0000	0100 0000	A	P100 0001	1100 0001
.	P010 1110	0100 1011	B	P100 0010	1100 0010
(P010 1000	0100 1101	C	P100 0011	1100 0011
+	P010 1011	0100 1110	D	P100 0100	1100 0100
S	P010 0100	0100 1011	E	P100 0101	1100 0101
*	P010 1010	0100 1101	F	P100 0110	1100 0110
)	P010 1001	0110 0000	G	P100 0111	1100 0111
	P010 1101	0110 0001	H	P100 1000	1100 1000
/	P010 1111	0110 1011	I	P100 1001	1100 1001
'	P010 1100	0111 1101	J	P100 1010	1101 0001
	P010 0111	0111 0001	K	P100 1011	1101 0010
=	P010 1101	0111 0001	L	P100 1100	1101 0011
0	P010 0000	1111 0000	M	P100 1101	1101 0100
1	P010 0001	1111 0001	N	P100 1110	1101 0101
2	P010 0010	1111 0010	O	P100 1111	1101 0110
3	P010 0011	1111 0011	P	P101 0000	1101 0111
4	P010 0100	1111 0100	Q	P100 0001	1101 1000
5	P010 0101	1111 0101	R	P100 0010	1101 1001
6	P010 0110	1111 0110	S	P100 0010	1110 0001
7	P010 0111	1111 0111	T	P100 0100	1110 0010
8	P010 1000	1111 1000	U	P100 0101	1110 0011
9	P010 1001	1111 1001	V	P100 0110	1110 0101
			W	P100 0111	1110 0110
			X	P100 1000	1110 0111
			Y	P100 1001	1110 1000
			Z	P100 1010	1110 1001

Tabla 3.7 Códigos ANSCII y EBCDIC (Mano, 1986: 22)

donde:

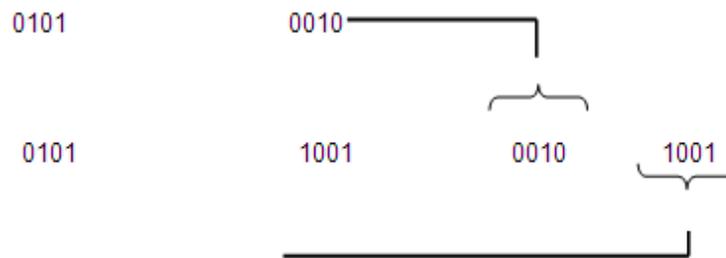
P es un bit de paridad.

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).

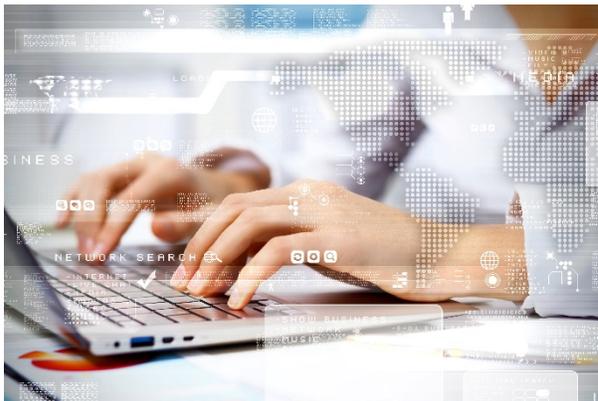
En los equipos de entrada/salida se utiliza el código de ocho niveles para representar los caracteres. Una vez introducidos en la computadora, los caracteres pueden ser tratados del modo más conveniente para las distintas operaciones. Por

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 aplicado el término descriptivo "*byte*" a un grupo de 8 bits que representan una palabra digital. La información se procesa por bytes en lugar de por bits, en el interior de una computadora. El byte (octeto: palabra digital de ocho bits) 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.

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 es que se creó el Código ASCII Extendido. Debido a las limitaciones del ASCII se establecieron varios códigos de caracteres de 6 u 8 bits, entre ellos el código BCDIC (*Binary Coded Decimal Interchange Code*, de 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 otras codificaciones 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 acordó 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.⁶

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 recepción.

Un inconveniente de este código es que 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 menor orden, b7, y al último se transmite el bit de mayor orden, b0. Este código no facilita el uso de un bit de paridad.

⁶ W3C. (s.f.). *Codificación de caracteres: Conceptos básicos*. Recuperado 27 noviembre, 2018, de <https://www.w3.org/International/articles/definitions-characters/index.es>.



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 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 y al ruido presente en el canal de transmisión y en el medio de almacenamiento. De suma importancia en la selección de medios de almacenamiento en servidores de datos y en sistemas de comunicación, la redundancia de datos es parte del protocolo de seguridad de sistemas de cómputo y de comunicaciones de alto desempeño y operaciones financieras ultrarrápidas.

Si en el envío de una información, codificada, por ejemplo, con la cadena binaria 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 puede tomar para detectar si hubo algún error, es la de agregar a cada símbolo o carácter alfanumérico un bit a la izquierda del mismo; en esquemas prácticos de detección de errores este bit recibe el nombre de "Bit de paridad". Estos bits pueden ser de paridad par o impar, según el método de verificación que se seleccione, mismos que 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:

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

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:

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

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 o 1, se puede demostrar que:

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

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

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

Tabla 3.8 Paridad en el código BCD (Mano, 1986: 20)

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, 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. Este es un código ponderado de 7 bits cuya distancia mínima es dos y permite la detección de errores de un bit. 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.

Mencionábamos que 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

Tabla 3.9 Código Biquinario⁷

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 de la tabla anterior 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 electrónicamente 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.

⁷ Fuente: Esacademic.com. (s.f.). *Código biquinario*. Recuperado: 27 noviembre, 2018, de <http://www.esacademic.com/dic.nsf/eswiki/278151>



	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

Tabla 3.10 Comparación biquinario-decimal.

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

La importancia de los códigos para la detección de errores en el almacenamiento y transmisión de información es evidente en algunos casos: la velocidad y cantidad de información susceptible que se maneja constantemente en los sistemas financieros y de toma de decisiones no permite eventos erróneos ni, menos aún, que requieran la intervención humana directa, porque no tan solo pueden afectar el patrimonio, sino aún más, la integridad de sistemas y personas. Por esta razón los equipos de alto desempeño tienen interconstruido *hardware* (equipo, circuitos) y protocolos especializados que “al vuelo” detectan comportamientos anómalos y corrigen automáticamente el equívoco, notificando a la fuente de la información la necesidad de reenvío. Los profesionales de la Informática debemos estar conscientes de ello, ya que debemos seleccionar equipos y sistemas, por lo que debemos conocer los riesgos intrínsecos. Por estas razones sabremos que no debemos utilizar una computadora de escritorio para misiones críticas y de alta velocidad (por ejemplo servidores de bases de datos), pues carecen de las prestaciones necesarias interconstruidas para manejar de manera confiable la información; las placas para CPU de las PC de escritorio no vienen equipadas con la circuitería automática para la detección y corrección rápida de errores que evite que la acumulación de estos colapse o inutilice el funcionamiento del sistema de cómputo; es indispensable saber los factores de interés para la misión a desarrollar,



y si el equipo con el que trabajaremos cumple con los requisitos técnicos (dimensionamiento relacionado a las capacidades de procesamiento, rendimiento, almacenamiento y transmisión de la información) para ejecutar confiablemente la tarea computacional a la que lo dedicaremos.



Imagen. Procesadores de tipo CISC (*Complex Instruction Set Computer*) y RISC (*Reduced Instruction Set computer*).⁸

⁸ Para equipos servidores y estaciones de trabajo para misiones críticas, en configuraciones de múltiples núcleos y estaciones de trabajo para misiones críticas, en configuraciones de múltiples núcleos o multiprocesador, procesamiento paralelo o distribuido, son algunas de las muchas opciones que un profesional informático debe conocer en su actuar cotidiano.

Fuente: Isaac P. E. (s.f.). *Una bestia de la potencia: microprocesador SPARC T5*. Recuperado 27 noviembre, 2018, de <http://arquitectologia.blogspot.com/2013/06/una-bestia-de-la-potencia.html>

RESUMEN

Las computadoras digitales manejan la información mediante los símbolos cero y uno, que corresponden a los valores binarios con los que operan internamente los dispositivos electrónicos a partir de los que están construidas. Las reglas de asociación de estos dos valores con la información que manejan las computadoras requiere 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 auto complementado. Es importante señalar que codificar y convertir una cantidad numérica no tiene 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 DE LA UNIDAD



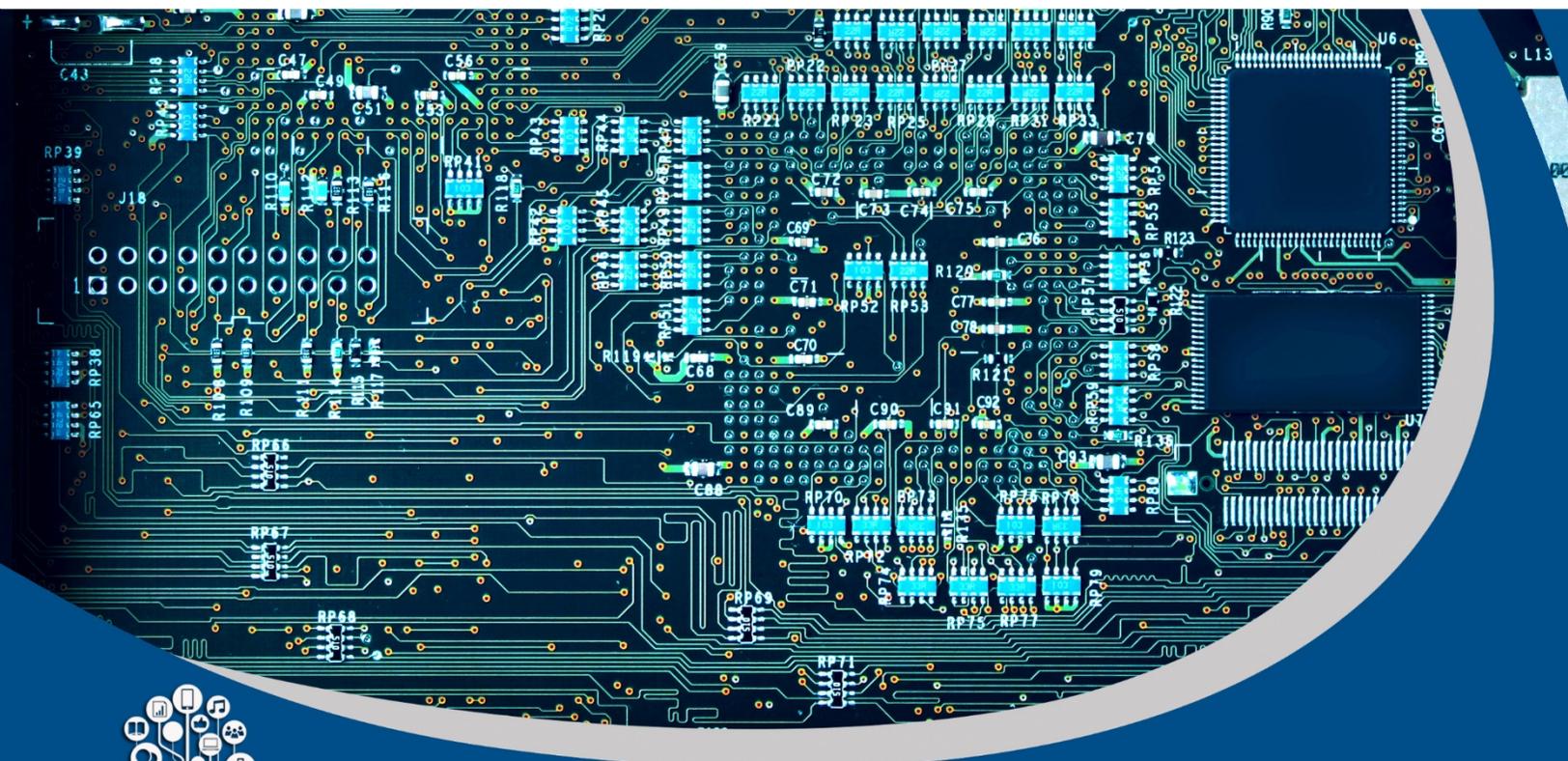
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 4

ÁLGEBRA DE BOOLE





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

Un microprocesador es un dispositivo que debe realizar operaciones aritméticas y lógicas, eso indican el modelo de von Neumann y la máquina de Turing. Esto es natural para la humanidad, sin embargo, los elementos que constituyen una computadora no tienen asociado un comportamiento automático de esta naturaleza a partir de su construcción. Así, con ellos diseñamos y construimos sumadores, multiplicadores, etc. pero para la toma de decisiones la matemática ha desarrollado otras herramientas relacionadas a los conceptos de conjuntos y la lógica binaria, integrados estos conocimientos en el álgebra de Boole. Ésta en su funcionamiento, define elementos operativos conocidos como compuertas lógicas. Así el hombre ha tenido ahora que desarrollar la tecnología que le permita construir compuertas lógicas digitales a través de transistores, y mediante ellas los circuitos digitales empleados en las computadoras. De esta forma las computadoras realizan también operaciones lógicas mediante el álgebra de Boole, aplicada al funcionamiento binario de los circuitos electrónicos denominados compuertas lógicas digitales. El álgebra booleana es fundamental pues permite la sistematización y representación matemática del funcionamiento de los circuitos electrónicos digitales, requeridos para la toma de decisiones en el interior de los microprocesadores. A continuación, presentamos una ventana de tiempo que enmarca tres eventos fundamentales en el desarrollo de esta rama de las matemáticas y que son fundamento en la operación de los circuitos digitales:

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.

Tabla 4.1 Fundamento de operación de circuitos digitales. Elaboración propia.

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

Es importante aclarar que la relevancia de este último tema estriba en que los problemas y procesos en el funcionamiento de circuitos digitales, se modelan mediante estas funciones booleanas y que, para *su construcción eficiente*, éstas deben representarse en su forma mínima, motivo por el cual el proceso de minimización de Karnaugh adquiere relevancia.

4.1. Principios de electrónica básica

La electrónica es una rama de la física que estudia el comportamiento de los electrones en los materiales, con el fin de aprovechar estos comportamientos en el desarrollo de dispositivos artificiales encaminados a ejecutar funciones complejas. De esta manera la electrónica puede construir bulbos, transistores y circuitos integrados de funciones específicas como: sumadores, compuertas lógicas, microprocesadores, etc. De manera general, podemos considerar que la electrónica se dedica al análisis y síntesis de circuitos electrónicos. Esta rama de la física se puede dividir en tres áreas: Analógica, Digital e Industrial. La electrónica analógica se enfoca en operar con dispositivos que tienen como entradas señales provenientes del mundo real en forma directa y, en consecuencia, pueden tomar un conjunto muy extenso de valores, por ejemplo, cuál es la temperatura de un enfermo, el nivel de



humedad en un invernadero, etc. La Electrónica Digital es aquella que trabaja con señales eléctricas discretas, esto es, que la señal con la que opera el circuito electrónico solo puede tomar un conjunto reducido de valores, por ejemplo, si hay agua o no; si la luz de la habitación está encendida o no, la clave de acceso a un local, etc. La electrónica digital es la herramienta principal para el diseño y construcción de las unidades que constituyen una computadora digital, por ejemplo, la unidad aritmética-lógica, compuertas lógicas, etc.

La electrónica industrial opera con señales del mundo real, pero las utiliza para el control de procesos y toma de decisiones (esto implica un universo de decisión

limitado), por lo que podemos considerar que es una combinación de la electrónica analógica y la electrónica digital.

De la presentación previa y del capítulo de estos apuntes podemos recordar que la electrónica digital puede ser modelada por eventos de tipo binario, es decir, que pueden tomar un valor de dos alternativas posibles, por lo que es necesario establecer un marco de referencia matemático y lógico para poder utilizar estas capacidades de funcionamiento, por lo que a continuación abordaremos estas temáticas.

4.1.1. Lógica binaria

Por la naturaleza de los dispositivos electrónicos que la constituyen, la electrónica digital solo utiliza dos estados de operación: cero “0” lógico o uno “1” lógico.

Adicionalmente, esta lógica puede trabajar de dos formas:⁹



Tabla 4.2 Lógica binaria. Formas en que trabaja. Elaboración propia.

⁹ Si deseas consultar más información sobre este tema, se sugiere consultar el siguiente sitio: <http://platea.pntic.mec.es/~lgonzale/tic/binarios/logica.html>



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. Para el desarrollo de la computación el álgebra de Boole es fundamental y, a partir de esta invención, se ha incrementado su uso en el campo de la electrónica digital, donde se utiliza para el diseño y construcción de circuitos lógicos combinación ales y/o secuenciales. El álgebra de Boole puede ser aplicada a diversos tipos de proposiciones de la vida cotidiana como: “es o no es”; “blanco o negro”, “derecha o izquierda”, etc., de esta forma podemos observar que también tiene aplicación directa para describir circuitos digitales cuyo estado puede caracterizarse por 0 o 1. Los signos lógicos 1 o 0 pueden ser los números base del sistema de numeración binario, también pueden identificarse con las condiciones de “abierto” o “cerrado” de los transistores de los circuitos, con condiciones de “verdadero” o “falso” en un programa, entre otras alternativas.

4.2.1. Leyes de De Morgan

El álgebra de Boole parte de un conjunto de teoremas y leyes y en la práctica permiten diseñar y construir circuitos combinacionales y secuenciales. Dicho conjunto de teoremas y leyes se resumen en la tabla “Teoremas de Álgebra de Boole”

Relación	Dual	Propiedad
$AB = BA$ $A(B+C) = AB + AC$ $1A = A$ $A \bar{A} = 0$	$A + B = B + A$ $A + BC = (A+B)(A+C)$ $0 + \bar{A} = \bar{A}$ $A + A = 1$	Commutativa Distributiva Identidad Complemento
$0A = 0$ $AA = A$ $A(BC) = (AB)C$ $\bar{\bar{A}} = A$ $\overline{AB} = \bar{A} + \bar{B}$ $\overline{AB + AC} = \bar{A} \bar{B} \bar{C}$ $A(A+B) = A$	$1 + A = 1$ $A + A = A$ $A + (B+C) = (A+B) + C$ $\overline{\bar{A} + \bar{B}} = \bar{\bar{A}} \bar{\bar{B}}$ $\overline{(A+B)(A+C)(B+C)} = \bar{A} \bar{B} \bar{C}$ $A + \bar{A}B = A$	Teoremas del cero y el uno Idempotencia Asociativa Involución Teorema de DeMorgan Teorema del consenso Teorema de absorción

Tabla 4.3. Teoremas del Álgebra de Boole. Fuente: Donald L. Schilling and Charles Belove. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill. p. 528.



Las leyes de De Morgan son:

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

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

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

Se intercambia el operador + (también conocido como OR) por el operador producto (también conocido como AND) o, si es el caso, intercambiar el operador producto por el operador +.

Se niega cada una de las variables involucradas.

Se niega todo el término.

Para comprender la aplicación de los pasos mencionados anteriormente, mostraremos su aplicación a la segunda ley de De Morgan, para el caso de 2 variables.

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

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

$$\overline{A \overline{B}} \Rightarrow \overline{\overline{A} + B}$$

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

Se niega cada una de las variables:

$$\overline{A \overline{B}} \Rightarrow \overline{\overline{\overline{A}} + \overline{\overline{B}}}$$



Se niega todo el término:

$$\overline{A B} \Rightarrow \overline{\overline{A} + \overline{B}}$$

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

$$\overline{A B} \Rightarrow \overline{\overline{\overline{A} + \overline{B}}}$$

Con lo cual se obtiene el resultado:

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

y de esta manera queda demostrada 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 electrónico que implementa una función básica del álgebra de Boole. La electrónica digital utiliza tres compuertas básicas para la construcción de circuitos: la compuerta OR, AND y NOT y a partir de estas compuertas se crean compuertas complementarias (o negadas como vimos en la demostración del teorema 2) y son: NAND, NOR, OR-exclusiva y NOR-exclusiva, las cuales explicaremos a continuación.

Compuerta AND. La figura 1a muestra su símbolo lógico (al centro), lo que se conoce como “tabla de verdad” (la tabla a la izquierda) y la ecuación algebraica característica de la compuerta lógica (a la derecha).

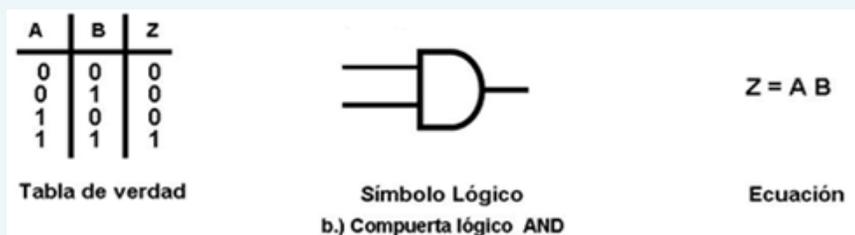


Figura 1a

Compuerta OR

Empezaremos por la lectura de la tabla de verdad, esta considera que la compuerta puede tener dos variables de entrada “A” y “B” y que cada una de ellas puede tomar un valor “0,1”, lo que es cardinalmente representativo de cualquier evento de naturaleza binaria. Un hecho que debe notarse es que el funcionamiento de la compuerta lógica solo depende de estos eventos en su ocurrencia, sin importar para estas condiciones *el cuándo* ocurren, es decir no dependen del tiempo en su funcionamiento. Así podemos observar que para la formación de la tabla se van tomando las posibles combinaciones de ocurrencia para las variables “A” y “B”

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

Tabla de verdad

De esta tabla observamos que para los posibles valores que puedan tomar sus entradas, su salida siempre será cero, excepto cuando ambas variables toman el valor “1”. Esto lo podemos aplicar para cuando ambas condiciones son verdaderas (en lógica positiva), situación la podemos ejemplificar con un caso práctico: yo puedo entrar a mi departamento si la puerta de acceso del edificio y la del departamento se pueden abrir (ambos eventos son ciertos

simultáneamente), entonces puedo entrar a mi departamento; si alguno de ellos no es cierto (alguna de las puertas no se puede abrir), no podré entrar en mi departamento. Como vemos de esta ejemplificación, la versatilidad que presenta esta compuerta de decisión lógica tiene una variedad muy amplia de aplicaciones. En resumen, la compuerta AND presenta en su salida un nivel alto (“1”), solo si sus dos entradas A o B están simultáneamente en nivel alto. La salida tiene un nivel bajo para todos los otros casos.

A partir de la explicación anterior procedemos a mostrar de manera resumida los símbolos, tablas de verdad y ecuaciones de las tres compuertas básicas: OR (+), AND (producto) y NOT (negación).

En la tabla de verdad de la figura Compuertas básicas; en el funcionamiento de estos circuitos digitales el dígito binario 1 representa un nivel alto de voltaje, y el dígito binario 0 un nivel bajo de voltaje.

<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr> <th style="padding: 2px;">A</th> <th style="padding: 2px;">B</th> <th style="padding: 2px;">Z</th> </tr> </thead> <tbody> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;">0</td><td style="padding: 2px;">0</td></tr> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;">1</td><td style="padding: 2px;">1</td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">0</td><td style="padding: 2px;">1</td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">1</td><td style="padding: 2px;">1</td></tr> </tbody> </table> <p style="text-align: center;">Tabla de verdad</p>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	1		$Z = A + B$
A	B	Z															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
<p style="text-align: center;">a.) Compuerta lógico OR</p>	<p style="text-align: center;">Símbolo Lógico</p> 	<p style="text-align: center;">Ecuación</p> $Z = A B$															
<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr> <th style="padding: 2px;">A</th> <th style="padding: 2px;">B</th> <th style="padding: 2px;">Z</th> </tr> </thead> <tbody> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;">0</td><td style="padding: 2px;">0</td></tr> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;">1</td><td style="padding: 2px;">0</td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">0</td><td style="padding: 2px;">0</td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">1</td><td style="padding: 2px;">1</td></tr> </tbody> </table> <p style="text-align: center;">Tabla de verdad</p>	A	B	Z	0	0	0	0	1	0	1	0	0	1	1	1	<p style="text-align: center;">b.) Compuerta lógico AND</p>	<p style="text-align: center;">Ecuación</p> $Z = \overline{A}$
A	B	Z															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr> <th style="padding: 2px;">A</th> <th style="padding: 2px;">Z</th> </tr> </thead> <tbody> <tr><td style="padding: 2px;">0</td><td style="padding: 2px;">1</td></tr> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">0</td></tr> </tbody> </table> <p style="text-align: center;">Tabla de verdad</p>	A	Z	0	1	1	0	<p style="text-align: center;">c.) Compuerta lógico AND</p>	<p style="text-align: center;">Ecuación</p> $Z = \overline{A}$									
A	Z																
0	1																
1	0																
<p style="text-align: center;">Símbolo Lógico</p> 	<p style="text-align: center;">Ecuación</p> $Z = \overline{A}$																



Compuerta AND

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

Tabla de verdad



Símbolo Lógico
b.) Compuerta lógico AND

$$Z = A B$$

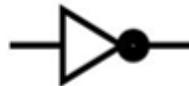
Ecuación

Compuerta NOT

La compuerta más sencilla de las anteriormente ilustradas es la compuerta inversora o NOT. La compuerta inversora solo tiene una entrada ("A") y es aquella en la cual su salida tiene una condición inversa a la que presenta su entrada, esto es si a la salida tengo la ocurrencia de un nivel bajo ("0"), entonces a la entrada presenta debo tener un nivel alto ("1") y viceversa.

A	Z
0	1
1	0

Tabla de verdad



Símbolo Lógico
c.) Compuerta lógico AND

$$Z = \overline{A}$$

Ecuación

Compuerta NOT

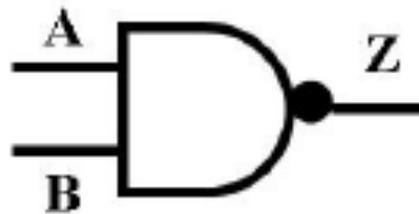
Tabla 4.4 Fuente de las imágenes anteriores: (Mano, 1986: 56)

La correcta combinación de la compuerta NOT con las compuertas AND y OR produce las compuertas complementarias NAND, NOR, OR-exclusiva y NOR-exclusiva, las que se ilustran a continuación. Matemáticamente la justificación de estas puede ser obtenida por medio de la lógica complementaria, sin embargo, desde el punto de vista de construcción física (de los circuitos electrónicos), es que las compuertas NOR y NAND son de fácil implementación lo que disminuye su

costo, disipan menos potencia en su funcionamiento y dan una respuesta de operación rápida.

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

Tabla de Verdad



Símbolo Lógico

$$Z = \overline{AB}$$

Ecuación

2a Compuerta Lógica NAND

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

Tabla de Verdad



Símbolo Lógico

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

Ecuación

2b. Compuerta Lógica NOR

Figura 4.1 Compuertas complementarias.
Fuente: (Mano, 1986: 56)



Compuerta NAND

De las tablas mostradas podemos observar que 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”).

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

Tabla de Verdad

Símbolo Lógico

$$Z = \overline{AB}$$

Ecuación

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).

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

Tabla de Verdad

Símbolo Lógico

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

Ecuación

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 o “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 alto si sus entradas son iguales (ambas son simultáneamente 0 o 1) y presentan un nivel bajo cuando sus entradas son diferentes. La figura 2c. muestra el símbolo lógico, tabla de verdad y la ecuación característica.

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

Tabla de Verdad



Símbolo Lógico

$$Z = A \oplus B$$

$$Z = \bar{A}B + A\bar{B}$$

Ecuación

2.c Compuerta Lógica OR-EXCLUSIVA

Compuerta NOR-Exclusiva

La compuerta NOR-exclusiva es aquella en la cual la salida es un nivel alto ("1") si las entradas son diferentes (son "0" o "1") y presentan un nivel bajo ("0") cuando sus entradas son iguales. La figura 2d muestra el símbolo lógico, tabla de verdad y ecuación característica.

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

Tabla de Verdad



Símbolo Lógico

$$Z = A \oplus \bar{B}$$

$$Z = \bar{A}\bar{B} + AB$$

Ecuación

2.d Compuerta Lógica NOR-EXCLUSIVA

Tabla 4. 5. Compuertas.

Fuente de las imágenes anteriores: (Mano, 1986: 56)



4.2.3. Función booleana

Una función booleana representa el análisis y síntesis de un problema lógico determinado. Una función booleana puede depender de n variables de entrada y resultar en una sola salida (lo cual tiene un sentido real, ya que ante una serie de eventos simultáneos puede haber una sola consecuencia; por ejemplo: *al conducir el auto vi un bache, se atravesó un peatón y cambio la luz del semáforo: acción, frené el auto*).

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 debe ser llevada a su mínima expresión debido a que a la hora de construir el circuito integrado real equivalente se necesita ocupar el menor número de transistores y compuertas; sin embargo, no siempre es fácil de reducir y generalmente exige cierta intuición e ingenio. Se han desarrollado 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 Álgebra de Boole.

Solución:

$$f(A,B,C) = \overline{AB} + C + \overline{\overline{A}CB} + AC(B + BA)$$

Primero marcamos cada uno de los minitérminos (en este caso elementos no negados y separados por “+”)

$$f(A,B,C) = \underbrace{\overline{AB} + C}_I + \underbrace{\overline{\overline{A}CB}}_{II} + \underbrace{AC(B + BA)}_{III}$$

Factorizando el término III (ver “tabla de teoremas del álgebra de Boole”, del tema 4.2.1) y utilizando el teorema de complemento:



$$f(A,B,C) = \underbrace{\overline{AB + C}}_I + \underbrace{\overline{ACB}}_II + \underbrace{AC(B(1+A))}_III$$

y ordenando (propiedad conmutativa) la ecuación:

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

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

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

Con lo cual obtenemos la expresión:

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

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

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

Se obtiene la siguiente expresión:

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

Utilizando la propiedad distributiva:

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

Factorizando y utilizando la propiedad de complemento, se obtiene:

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

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

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

4.3.2. Mapas de Karnaugh

El método de los mapas de Karnaugh es una técnica gráfica que puede utilizarse para obtener los términos mínimos de una función lógica utilizando las variables que le 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.

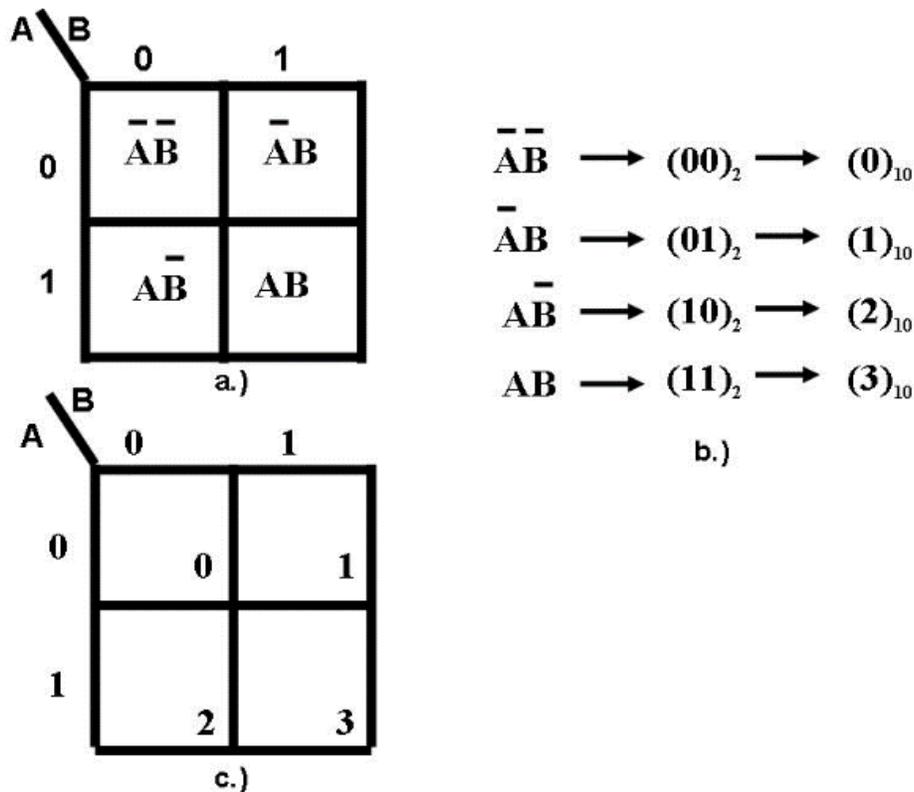


Figura 4.2. Mapa de Karnaugh para 2 variables.
Fuente: Mano, 1986: 76

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: 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.

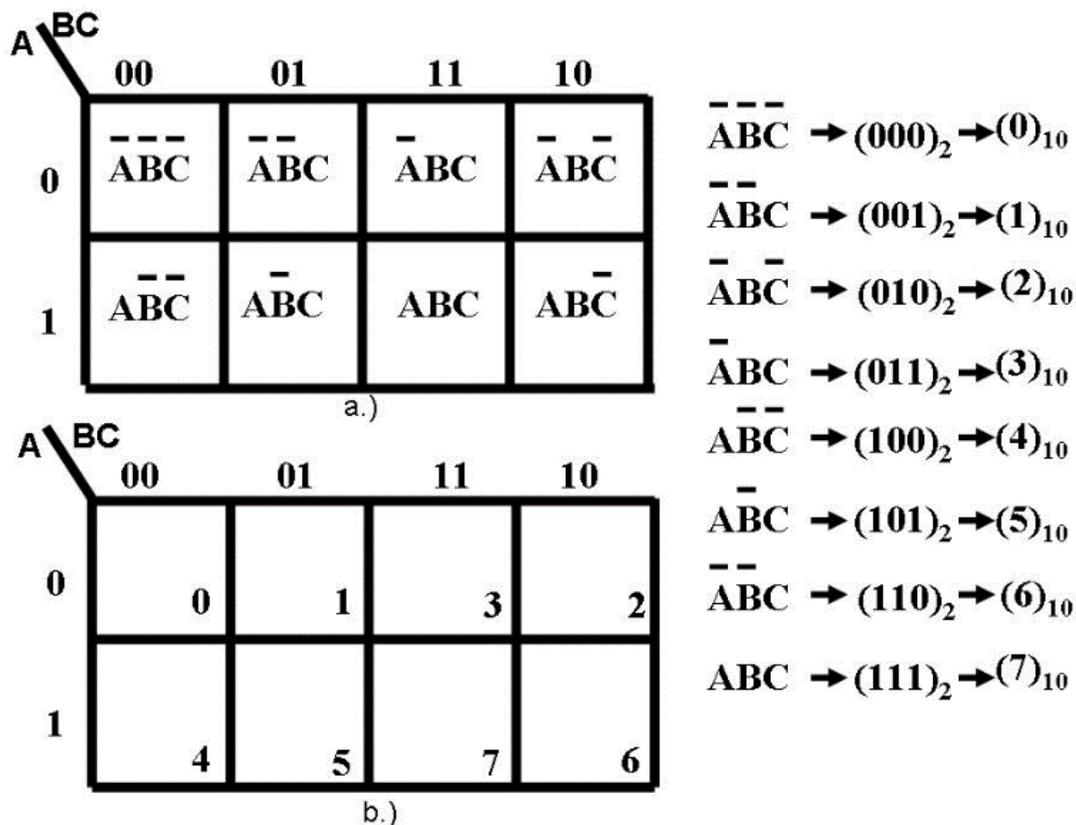


Figura 4.3. Mapas de Karnaugh para 3 variables (Mano, 1986: 77)

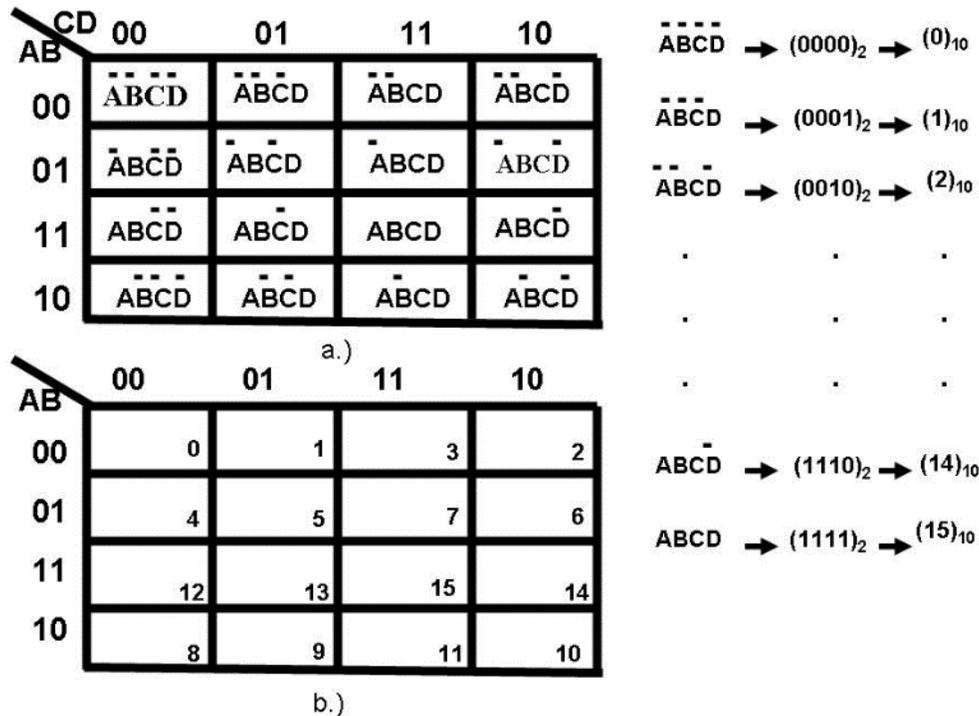


Figura 4.4. Mapas de Karnaugh para 4 variables. (Mano, 1986: 81)

Procedimiento de reducción utilizando mapas de Karnaugh

El proceso de reducción de una expresión booleana utilizando mapas de Karnaugh consiste en la aplicación de los pasos siguientes:

Paso 1

Definir el tamaño del Mapa de Karnaugh.

El tamaño del mapa de Karnaugh se define en función del número de las variables de entrada (n) que forman la expresión booleana, por ejemplo, si se tienen 3 ($n=3$) variables, el tamaño del mapa de Karnaugh es de 8 (2^n) celdas contiguas, si tuviera cuatro variables de entrada ($n = 4$) se forma o construye un Mapa de Karnaugh de 16 celdas ($2^4 = 16$), etc.



Paso 2

Depositar en cada una de las celdas el valor de “1” donde la función es verdadera y el valor de 0 en las celdas donde la función es falsa. *Por claridad únicamente se depositan los “1”s.*

Paso 3

Realizar encierros de cajas o celdas (cuyo contenido sea “1”) adyacentes y contiguos de tamaño $2^n, 2^{n-1}, 2^{n-2}, \dots, 2^0$, 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.

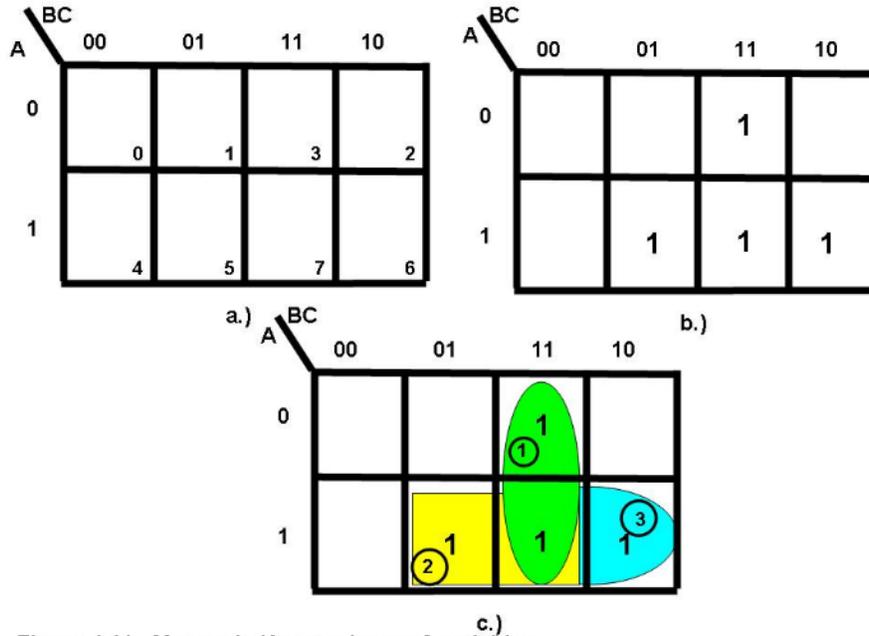


Figura 4.5. Mapas de Karnaugh para 3 variables a-c (Mano, 1986: 80)

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. Por tanto, al final, al agrupar los tres términos, tenemos la función booleana reducida siguiente:

$$f(A, B, C) = AC + BC + AB$$

Paso 5

Por último, a partir de la ecuación reducida, construimos el circuito lógico correspondiente, el cual se muestra en la figura Circuito lógico.

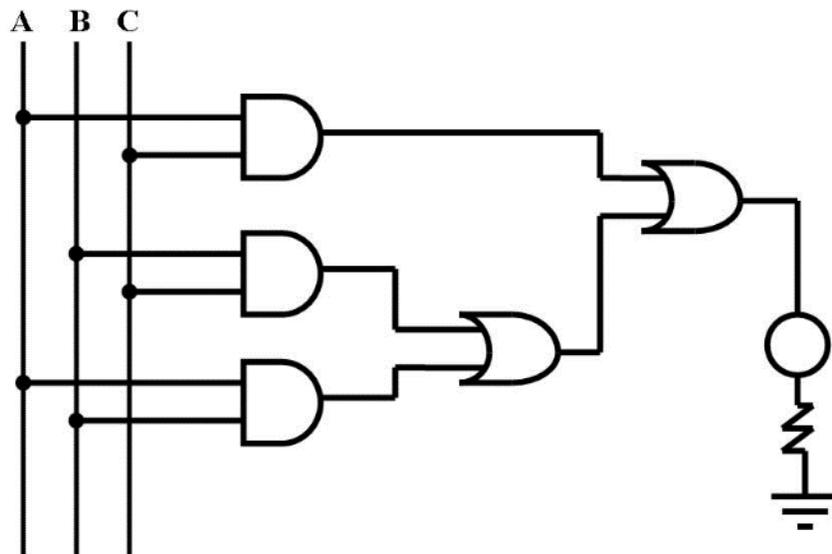


Figura 4.6. Circuito lógico
Fuente: (Mano,1986: 59, 61).

Ejemplo: Utilizando los mapas de Karnaugh, reduce la siguiente función:

Solución:

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

A partir de la función tenemos los términos y su equivalencia en binario y decimal:

$$\overline{A} \overline{B} C = (000)_2 = (0)_{10} \quad A \overline{B} C = (001)_2 = (1)_{10} \quad \overline{A} B \overline{C} = (010)_2 = (2)_{10}$$

$$\overline{A} B C = (100)_2 = (4)_{10} \quad A B \overline{C} = (101)_2 = (5)_{10} \quad A B C = (110)_2 = (6)_{10};$$

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 canónica* (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:

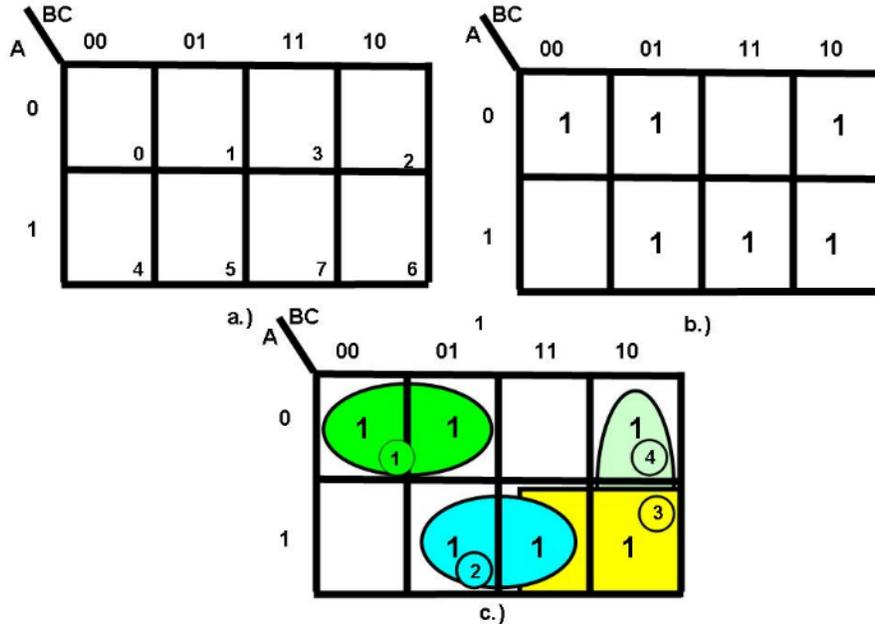


Figura 4.7: Mapas de Karnaugh para 3 variables (Mano, 1986: 80)

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: $B\overline{C}$

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

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**.

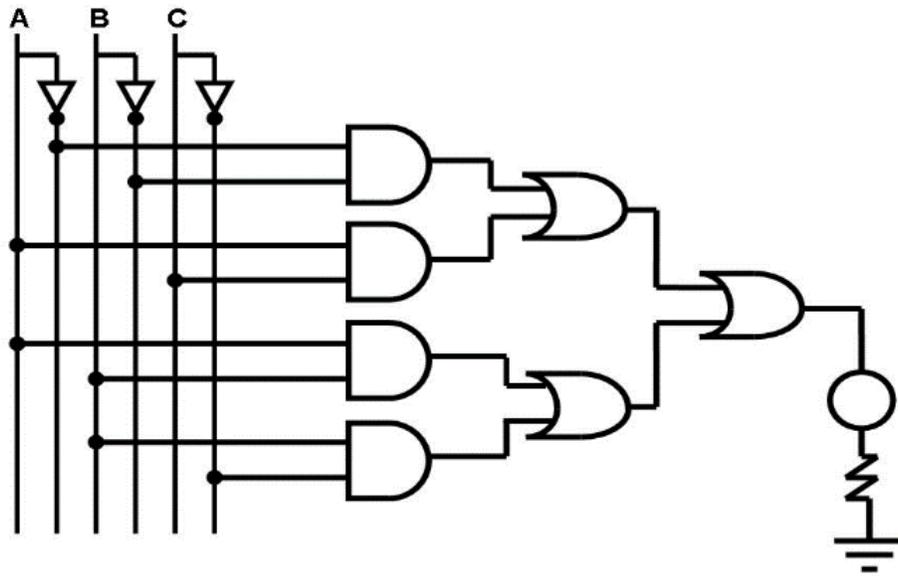
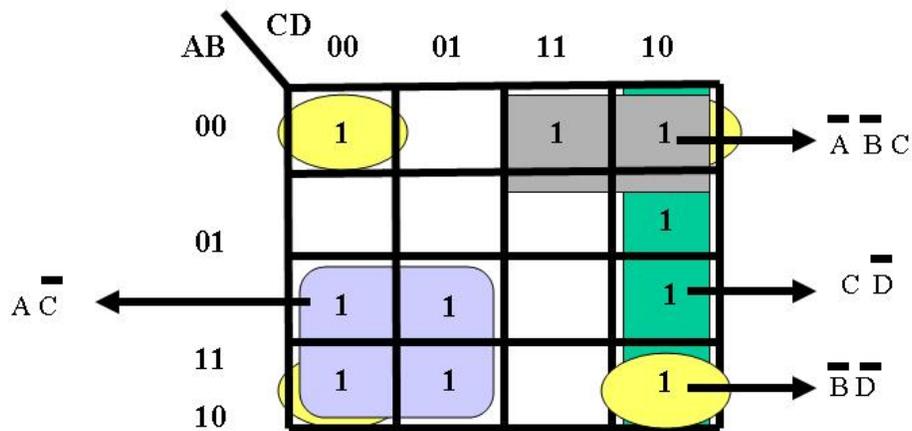


Figura 4.8. Circuito lógico (Mano, 1986: p. 59-61)

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



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

Figura 4.9. Mapa de Karnaugh para 4 variables $f(A, B, C, D)$ (Mano, 1986: 82-83)

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:

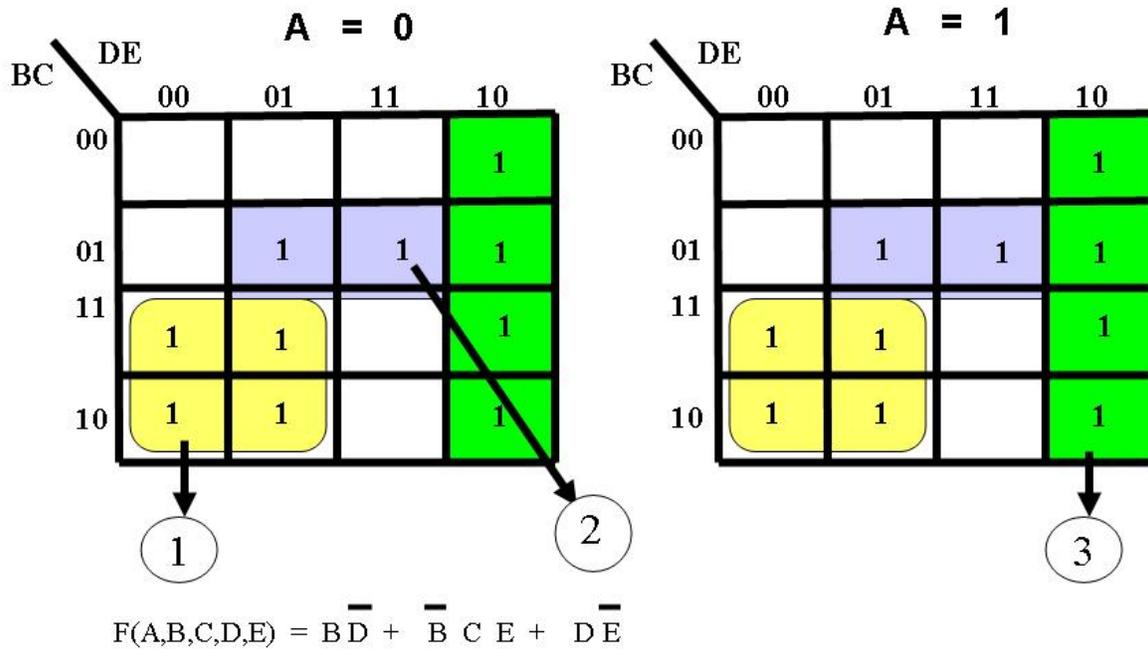


Figura 4.10 Mapa de Karnaugh para 5 variables (Mano, 1986: p. 82-83)



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		
<ul style="list-style-type: none">• El operador OR (O) designado también como + es la representa la operación de “suma binaria” no es una suma en el sentido aritmético, sino logico. $C=A + B$ significa que la variable C será válida cuando alguna de las dos variables A o B sean válidas.	<ul style="list-style-type: none">• El operador AND (Y) designado también como es la representación de la multiplicación “binaria” logica. C es válida cuando las dos variables A y B (ambas) sean válidas.	<ul style="list-style-type: none">• El operador logico 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, sí 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 solo 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 booleanas.



BIBLIOGRAFÍA DE LA UNIDAD



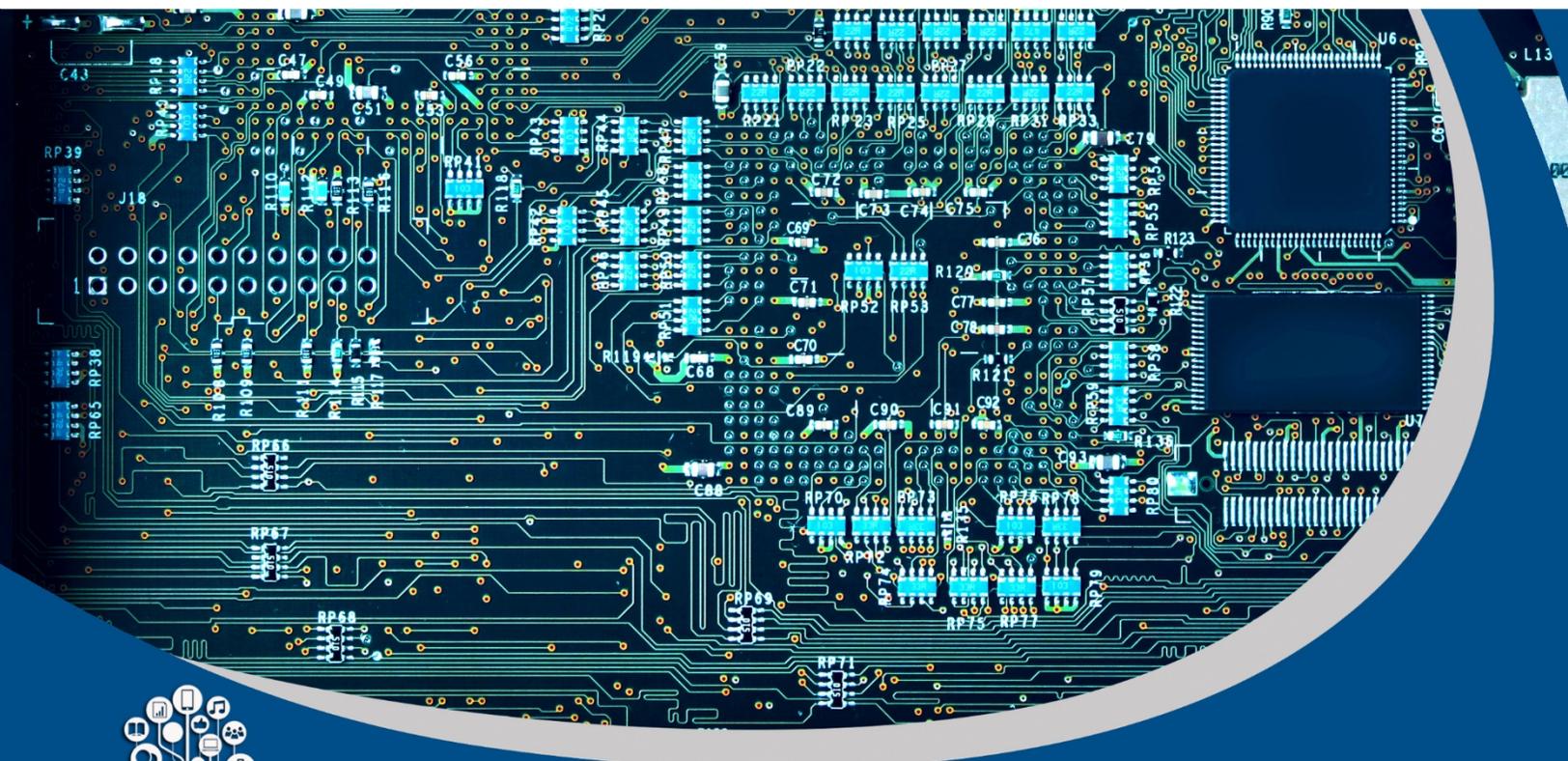
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 5

CIRCUITOS COMBINATORIOS O COMBINACIONALES





OBJETIVO PARTICULAR

El alumno podrá reconocer el funcionamiento y la construcción de sumadores, decodificadores y multiplexores a partir de compuertas básicas, diseñar circuitos combinatoriales mediante compuertas digitales y deducir 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 inmediatamente 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 i_0, \dots, i_n y produce un conjunto de salidas f_0, \dots, f_m , las que dependerán de las funciones lógicas correspondientes. En este tipo de circuito combinacional no existe retroalimentación de las salidas sobre las entradas como en el caso de los circuitos secuenciales (ver Unidad 6).

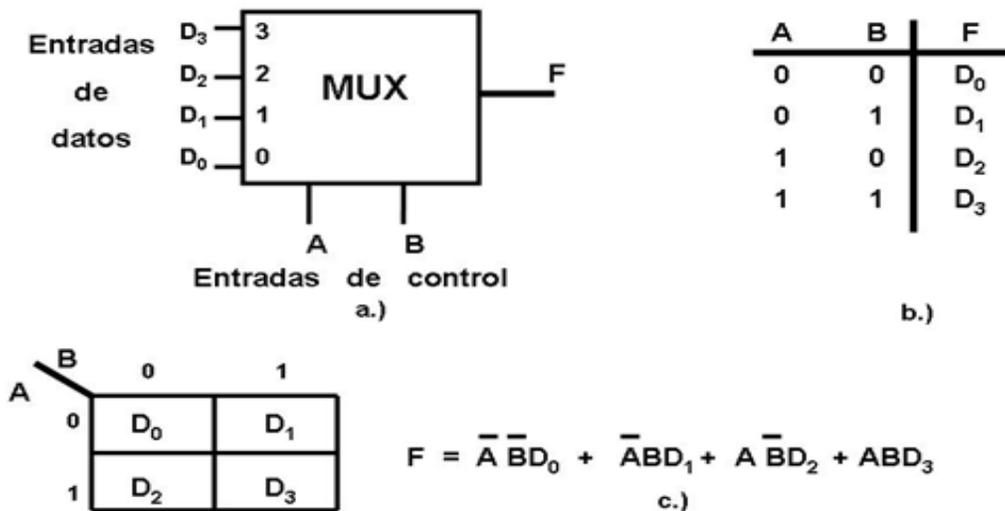


Figura 5.1 Diagrama en bloques de una unidad lógica combinacional (Mano, 1986: p. 121)

Un circuito combinacional recibe entradas y genera salidas en las cuales es habitual considerar como valor bajo, el “0” lógico o 0 Volts, en tanto que se adopta como valor alto, el “1” lógico o 5 Volts. Esta convención no es de uso universal. En los circuitos de alta velocidad y de menor consumo de potencia se tiende a usar menores valores de tensión (3 o menos Volts).

5.1. Multiplexores

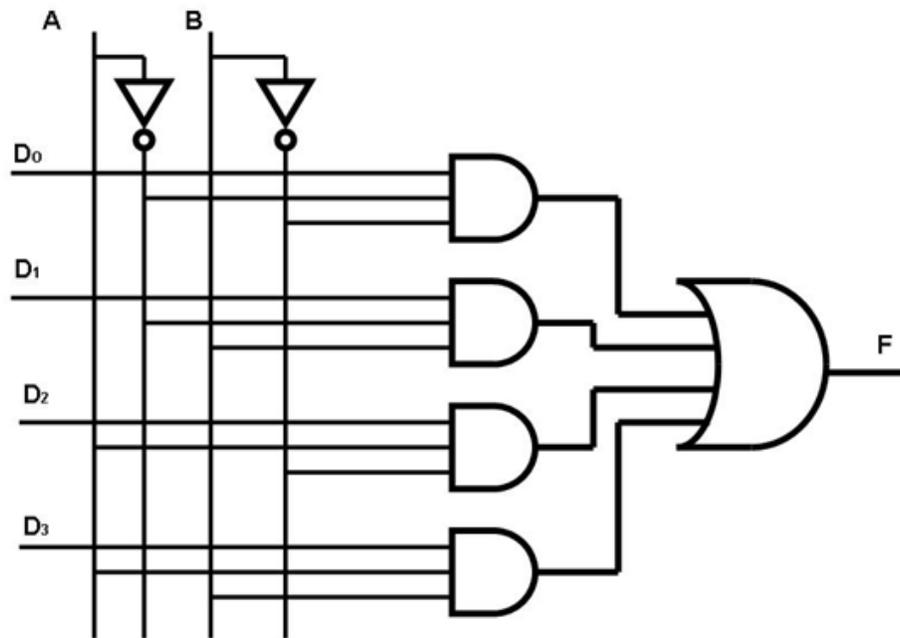
Un circuito multiplexor (MUX) es un elemento que conecta una cantidad dada de entradas a una única salida. La figura Multiplexor 4 entradas 1 salida muestra el diagrama en bloques y la tabla de verdad de un multiplexor de 4 entradas y 1 salida. La salida F adopta el valor correspondiente a la entrada 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, que 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, que 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

Figura 5.2 (Mano, 1986: 182)



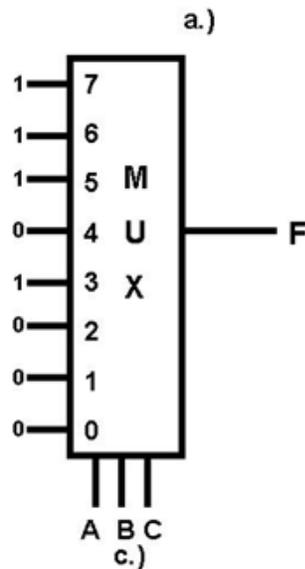
Multiplexor de 4 entradas y 1 salida

d.) Diagrama lógico

Figura 5.3 (Mano, 1986: 182)

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.

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



b.)

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

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

Figura 5.4 Fuente: (Mano, 1986: 183)

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,

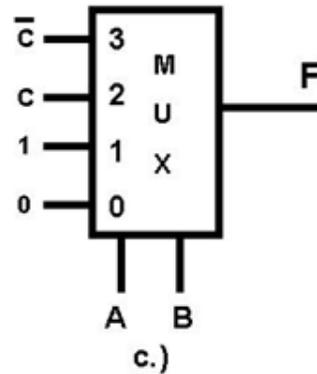
$B = 0$, la función $F = C$ dado que su valor es 0 cuando C es 0 y es 1 cuando C es 1. Finalmente, cuando $A = 1, B = 1$, la función $F = \bar{C}$, por lo tanto, la entrada de datos 11 adopta el valor de C . De esta manera, se puede implementar una función de tres variables usando un multiplexor con cuatro entradas de datos y dos entradas de control.

$$F = A B \bar{C} + A \bar{B} C + \bar{A} B C + \bar{A} B \bar{C}$$

a.)

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

b.)

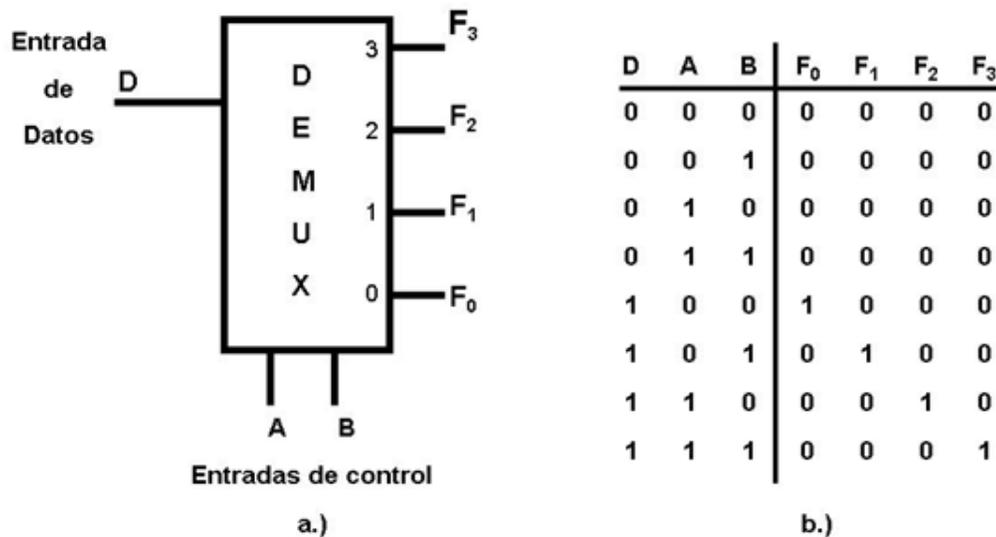


Implementación de una función utilizando un multiplexor de 4 entradas de datos. a.) Función a implementar, b.) Tabla de verdad y c.) Diagrama Lógico.

Figura 5.5 (Mano, 1986: 185)

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 F_i 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.



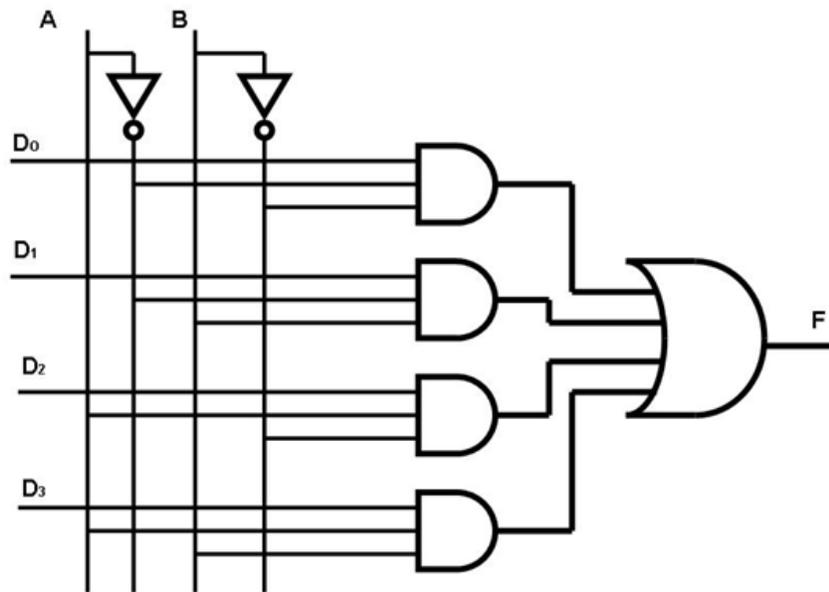
c.)

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

Demultiplexor de 2x4.

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

Figura 5.6 Fuente: (Mano, 1986: 178)



Multiplexor de 4 entradas y 1 salida

d.) Diagrama lógico

Figura 5.7 Fuente: (Mano, 1986: 178)

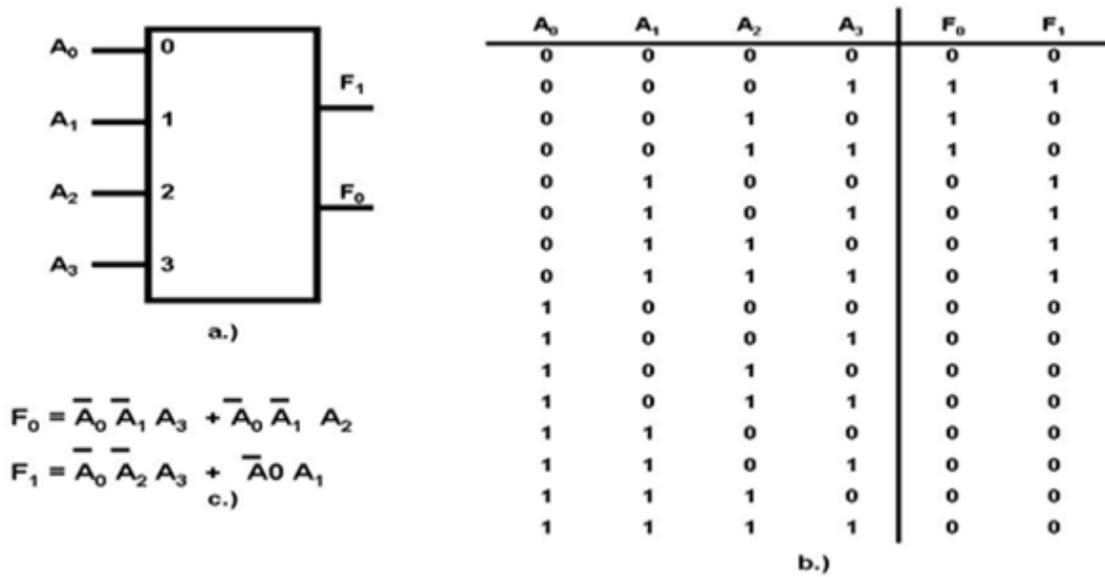


5.3. Codificadores

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 $F_1 = 0$).

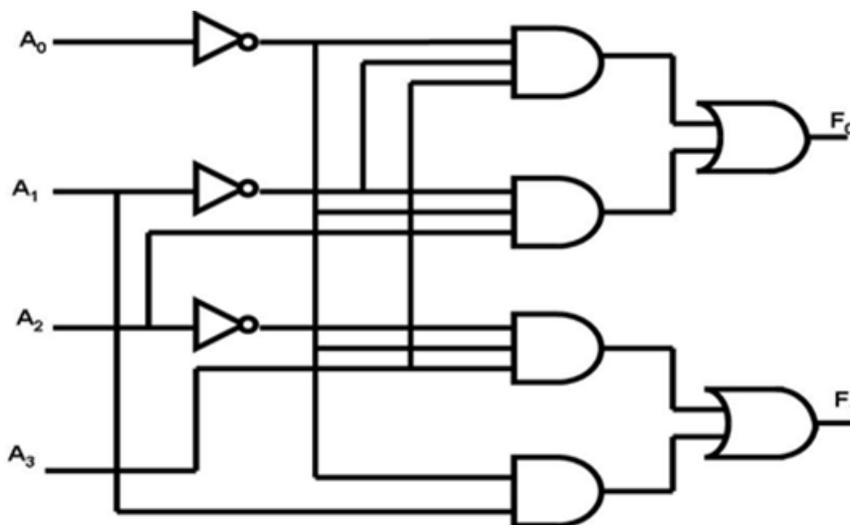
Los codificadores de prioridad se utilizan para arbitrar entre una cantidad de dispositivos que compiten por un mismo recurso, como cuando se produce el intento de acceso simultáneo de una cantidad de usuarios a un sistema de computación. La figura “Codificador de prioridad de 4 a 2”. c) Función de verdad ilustra el diagrama lógico para un codificador de prioridad de 4 entradas y 2 salidas.



Codificador de prioridad de 4 a 2.

a.) Diagrama en bloques, b.) Tabla de verdad, c.) Funciones de salida.

Figura 5.8 Fuente: (Mano, 1986: 165)

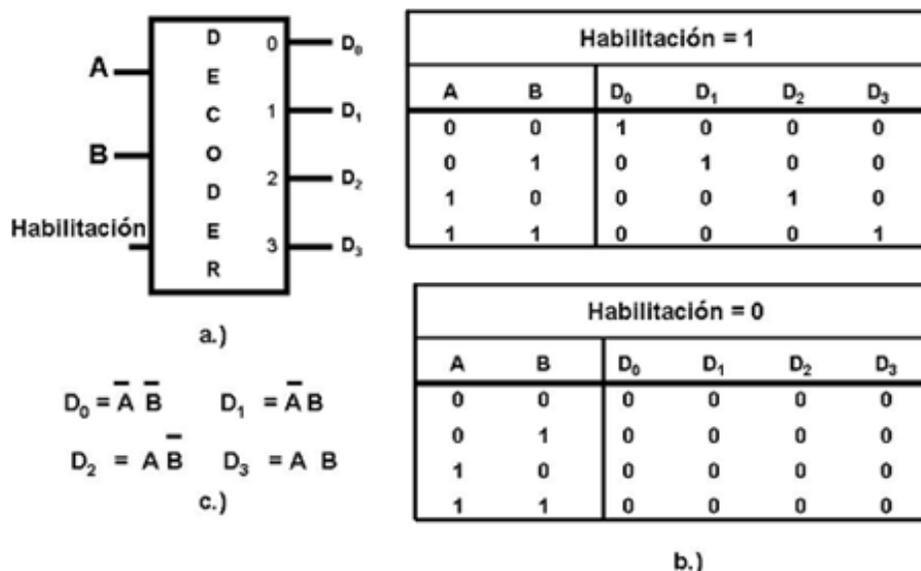


d.) Diagrama lógico.

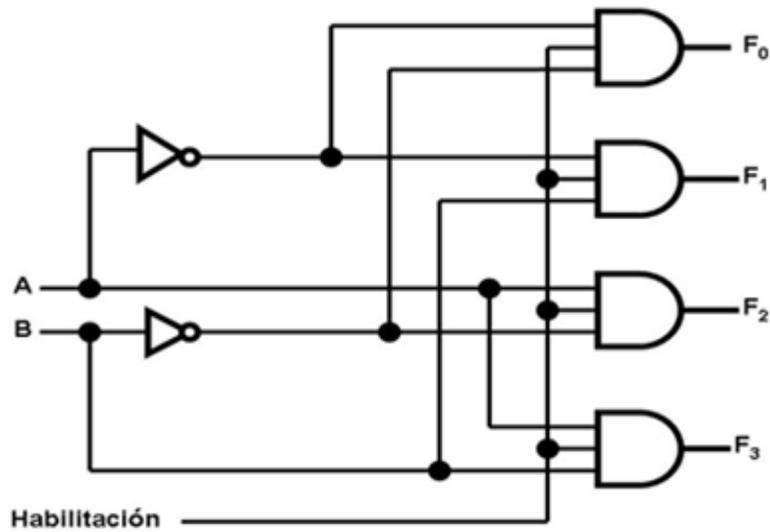
Figura 5.9 Fuente: (Mano, 1986, 179)

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.



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.

Figura 5.10 Elaboración con base en: (Mano, 1986: 175,176)

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.

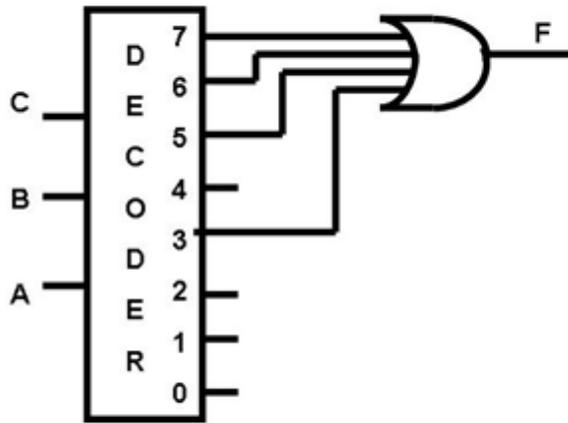


$$F = \bar{A}BC + A\bar{B}C + ABC + A\bar{B}\bar{C}$$

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.)



c.)

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

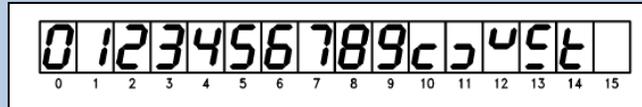
Figura 5.11 Elaboración con base en: (Mano, 1986: 175,176)

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 les 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	0
0	0	0	1	1	0	1	1	0	0	0
0	0	1	0	2	1	1	0	1	1	0
0	0	1	1	3	1	1	1	1	0	0
0	1	0	0	4	0	1	1	0	0	1
0	1	0	1	5	1	0	1	1	0	1
0	1	1	0	6	0	0	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0
1	0	0	0	8	1	1	1	1	1	1
1	0	0	1	9	1	1	1	0	0	1
1	0	1	0	10	0	0	0	1	1	0
1	0	1	1	11	0	0	1	1	0	0
1	1	0	0	12	0	1	0	0	0	1
1	1	0	1	13	1	0	0	1	0	1
1	1	1	0	14	0	0	0	1	1	1
1	1	1	1	15	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.

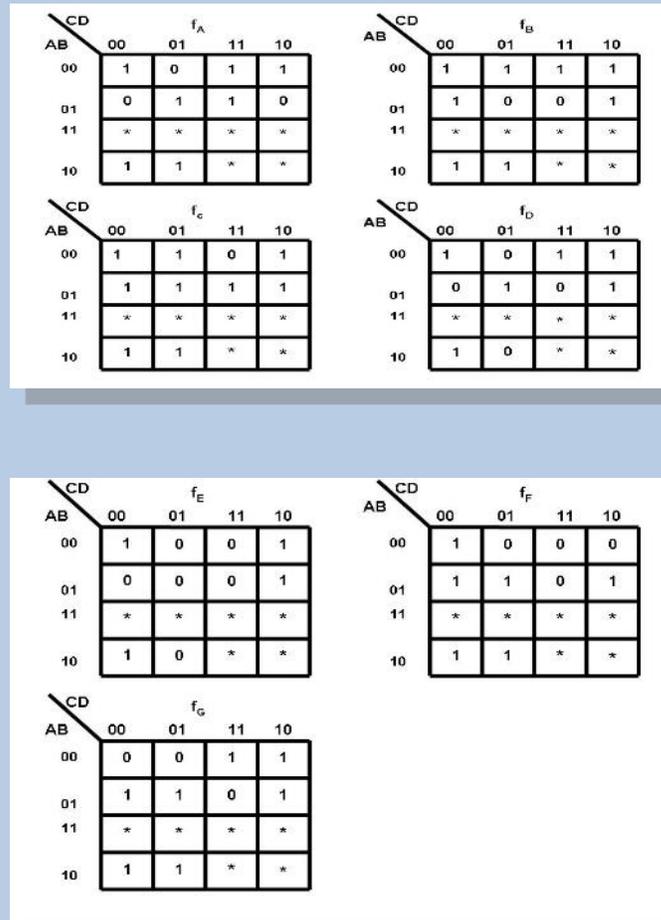


Figura 5.12 Diseño de decodificador.
Elaboración con base en: (Mano, 1986: 163,174)



Se dibuja el diagrama lógico del decodificador 7 segmentos

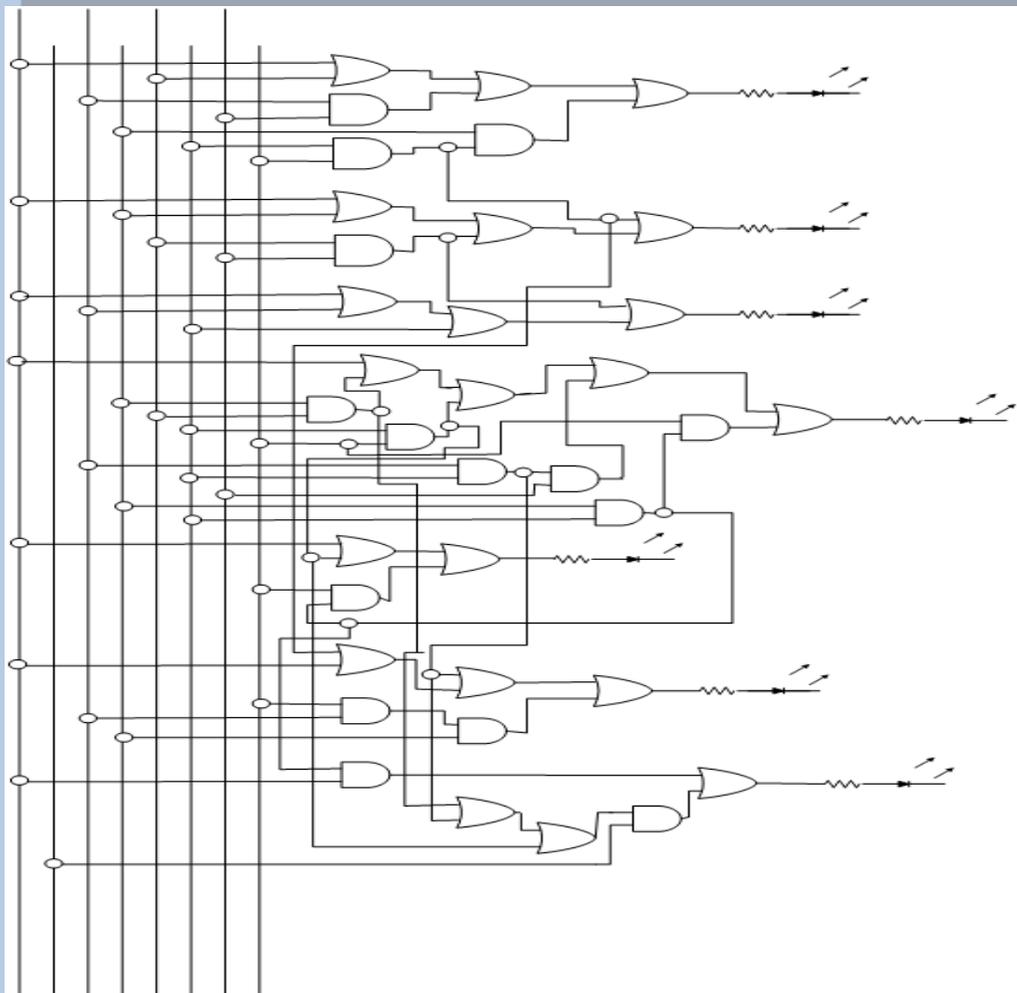


Figura 5.13 Segmentos del diagrama lógico del decodificador 7.
Fuente: (Mano, 1986: pp. 163, 174)



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 $2^2 = 4$ combinaciones

A_0	0	0	1	1
+	+	+	+	+
B_0	0	1	0	1
-----	-----	-----	-----	-----
$C_0 S_0$	0 0	0 1	0 1	1 0
	C S	C S	C S	C S



Donde:

S es el bit del resultado de sumar dos bits, y

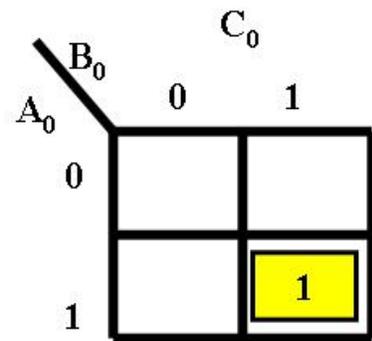
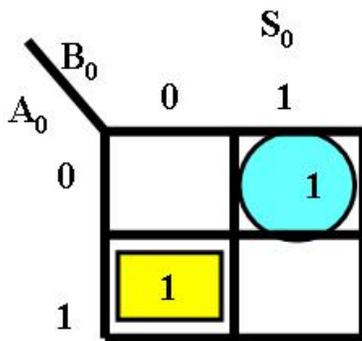
C es el bit de acarreo al momento de sumar dos bits

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

Tabla de verdad: Medio Sumador

A_0	B_0	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

A partir de la tabla de verdad, podemos encontrar la ecuación de salida para el resultado S_0 de la suma de dos bits, así como la ecuación de salida del bit de acarreo C_0 utilizando Mapas de Karnaugh, como se muestra en la figura “Obtención de la ecuación de S_0 y C_0 utilizando mapas de Karnaugh”.



$$S_0 = A_0 \bar{B}_0 + \bar{A}_0 B_0$$

$$C_0 = A_0 B_0$$

$$S_0 = A_0 \oplus B_0$$

Obtención de la ecuación de S_0 y C_0 utilizando mapas de Karnaugh. Mapas de Karnaugh.
 Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*.
 Singapore: McGraw Hill, p. 541.

La implementación (diagrama lógico) de la ecuación del medio sumador para S_0 y C_0 nos quedaría de la siguiente forma:

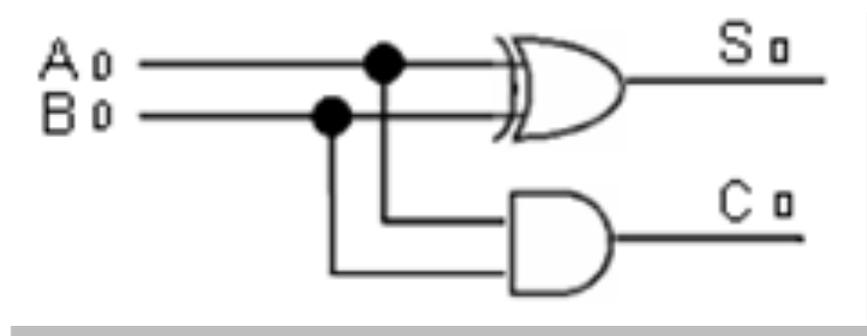


Figura 5.14 Diagrama lógico: Medio sumador
 Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete an integrated*.
 Singapore: McGraw Hill, p. 541.

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
<hr style="width: 100%;"/>				
$C_{i+1} S_i$	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
<hr style="width: 100%;"/>				
$C_{i+1} S_{i+1}$	0 1	1 0	1 0	1 1
	C S	C S	C S	C S

Figura 5.15 Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 670.

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 se presenta 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. Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 670.

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.

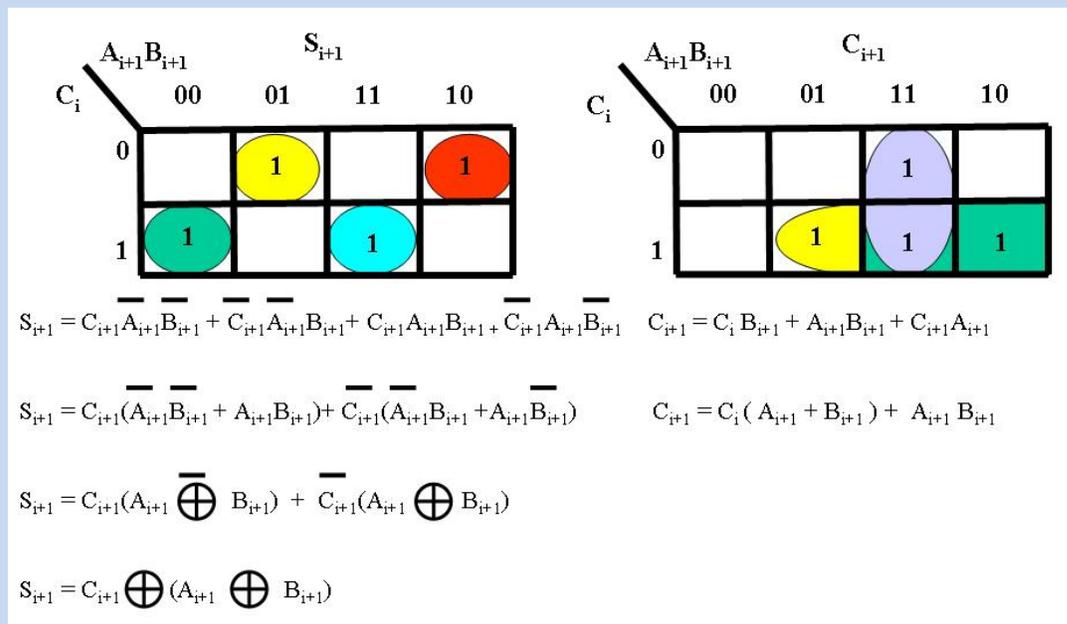


Figura 5.16 Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 670.

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

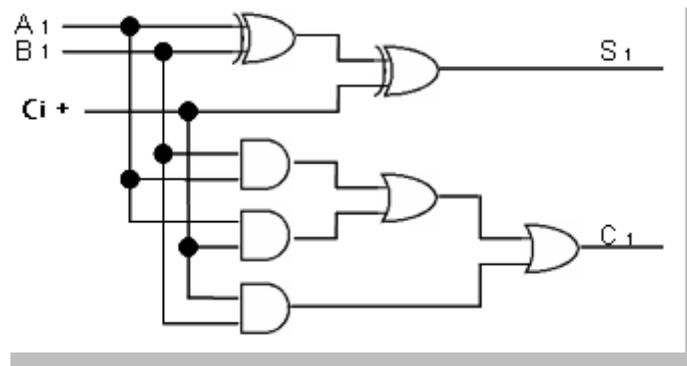


Figura 5.17 Diagrama lógico: Sumador completo
Fuente: (Mano, 1986 162)

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”.

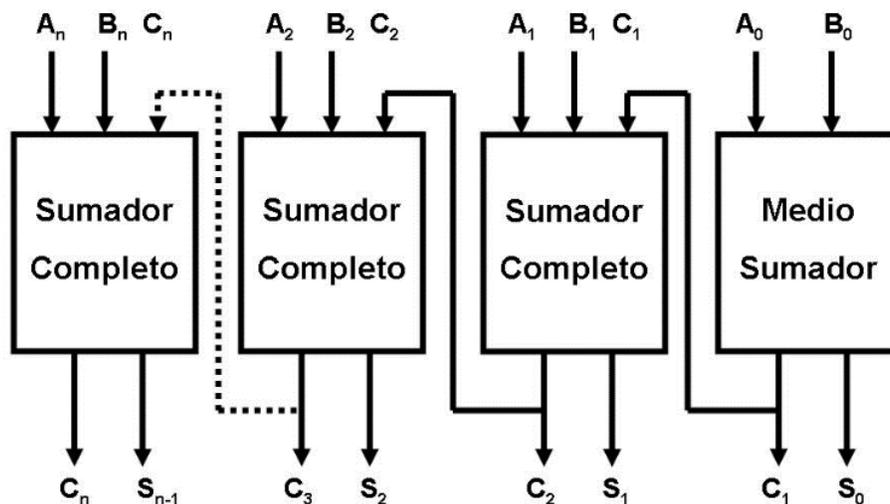


Figura 5.18 Sumador de n-bits implementados con n-1 sumadores completos.
Fuente: (Mano, 1986: 162)

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”.

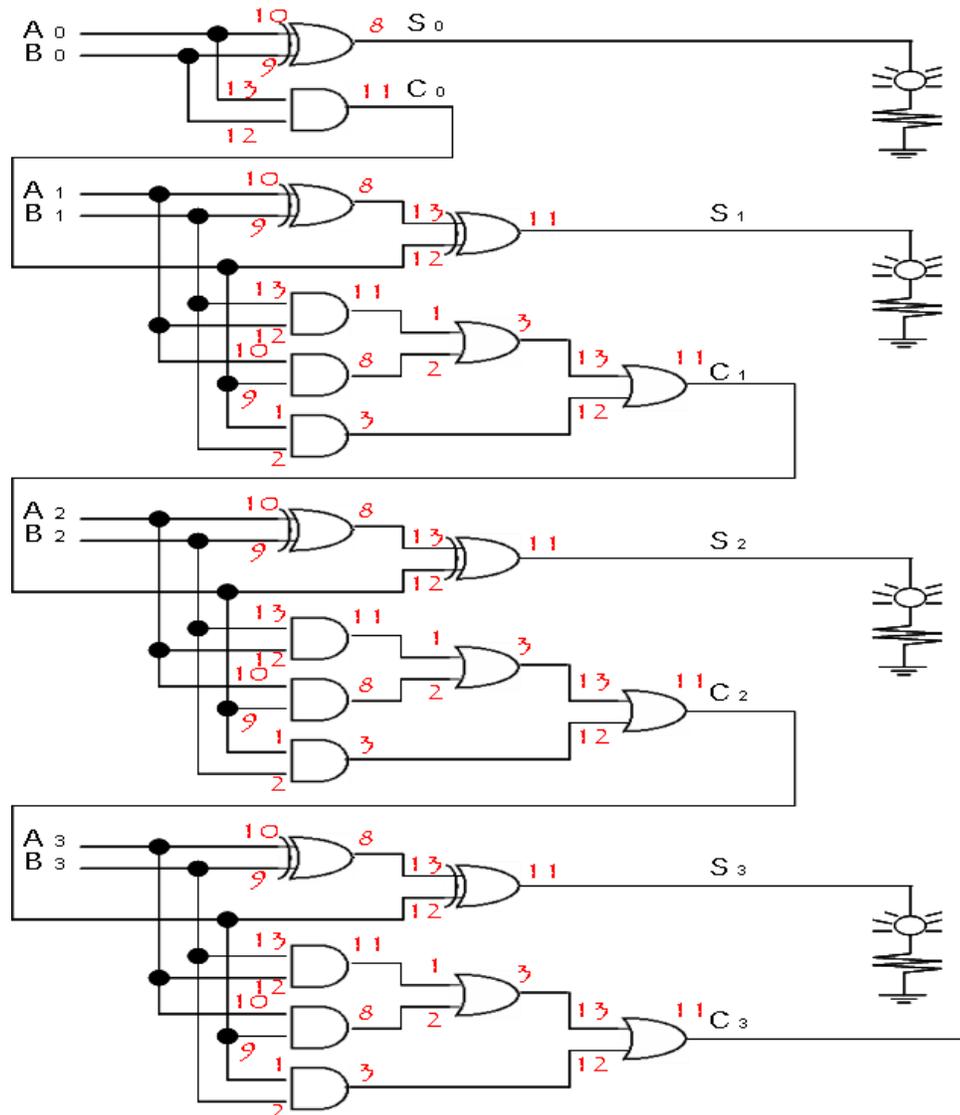


Figura 5.19 Diagrama eléctrico: Sumador de 4 bits
Fuente: (Mano, 1986: 162)

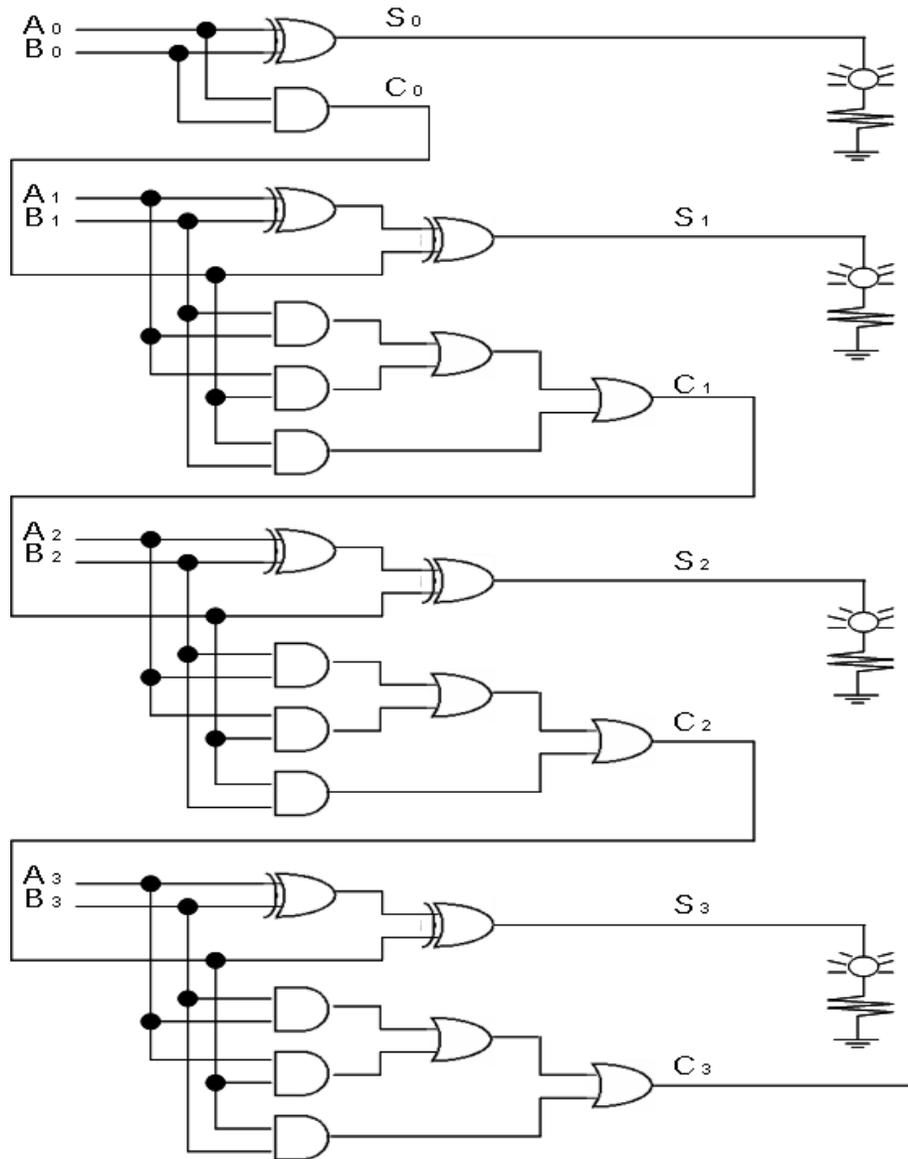


Figura 5.20 Diagrama eléctrico de un sumador de 4 bits en cascada.
Fuente: (Mano, 1986: 162)



5.7. Restadores

En el caso de los restadores, 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 (pedir prestado). Para el caso de un restador binario de un dígito 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 semirrestador 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	<i>Borrow</i>
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Funciones:

Mediante mapas K obtenemos las funciones asociadas:

	M	0	1
S	0	0	1
1	1	0	

	M	0	1
S	0	0	0
1	1	1	0

Figura 5.21 Elaboración propia.

La construcción a partir de compuertas AND y OR es:

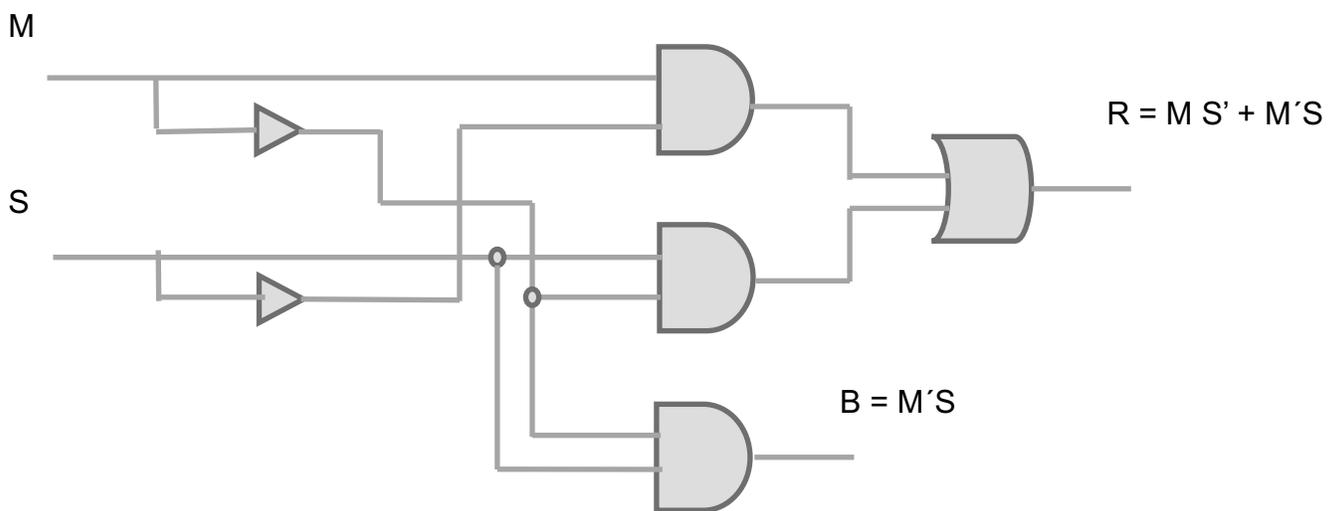


Figura 5.22 (Mano, 1986: 127)

Ejercicio:

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



5.8. Comparadores

Comparadores

Un tipo especial de circuitos lógicos combinación ales 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:

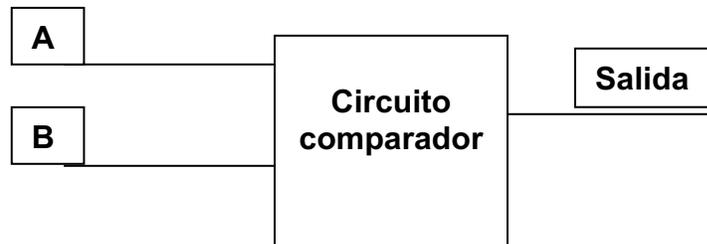


Figura 5.23.Circuito comparado.
Elaboración propia.

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:

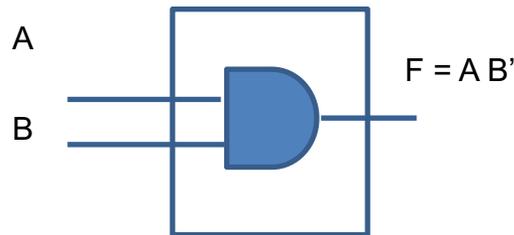


Figura 5.24 Circuito para F. Elaboración propia.

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:

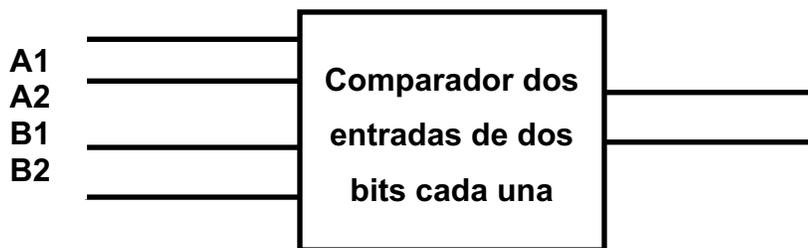


Figura 5.25 Comparador de dos entradas de dos bits. Elaboración propia.

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

Tabla.5.1 Función de salida para un comparador de dos entradas. Elaboración propia.

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

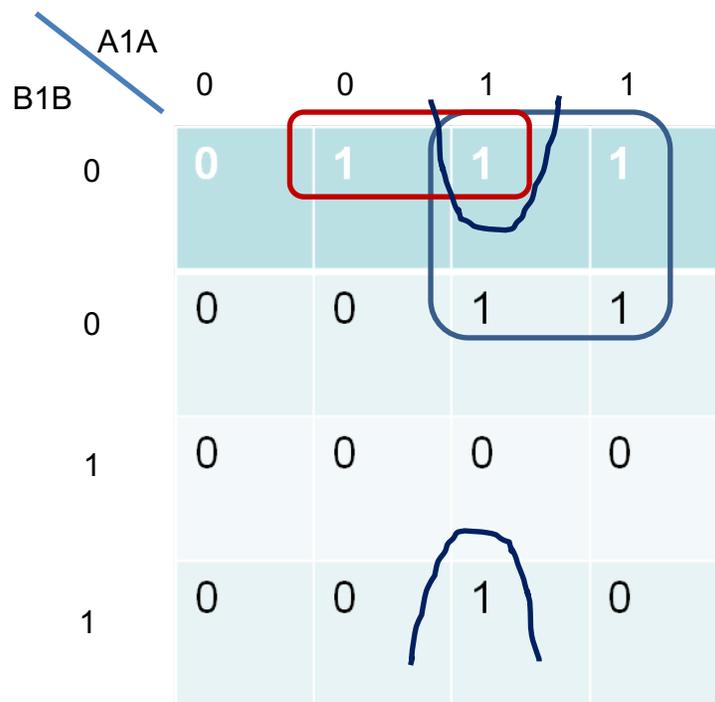


Figura 5.26 Mapa de Karnaugh. Elaboración propia. Fuente: (Mano, 1986: 179)



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 (en micrómetros o nanómetros), densidad y 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	<p>Pequeña escala de integración (SSI). Alrededor de 12 compuertas.</p> <p>Mediana escala de integración (MSI). Entre 12 y 100 compuertas.</p> <p>Gran escala de integración (LSI). De 100 a 1000 compuertas.</p> <p>Muy grande escala de integración. (VLSI). Más de 1000 compuertas.</p>

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 algunos tipos de memorias y arreglos lógicos programables y los primeros procesadores en los años 70. Los circuitos VLSI se utilizan actualmente y uno de sus usos es 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 desarrollará 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 DE LA UNIDAD



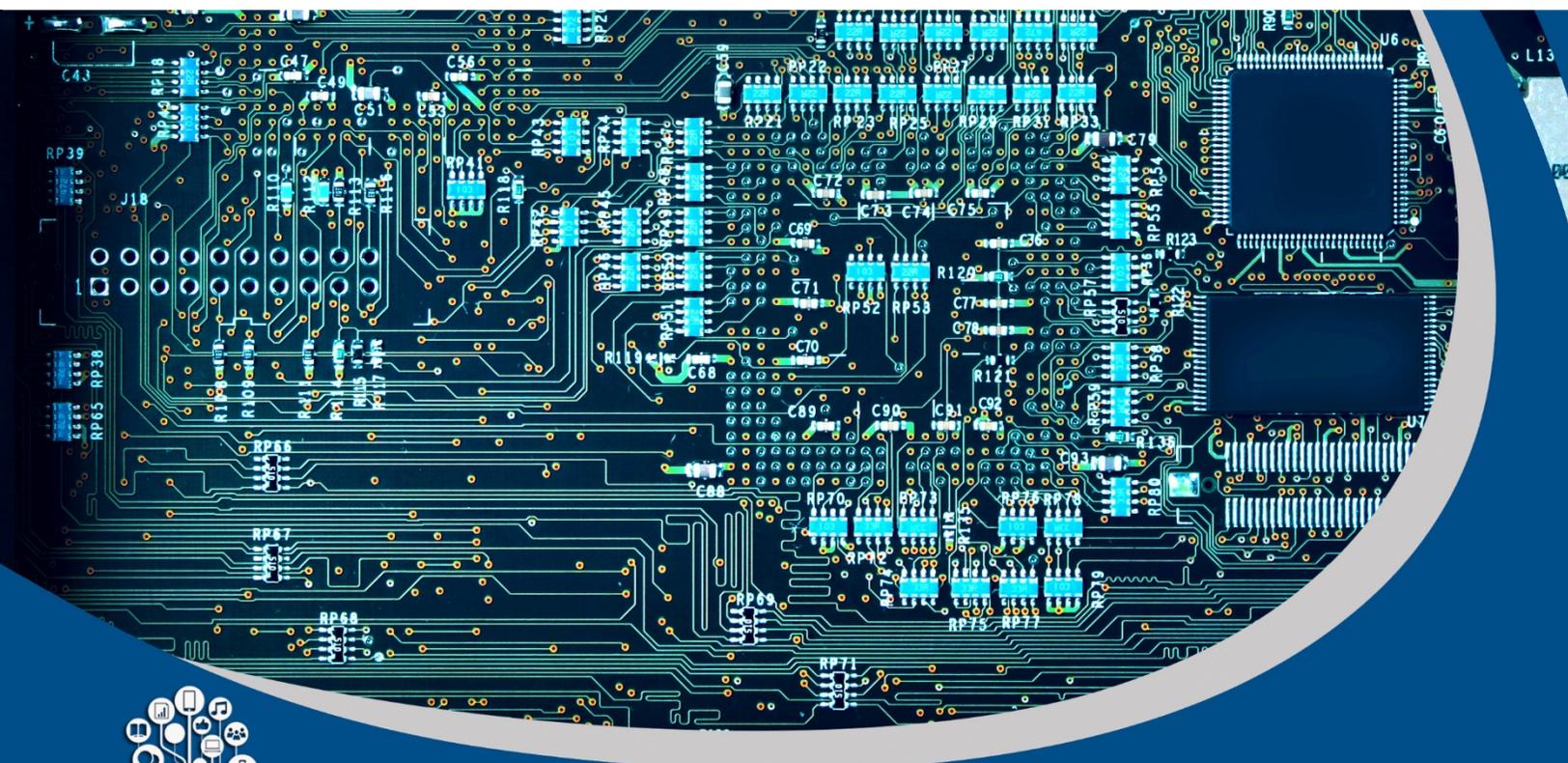
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 6

CIRCUITOS SECUENCIALES





OBJETIVO PARTICULAR

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

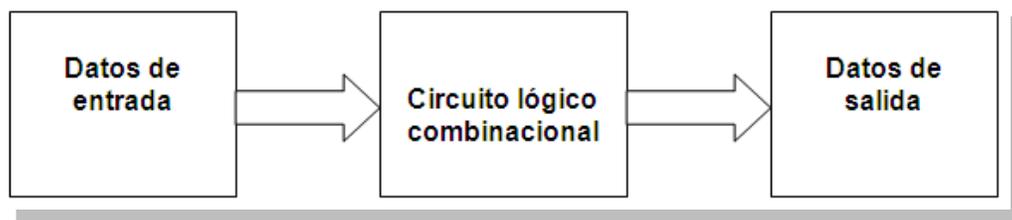
TEMARIO DETALLADO (10 horas)

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 atemporales. 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, sin embargo, 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 en 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 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:

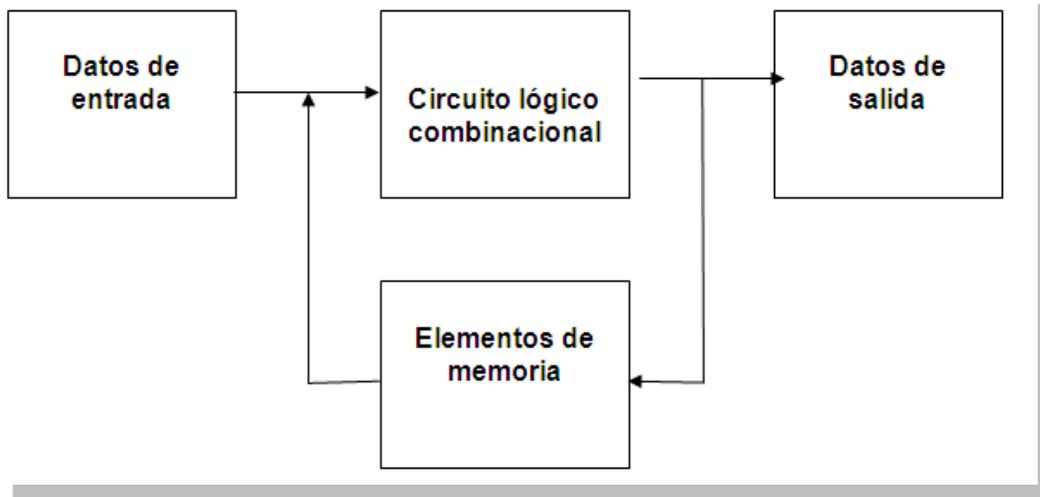


Figura 6.1 Modelo de proceso información
Elaboración propia con base en (Mano, 1986: 209)

6.1. Circuitos síncronos

El cambio de las variables internas de un circuito se puede producir de dos maneras en un sistema 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" o "1").

Por flanco o cambios de nivel

- La acción de las variables de entrada sobre el sistema se produce cuando ocurre un flanco (subida o bajada de un valor dado, en la transición) activo del reloj. Este flanco activo puede ser al cambiar de 0 a 1 o al cambio de 1 a 0. (Lopez, 2006: 2).

Figura 6.2 Cambio de variables internas de un circuito. Elaboración propia.



6.2. Circuitos asíncronos

Los circuitos secuenciales asíncronos son circuitos digitales donde los cambios de estado ocurren al cambiar las señales de entrada, atemporalmente, solamente afectados por los *retardos* asociados a las compuertas lógicas utilizadas en su implementación (aunque para efectos humanos, los retardos puedan ser intervalos de tiempo despreciables, para la velocidad a la que operan los circuitos esta diferencia de tiempo llega a ser relevante y se denomina *tiempo de retardo del circuito*, este puede ser del orden de nanosegundos), 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, ya que, como hemos visto, el proceso de la información se puede dar en secuencias temporales.

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.

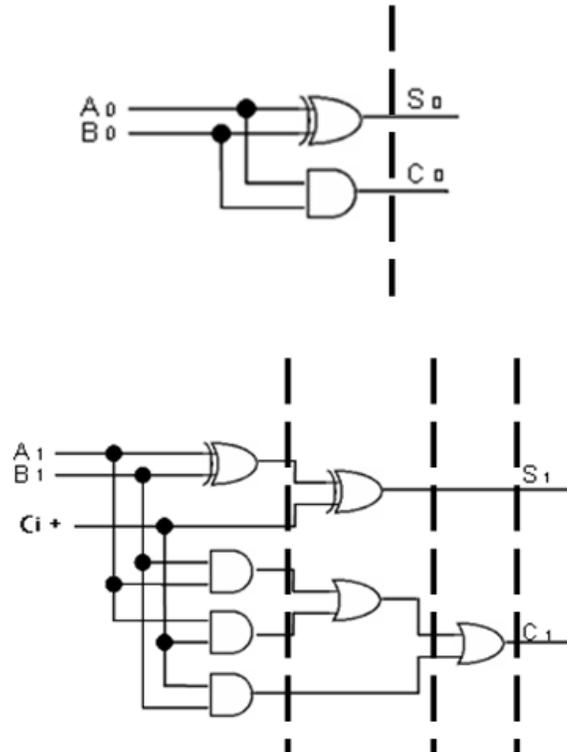


Figura 6.3 Circuitos de sumador medio y sumador completo. Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 669.

Para el medio sumador observamos un nivel de retraso (ver línea punteada) debido a que solo manejamos dos compuertas que están conectadas de forma paralela. Para los sumadores completos observamos tres niveles de diseño (ver secciones en línea punteada) lo que representa tres retrasos (la diferencia de tiempos en los que se presentan las entradas y se obtiene la salida). 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:

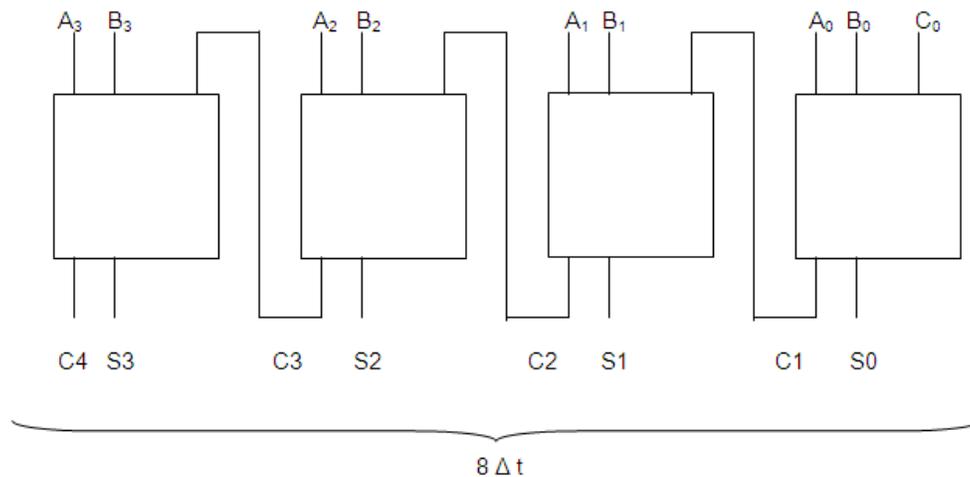


Figura 6.4 Diagrama de bloques. Fuente: Schilling Donald L. and Belove Charles. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill. p. 669.

Dado que cada sumador completo tiene un retraso de $2 \Delta t$, para los cuatro sumadores tenemos $8 \Delta t$. Si consideramos que el primer bloque puede ser un medio sumador, solo requerimos $7 \Delta t$. De manera general se requieren $2N-1 \Delta t$ 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 acarrees previamente y después realiza la suma de cada pareja de bits.

Para el Sumador completo:

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$$

$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$$



$$G_i \quad P_i$$

Donde G_i es la función generadora y P_i es la función propagadora.

Por lo que tenemos:

$$C_{i+1} = G_i + P_i C_i$$

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

$$i = 0$$

$$C_i = G_0 + P_0 C_0 = G_0 \quad \text{dado que } C_0 \text{ es cero} \rightarrow C_1 = G_0$$

Para $i = 1$



$$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0$$

Para $i=2$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0) = G_2 + P_2G_1 + P_2P_1G_0$$

Para $i=3$

$$C_4 = G_3 + P_3C_3 = G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0) = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

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

Valor de i	C	Valor del acarreo
0	C_1	G_0
1	C_2	$G_1 + P_1G_0$
2	C_3	$G_2 + P_2G_1 + P_2P_1G_0$
3	C_4	$G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$
4	C_5	$G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1G_0$

Tabla 6.1 Valores de i . Elaboración propia. Fuente: Donald L. Schilling and Charles Belove. (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 669.

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:

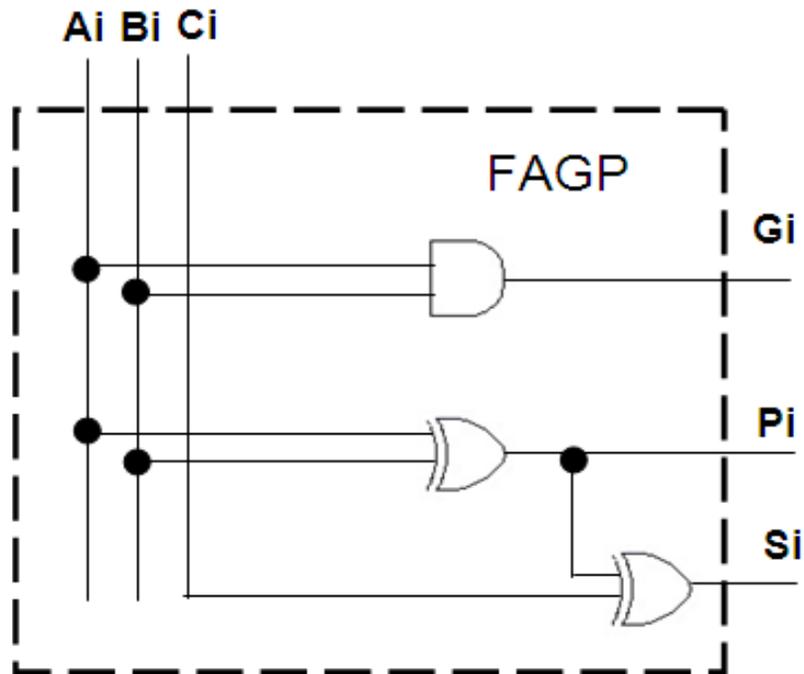


Figura 6.5 Circuito para el generador propagador. Fuente: Schilling Donald L. and Belove. Charles (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, p. 669.

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

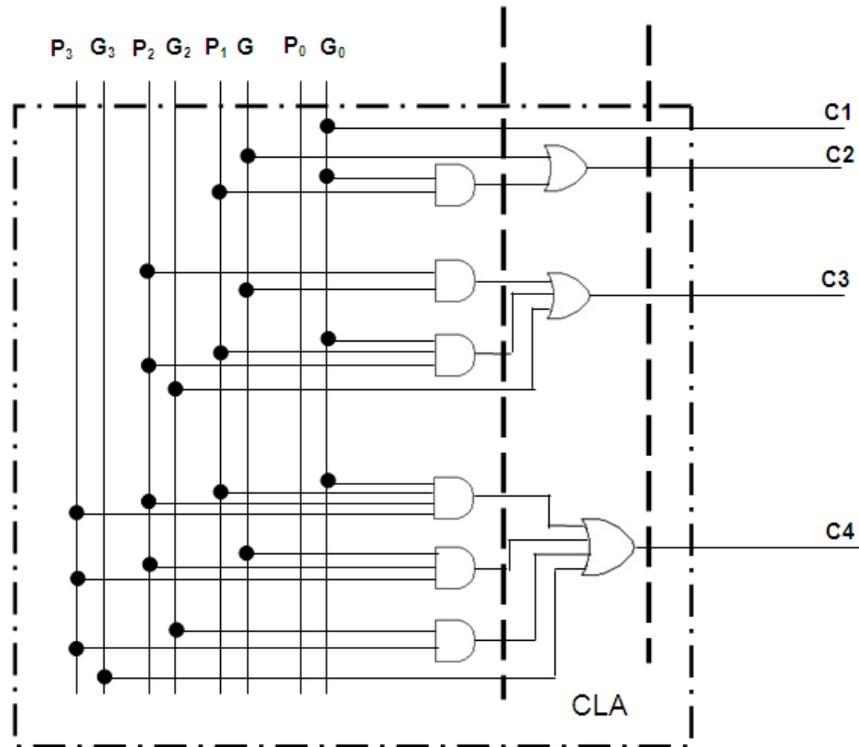


Figura 6.6 Nivel de compuerta 1 para el circuito para el Carry Look Ahead (CLA).

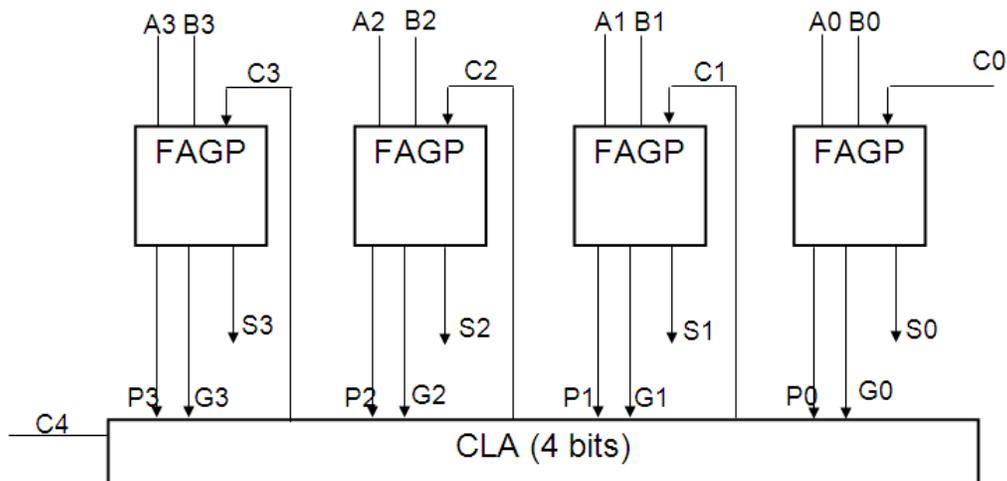


Figura 6.7 Nivel de compuerta 2 para el Carry Look Ahead (CLA).¹⁰

¹⁰ Circuito para el Carry Look Ahead (CLA). Fuente: Schilling Donald L. and Belove Charles (1985). *Electronic Circuits Discrete and integrated*. Singapore: McGraw Hill, página 673.



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



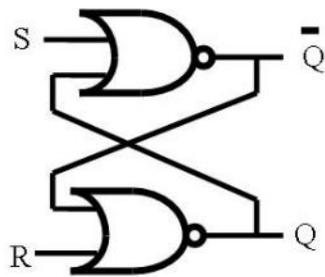
Y todos ellos tienen las siguientes propiedades:

- 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, \bar{Q} y Q , las cuales son siempre complementarias (es decir, si Q se encuentra en estado lógico “uno” la otra automáticamente tendrá un nivel lógico “0”).

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, \bar{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)

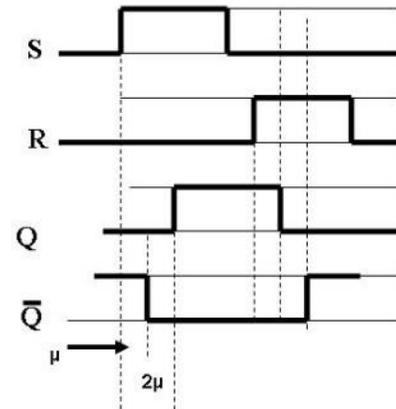


Figura 6.8 Un circuito Flip-Flop S-R a base de compuertas NOR.
Fuente: (Mano, 1986: 210)

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 $\bar{Q}=1$ 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 $Q=0$, $\bar{Q}=1$, $S=0$, $R=0$. Supongamos que S cambia al valor 1. Ahora las entradas a la compuerta NOR inferior son $S=1$, $\bar{Q}=0$.



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

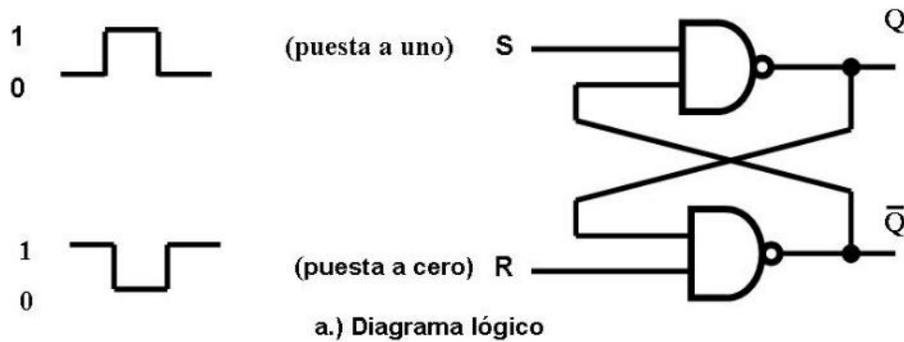
Así que, en este momento, las entradas a la compuerta NOR superior pasan a ser $R=0$, $\overline{Q}=0$. Después de otro retardo de puerta de (μ), la salida Q pasa a 1. Este de nuevo es un estado estable. Las entradas de la parte inferior son ahora $S=1$, $Q=1$, que mantienen la salida $Q=0$. Mientras $S=1$ y $R=0$, las salidas seguirán siendo $Q=1$, $\overline{Q}=0$. Además, si S vuelve a 0, las salidas permanecerán sin cambiar. En resumen, cuando la entrada S toma el valor de 1, a dicha acción se le conoce como “PRESET” (prefijado o predeterminado) 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 $Q=0$, $\overline{Q}=1$, sin importar el estado previo de Q y \overline{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{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*.



S	R	Q	\bar{Q}	
1	0	0	1	
1	1	0	1	(Después de S = 1, R = 0)
0	1	1	0	
1	1	1	0	(Después de S = 0, R = 1)
0	0	1	1	

b.) Tabla de Verdad

Figura 6.9 Flip-Flop SR a base de compuertas NAND.
Fuente: (Mano, 1986: 211)

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 \bar{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 pueden 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.

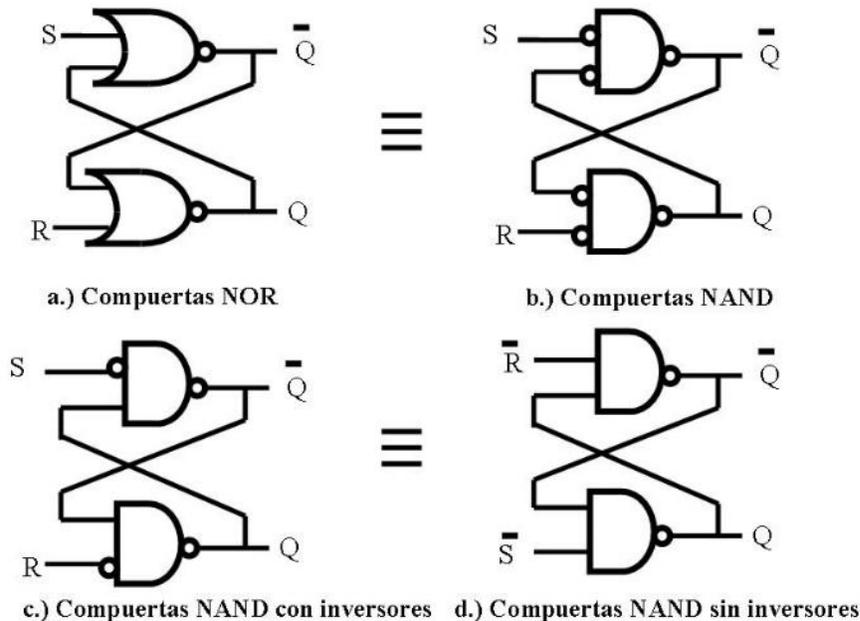


Figura 6.10 Implementación de Flip-Flop SR a partir de diversas compuertas básicas.
Fuente: (Mano, 1986: 212)

Existen otros tipos de flip-flops a los cuales se les conoce como *flip-flop temporizados* (RS, JK, T y D), 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 solo 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 solo 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.

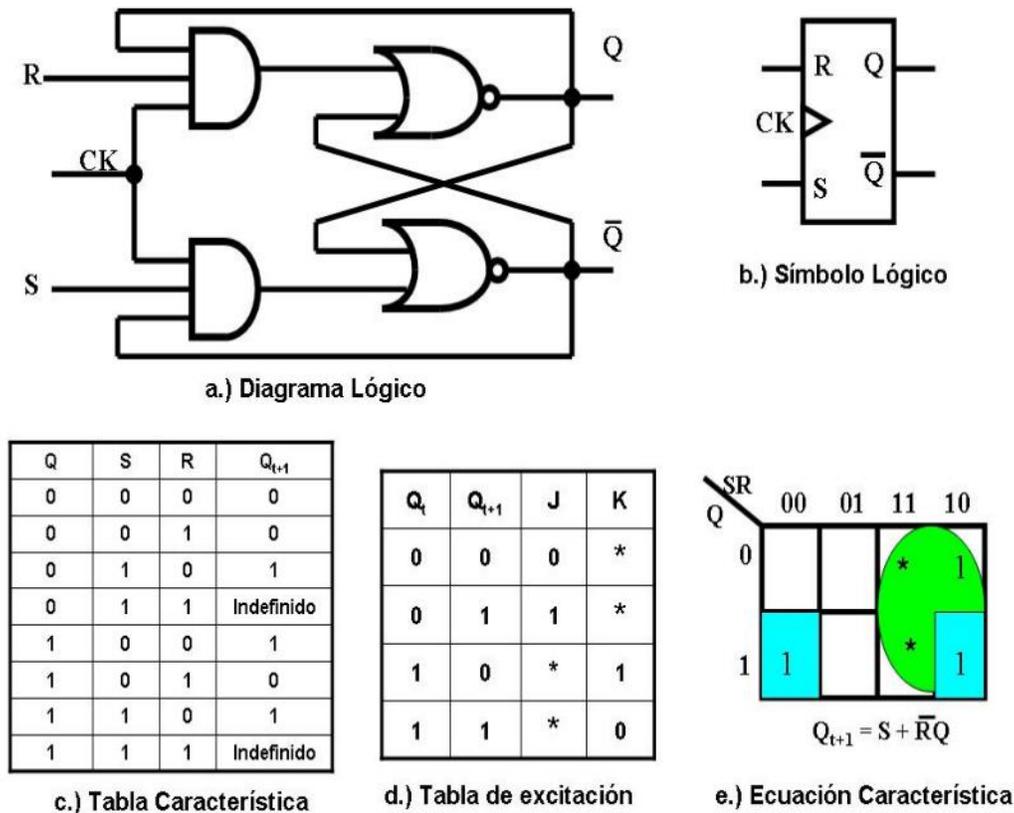


Figura 6.11 Flip Flop RS Temporizado
Fuente: (Mano, 1986: 2012).

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.

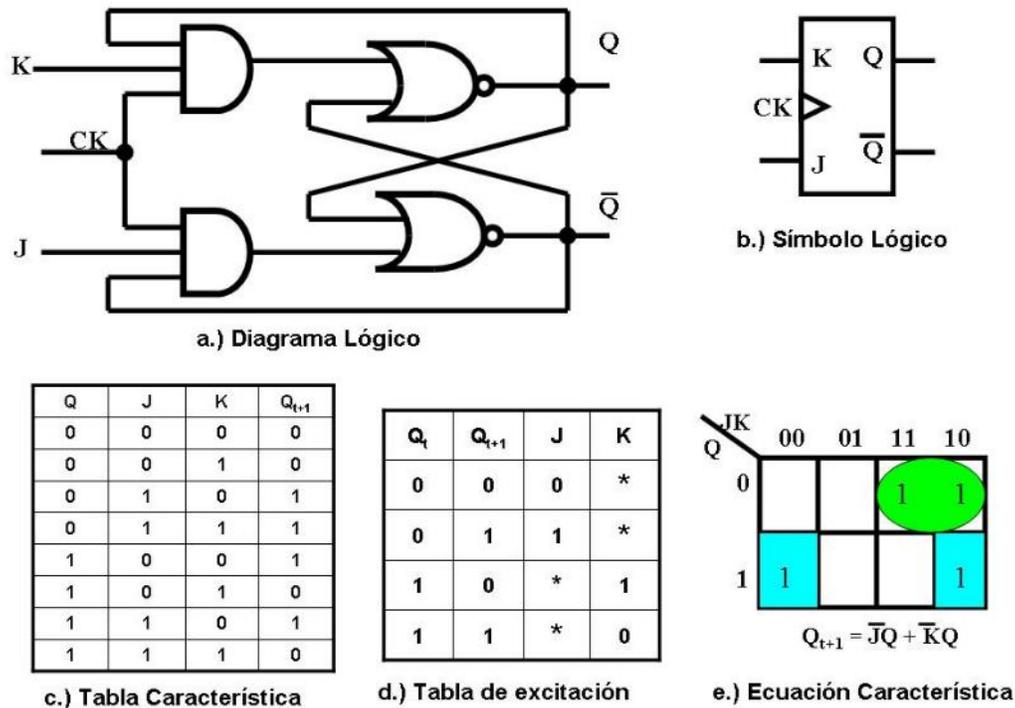


Figura 6.12 Flip-Flop JK Temporizado.
Fuente: (Mano, 1986: 215)

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. Debido a esta situación es 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 estos riesgos en los circuitos que controlan las entradas.

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

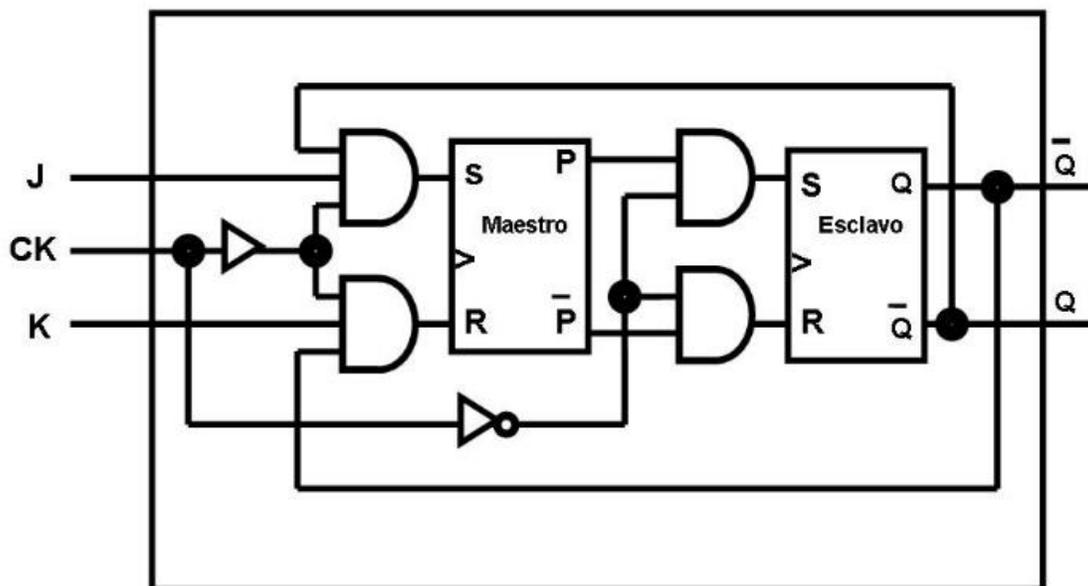
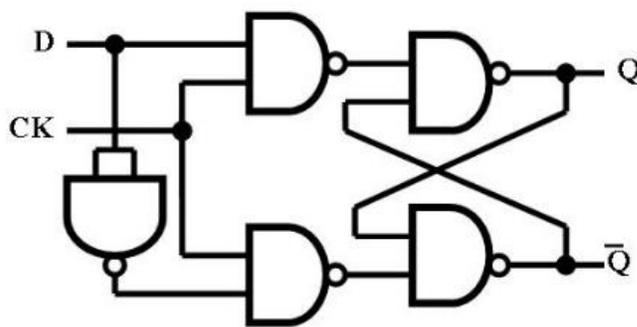


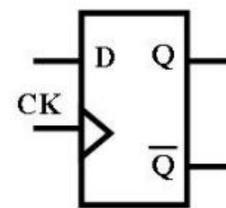
Figura 6.13 Flip-Flop Maestro-Esclavo JK
Fuente: (Mano, 1986: 215)

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.



a.) Diagrama Lógico



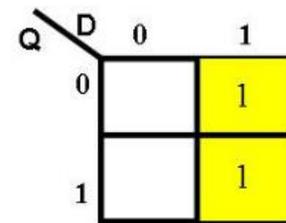
b.) Símbolo Lógico

Q	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

c.) Tabla Característica

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

d.) Tabla de excitación



e.) Ecuación Característica
 $Q_{t+1} = D$

Figura 6.14 Flip-Flop D Temporizado
Fuente: (Mano, 1986: 214)

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 E_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 “Flip-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.

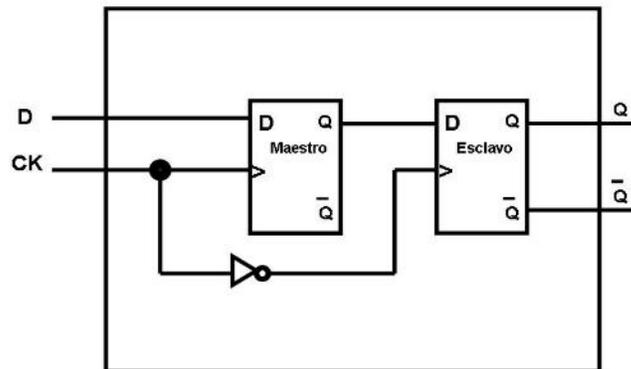


Figura 6.15 Flip-Flop Maestro-esclavo D
Elaboración propia con base en: (Mano, 1986: 218)

Flip-flop T

El flip-flop T (de “toggle”: conmutar en inglés) 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.

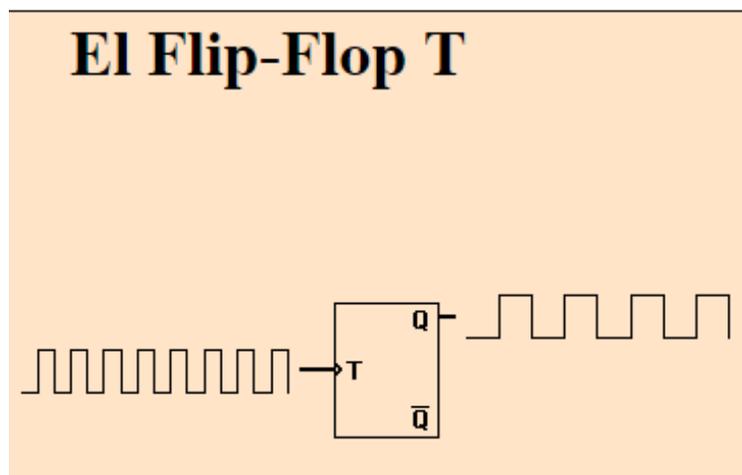


Figura 6.16. Flip-Flop T

Fuente: M Olmo R Nave (s/f), "El Flip-Flop T". Consultado el 5 de febrero de 2019 de: <http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/Tflipflop.html>

Diseño de un Circuito Secuencial

Haciendo nuevamente referencia al circuito secuencial de la figura "Modelo clásico de un circuito secuencial" y utilizando el modelo de máquina de Mealy, se pueden 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 el tema 6.5 Temporizadores.

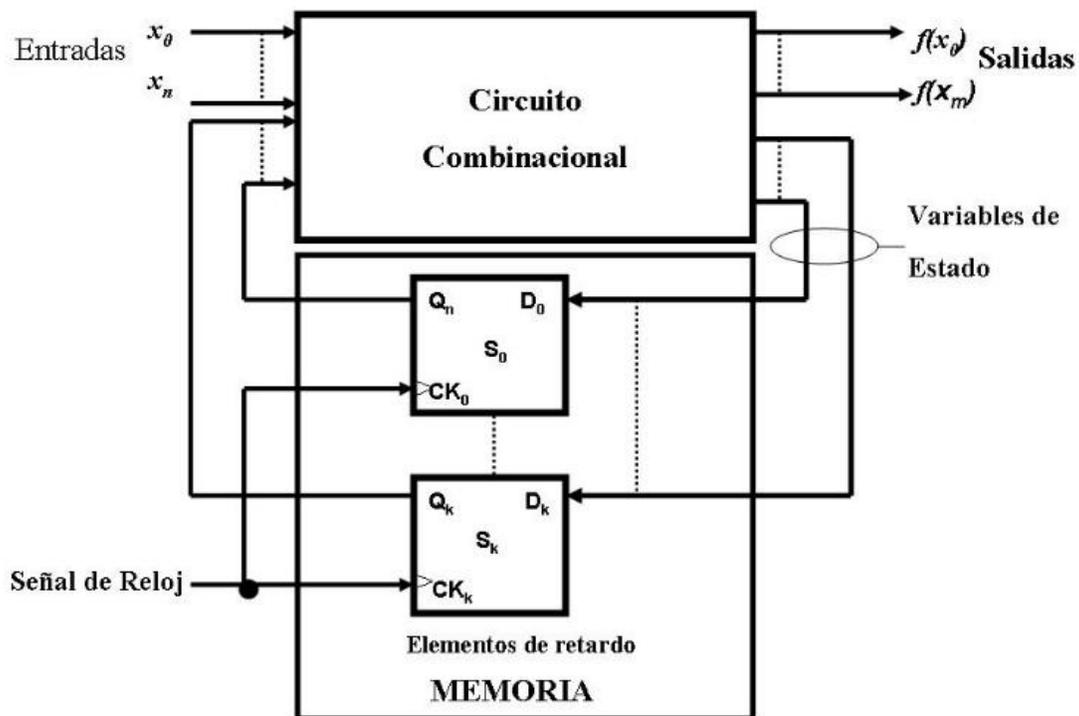


Figura 6.18 Modelo clásico de un circuito secuencial. Elaboración propia.
Fuente. (Mano, 1986: 245)

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



<p>Enunciado del problema</p>	<p>Se establece la descripción en palabras del comportamiento del circuito, esto puede acompañarse por:</p> <ul style="list-style-type: none"> • El diagrama de estado • Un diagrama de tiempos, u <p>Otra información pertinente (diagrama de flujo, <i>carta ASM</i>, etc.).</p>
<p>Obtención tabla de estado</p>	<p>De la información recabada del punto anterior, se obtiene la tabla de estado.</p>
<p>Reducción del número de estados en el circuito secuencial</p>	<p>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.</p>
<p>Asignación de valores binarios a cada estado</p>	<p>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.</p>
<p>Se obtiene el número de Flip-flops a utilizar</p>	<p>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:</p> $N = 2^n$ <p>es decir,</p>

	$n = \frac{\lg(N)}{\lg(2)}$ <p>donde: <i>n</i> Es el número de flip-flops necesarios <i>N</i> Número total de estados</p>
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 <i>n</i> 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.

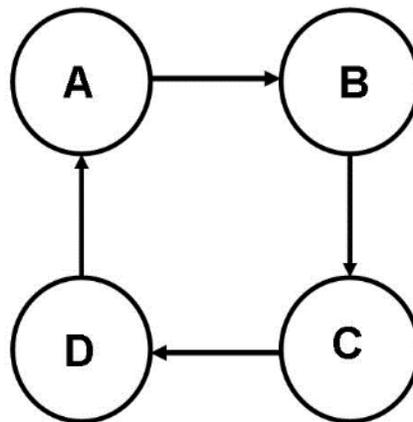
Tabla 6.1 Procedimiento para diseñar un circuito secuencial síncrono
 Fuente: (Mano, 1986: 141)

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.
Fuente: (Mano, 1986: 242)

2. Obtención tabla de estado

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

Tabla 6.2 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

Tabla 6.3 Asignación de estados

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 6.4 Estados. Fuente de las tablas anteriores:
Elaboración propia con base en: (Mano, 1986: 242)

5. Se determina el número de Flip-flops por utilizar

A partir de la ecuación:

$$N = 2^n$$

Donde

- N Número de estado
- n Número de flip-flop a utilizar

En nuestro caso $N = 4$ estados, hay que determinar el valor de n .

Despejando n obtenemos

$$n = \frac{\lg(N)}{\lg(2)} = \frac{\lg(4)}{\lg(2)} = \frac{0.602059991}{0.30102999566} = 2$$



Por lo tanto, se requieren de 2 flip-flops para representar los cuatro estados, los A, B, C y D.

6. Elección del flip-flop por utilizar

En este ejemplo se seleccionó (a partir del enunciado del problema) el flip-flop JK.

7. Obtención de la ecuación de excitación

A partir de las tablas de estado se deduce la excitación del circuito y la tabla de salida.

Estado Presente		Estado Futuro	
Q ₀	Q ₁	Q ₀	Q ₁
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Tabla 6.5 Estado
Fuente: (Mano, 1986: 244)

8. Obtención de las funciones de salida

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

Utilizando la tabla característica del flip-flop JK se obtiene las funciones de salida del circuito y las funciones de entrada de los 2 flip-flops de la siguiente manera:



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

Tabla de excitación

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

	Q_1	0	1
Q_0	0	0	1
	1	*	*

$J_0 = Q_1$

	Q_1	0	1
Q_0	0	*	*
	1	0	1

$K_0 = Q_1$

	Q_1	0	1
Q_0	0	1	*
	1	1	*

$J_1 = 1$

	Q_1	0	1
Q_0	0	*	1
	1	*	1

$K_1 = 1$

Figura 6.19 Fuente de las tablas anteriores:(Mano, 1986: 244)



9) Dibujar el diagrama lógico
Comprobar el circuito secuencial

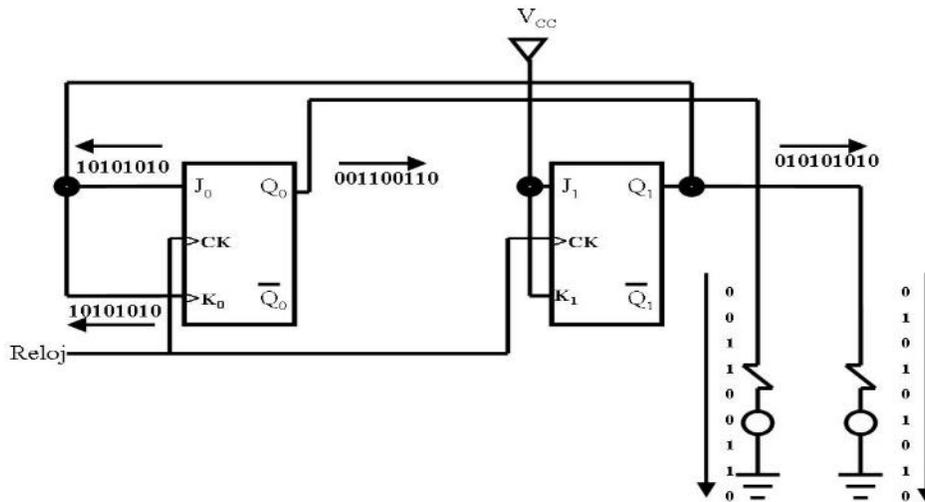


Figura 6.20 Diagrama lógico. Fuente: (Mano, 1986: 244)

10) Dibujar el diagrama eléctrico
Se alambra el diagrama eléctrico

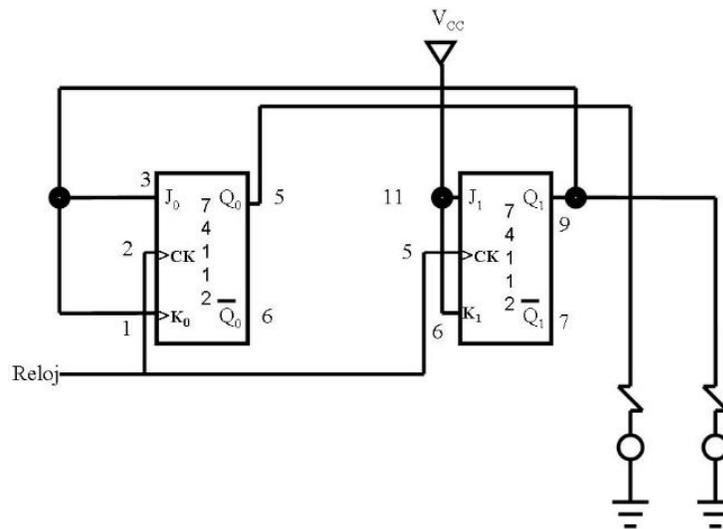


Figura 6.21 Diagrama eléctrico. Fuente: (Mano, 1986 p. 244)

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.

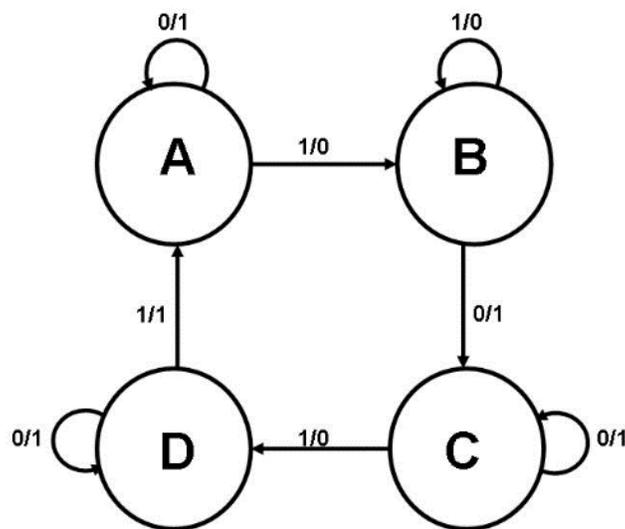


Figura 6.22. Ejemplo 2. Diagrama de Estados
Fuente: (Mano, 1986: 242)

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	
Q ₀	Q ₁	X = 0		X = 1		X = 0	X = 1
		Q ₀	Q ₁	Q ₀	Q 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

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 6.23 Excitación
Fuente: (Mano, 1986: 244)



Estado Presente		Estado Futuro				Salida	
Q ₀	Q ₁	X = 0		X = 1		X = 0	X = 1
Q ₀	Q ₁	Q ₀	Q ₁	Q ₀	Q 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

		J ₀				
		Q ₁ Q ₀	00	01	11	10
X						
0		*	1	*	*	
1		*	0	*	*	

$$J_0 = \bar{X}$$

		K ₀				
		Q ₀ Q ₁	00	01	11	01
X						
0		1	*	0	0	
1		1	*	1	0	

$$K_0 = \bar{Q}_0 + XQ_1$$

		J ₁				
		Q ₁ Q ₀	00	01	11	10
X						
0		0	*	*	0	
1		1	*	*	1	

$$J_1 = X$$

		K ₁				
		Q ₀ Q ₁	00	01	11	10
X						
0		*	1	0	*	
1		*	0	1	*	

$$K_1 = \bar{X} \bar{Q}_0 + XQ_1$$

Tabla 6.24 Estados. Fuente: (Mano, 1986: 244)



9. Dibujar el diagrama lógico

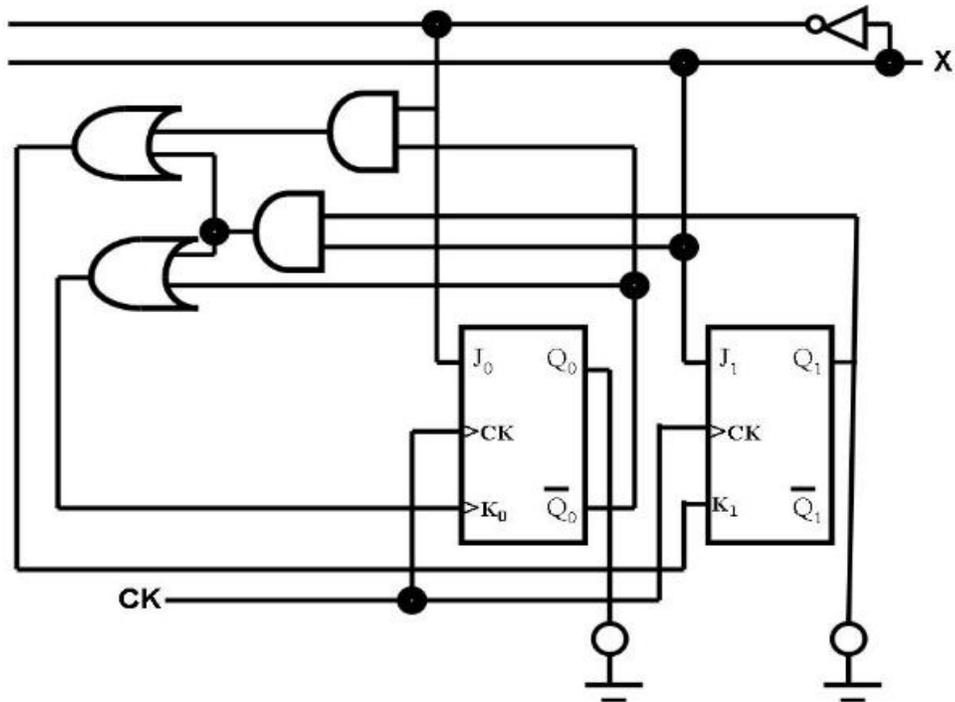


Figura 6.25 Diagrama lógico. Fuente: (Mano, 1986: 244)



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.

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, esperamos el *flanco de subida* (es decir, el instante 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-flop (Q_0) y los demás "0" s se recorren una posición hacia la derecha. Enseguida, introducimos el dato "1" en la entrada del primer flip-flop y durante el segundo pulso de reloj esperamos el siguiente *flanco de subida*, el flip-flop 0 muestra el dato "1" en su salida, y los demás datos ("0" s) se recorren a la derecha una posición. En el tercer pulso de reloj, se introduce el dato "0" en la entrada del flip-flop 0, se espera el *flanco de subida* y este dato se presenta a la salida del flip-flop 0, el dato "1" que se tenía anteriormente, se recorre una posición a la derecha y se presenta en la salida del flip-flop 1 (Q_1) y los demás datos se recorren una posición hacia la derecha. En el cuarto pulso de reloj, se introduce el dato "0" en la entrada del flip-flop 0, durante el *flanco de subida*, este dato se presenta a la salida del flip-flop 0, el "1" que se tenía a la salida de este flip-flop se recorre a la derecha y se presenta en la salida del flip-flop 1 y el "0" que se tenía en esta salida se recorre una posición hacia la derecha y los demás datos se recorren una posición a la derecha, y así sucesivamente hasta introducir todos los datos en el registro de corrimiento.

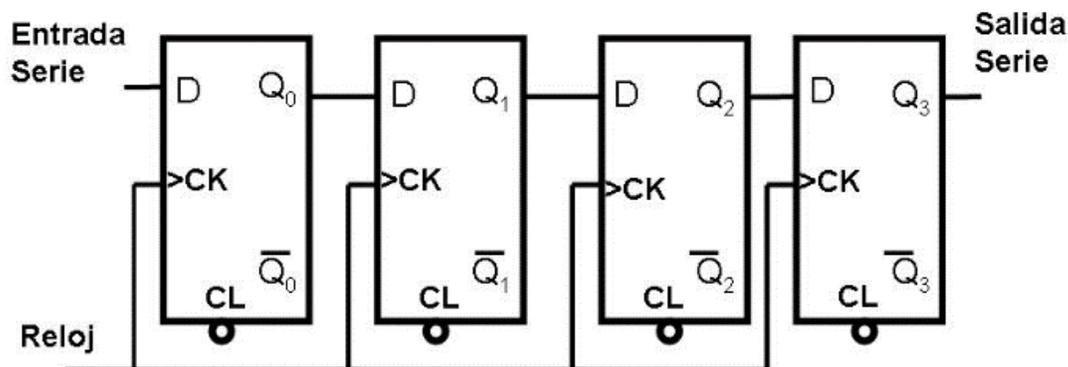


Figura 6.26 Registro de desplazamiento de 4 bits
Fuente: (Mano, 1986: 266)

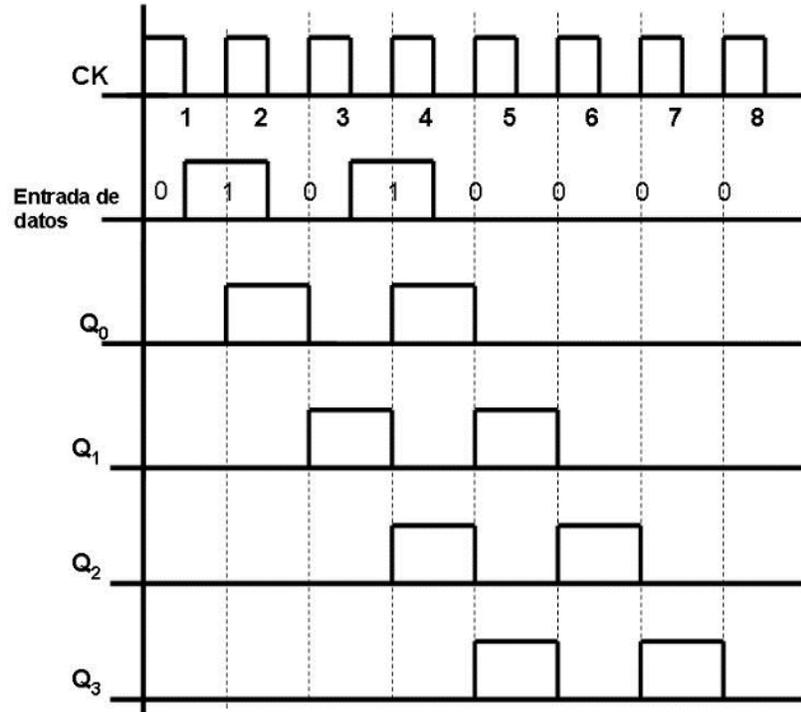


Figura 6.27 Registro de corrimiento (entrada serie-salida serie)
Elaboración propia, con base en: Mano, 1986: 268.

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 D₁₀ a la D₁₇.

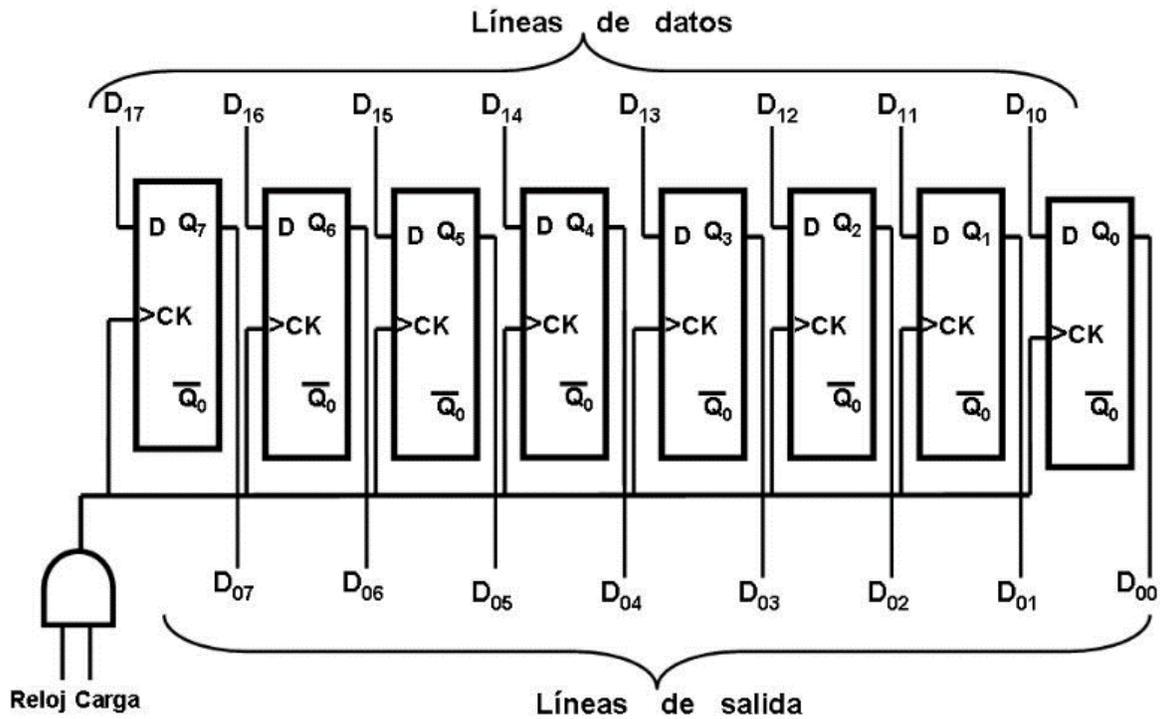


Figura 6.28 Registro en paralelo de 8 bits
Elaboración propia con base en: (Mano, 1986: 268)

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:

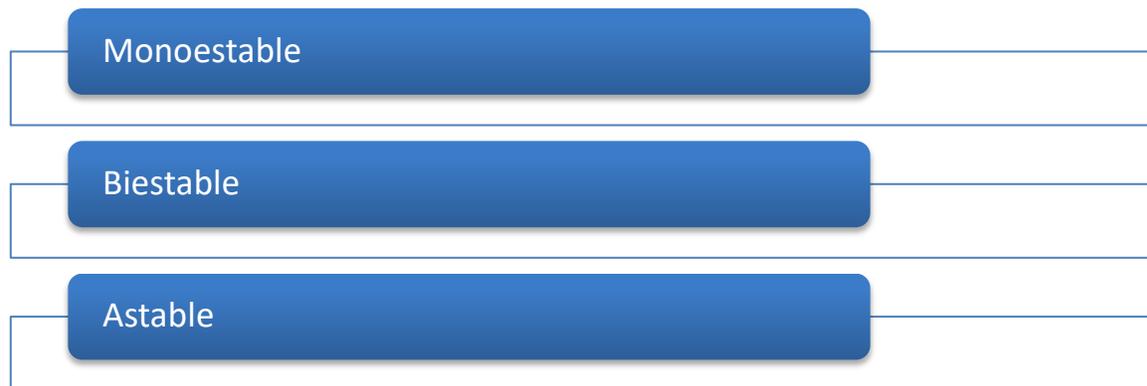


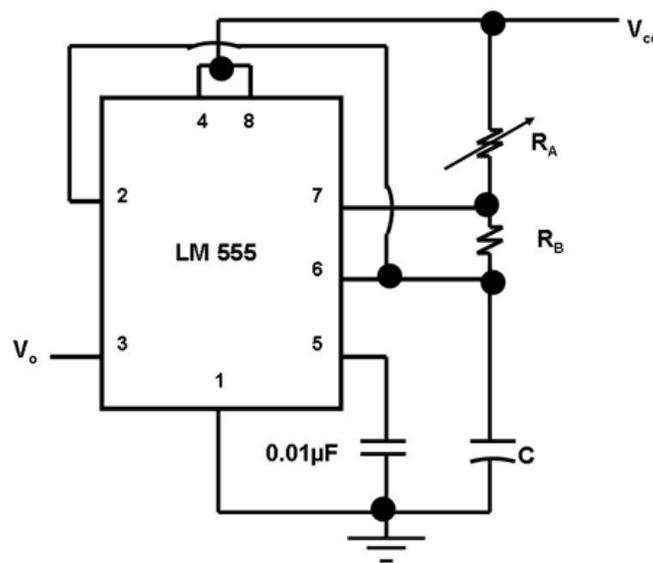
Figura 6.28 Modos de temporizador. Elaboración propia.

Los circuitos multivibradores (o de *conmutación regenerativa*) de tipo monoestable 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 cuadrada, 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)

Figura 6.29 C.I. LM555. Fuente: Robert F. Coughlin y Frederick Driscoll. (1987). *Circuitos integrados lineales y amplificadores operacionales*. México: Prentice Hall, p. 300.

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 $F1=1$ = $T1=1$ = 1
 tiene Hz /f1 seg
 que

$F2= 20$ = $T2=1$ = =0.0
 Hz /f2 5
 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 $C=200[\mu F]$ (Capacitor electrolítico de valor comercial)

Despejando R_B de la ec. (3)

$$R_B = \frac{0.05 \cdot 327.90}{(0.693)(220 \times 10^{-6})} \text{ [ohm]}$$

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 .

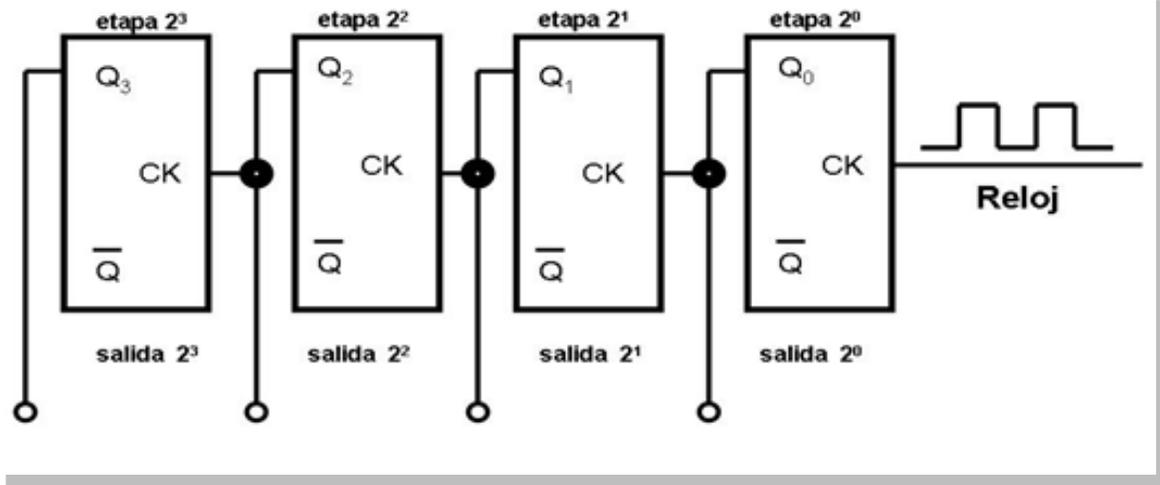


Figura 6.30 Diagrama a bloques de un contador binario. Fuente: Elaboración propia con base en: (Mano, 1986: 282)

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 o 0	0	0	0	0

Tabla.6.6 Contador binario de cuatro etapas
Fuente: (Mano, 1986: 283)

A partir de la tabla Contador binario de cuatro etapas se pueden 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 $256000/16$.

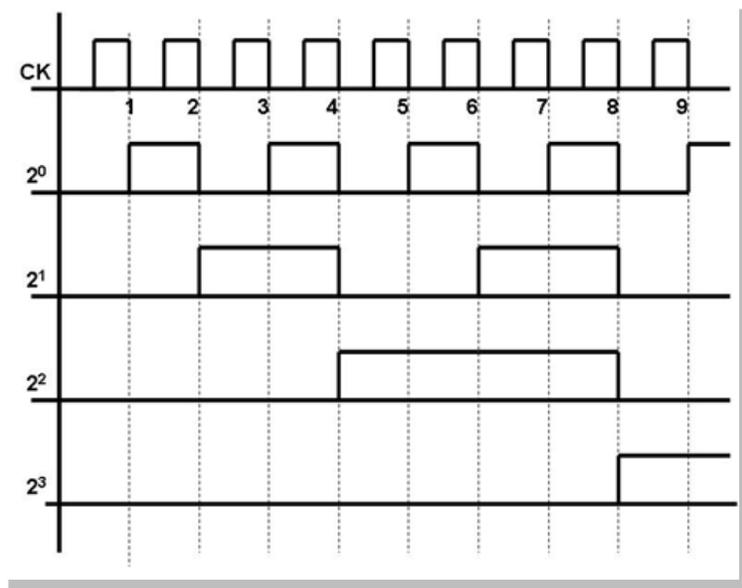


Figura 6.31 Formas de onda de un Divisor de frecuencias. Fuente: (Mano, 1986: 285)

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*).

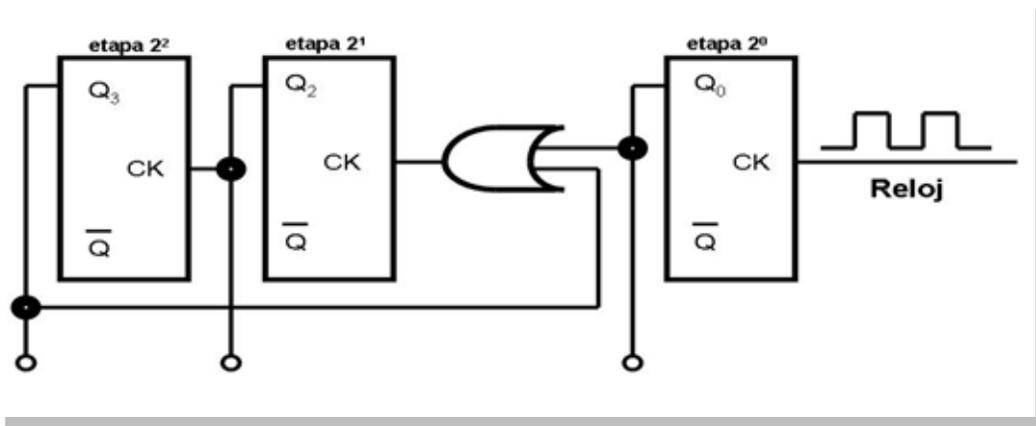


Figura 6.32 Diagrama a bloques de un contador binario mod 6. Elaboración propia con base en el extracto 7-18: (Mano, 1986: 289).

En la figura Diagrama a bloques se muestra un circuito para el conteo. Un contador para el conteo de 6 se suele llamar “modulo 6” (generalmente se abrevia mod. 6), indicando el modulo 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.

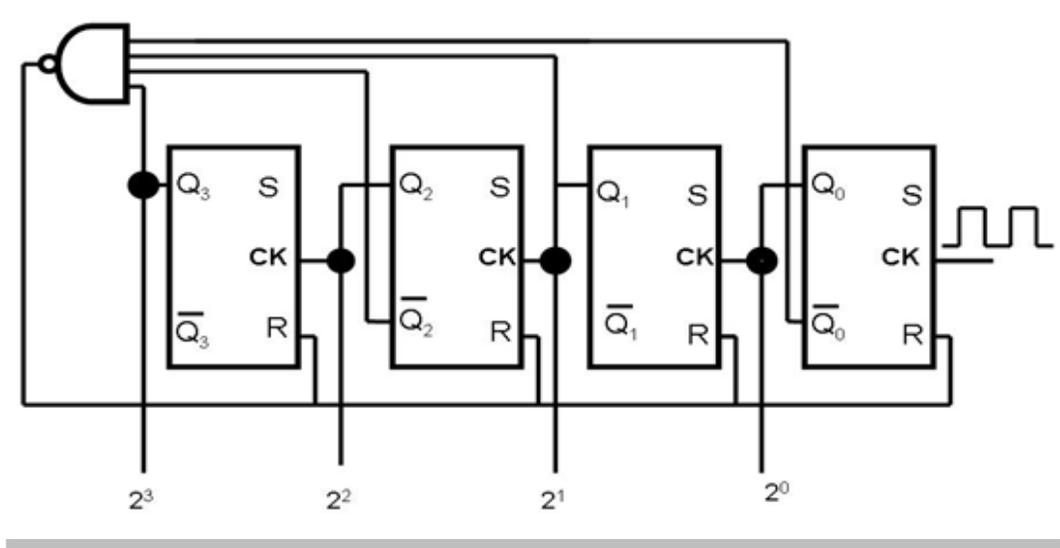


Figura 6.34 Contador binario de décadas. Fuente: (Mano, 1986: 249)

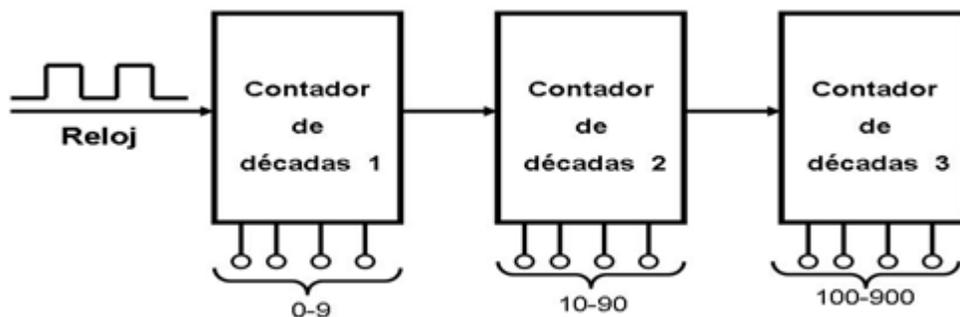
La compuerta produce un nivel bajo “0” cuando se alcanza el conteo 1010 (decimal 10) y repone (puesto a “0”) las cuatro etapas. La tabla muestra que solamente es necesario poner los flip-flops 2^1 y 2^3 para volver a contar desde cero. Se reponen todas las etapas para asegurar que el contador empieza 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

Figura 6.35 Contador binario decimal

Fuente: Elaboración propia con base en extracto 7-4. (Mano, 1986: 285).

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

Fuente: Elaboración propia con base en extracto 7-4. (Mano, 1986: 285).



RESUMEN

Los circuitos secuenciales incorporan un conjunto de dispositivos electrónicos capaces de almacenar datos de manera indefinida (para las memorias de tipo RAM, *memoria de acceso aleatorio*, mientras se mantenga la energía), estos circuitos son las memorias y funcionan como elementos de realimentación para un circuito combinatorial o procesador de datos. Al hablar de elementos de memoria, se incorpora también una variable que no se consideraba en los circuitos combinatoriales, *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 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 si está definido el uso para J=1 y K=1, en este caso la salida Q(t+1) es 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 DE LA UNIDAD



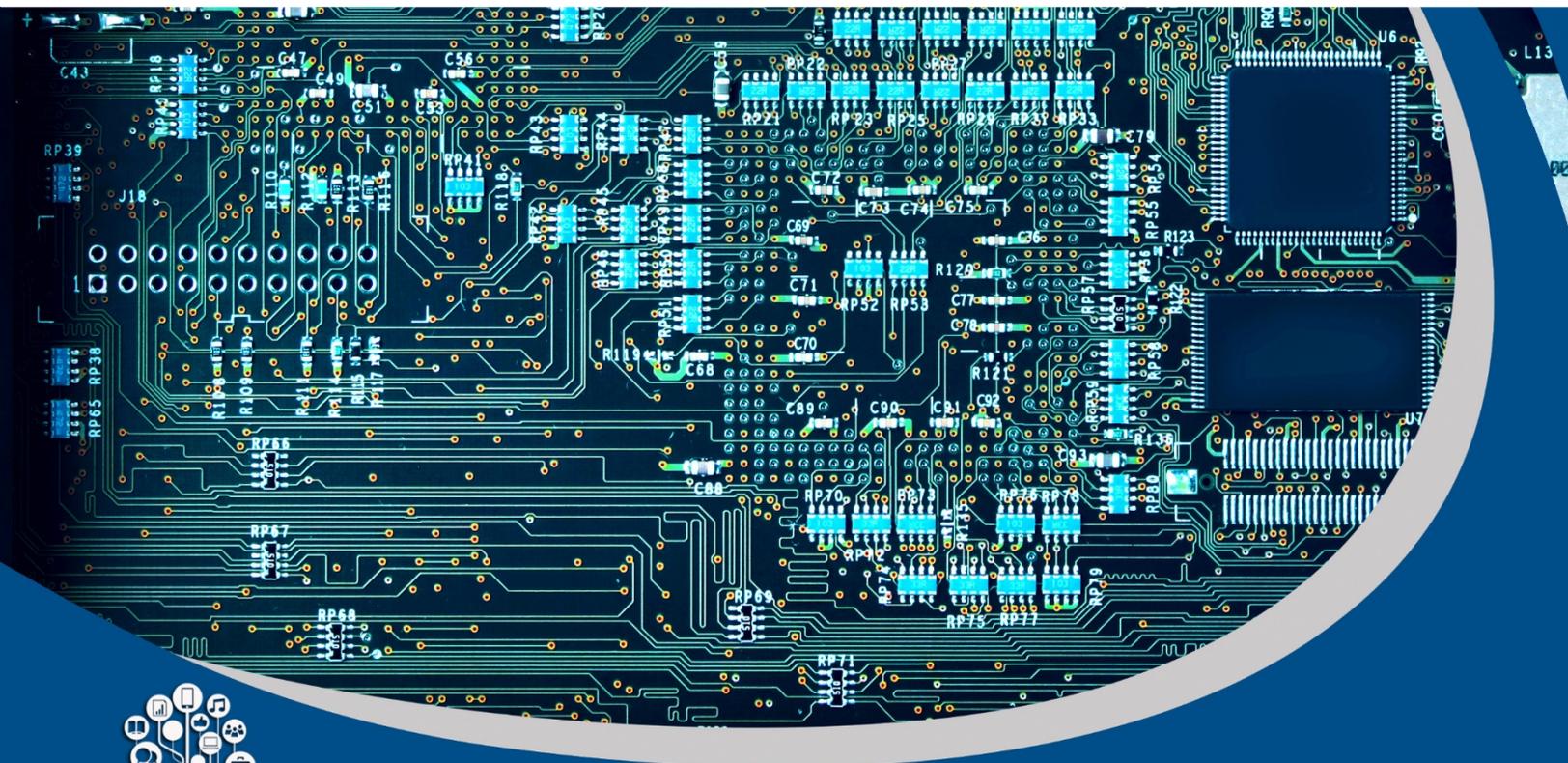
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 7

MEMORIAS





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 duros, mientras que los electrónicos conforman las memorias de



tipo RAM (memorias de lectura y escritura y son memorias de tipo no permanente, ya que al desconectar la energía del equipo su contenido se pierde) y ROM (memorias de solo lectura, son de tipo no volátil, su contenido no se pierde al desconectar la energía eléctrica del equipo). La diferencia

entre las unidades de almacenamiento (que se verán en detalle en la unidad 8) como discos duros y medios ópticos con las memorias es que los primeros guardan la información, aunque la computadora se apague, que son más baratas por byte almacenado 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 estudiamos en el capítulo anterior se utilizan para conformar circuitos VLSI para el almacenamiento de información y estos circuitos se denominan circuitos de memoria. Los registros de una computadora digital en el procesador son de dos tipos: operacionales y de almacenamiento. Los primeros incluyen circuitos lógicos combinación *ales* que, además de almacenar la información, permiten realizar alguna modificación a los mismos, mientras que los de almacenamiento solamente guardan 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, ROM, caché, así como sus características y la estructura de uso dentro de la computadora, llamada mapa de memoria.

7.1. Tipos de memoria

Los tipos de memoria que tiene una computadora digital básicamente son dos: 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*, o *memorias de acceso no secuencial*).



Las memorias RAM se utilizan para almacenar datos y programas. Los datos provienen de procesos y algoritmos y pueden ser resultados parciales o finales. Los programas se almacenan en discos duros y medios ópticos y para ejecutarlos o procesarlos, tienen que pasar primero a la memoria RAM para 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 de transistor 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 bit). Actualmente, las memorias RAM se fabrican con diversas tecnologías con lo cual se fabrican tarjetas de memoria con capacidades de hasta un máximo de 128 Gbytes con un tiempo de acceso de "fracciones de nanosegundos" (300 ps). 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 transistorizadas fueron construidas con tecnologías TTL y NNOS y a la vez están siendo reemplazadas por memorias RAM con tecnología *DDR* que ofrecen mejor funcionamiento, mayor densidad de transistores por mm² 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 o las memorias RAM MNOS que son "no volátiles", es decir, conservan los datos aun cuando se apague la fuente de alimentación. La mayoría de las memorias RAM son "volátiles", es decir, pierden su información al apagar el equipo.

Tipos de memorias RAM

Las memorias RAM se dividen por su diseño en tres tipos: Estáticas, dinámicas y pseudo-estáticas.



Memorias estáticas	<p>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".</p>
Memorias dinámicas	<p>Utilizan una estructura capacitiva. El término "dinámicas" se refiere a que cambian de estado con el paso del tiempo, debido a esto 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 (recargar o refrescar- "refresh"-) el contenido de cada localidad de memoria. Su ventaja de construcción es que están compuestas por un solo transistor y un capacitor integrado, lo que permite tener altísimos niveles de integración, asociado a un costo reducido en contraparte de las memorias estáticas, que como ventaja ofrecen una mayor rapidez de operación.</p> <p>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 circuito 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.</p>



	<p>La lógica más utilizada para actualizar memorias es un circuito externo que genera números de filas (a través de un contador como los que hemos visto) de manera secuencial por el Bus de Dirección y activa la señal MEMR (lectura de memoria) cuando la CPU no está accediendo a las memorias. Esto para no interrumpir el funcionamiento normal de la CPU. Su funcionamiento puede verse como muy complicado, sin embargo dada la velocidad de los circuitos integrados y su nivel de integración esta operación se facilita.</p>
Memorias pseudo-estáticas	<p>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.</p> <p>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 hasta 128 Gbytes y estáticas de 16 Mbytes. Las memorias dinámicas son más baratas que las estáticas.</p>

Tabla 7.1 Tipos de memoria RAM. Elaboración propia.

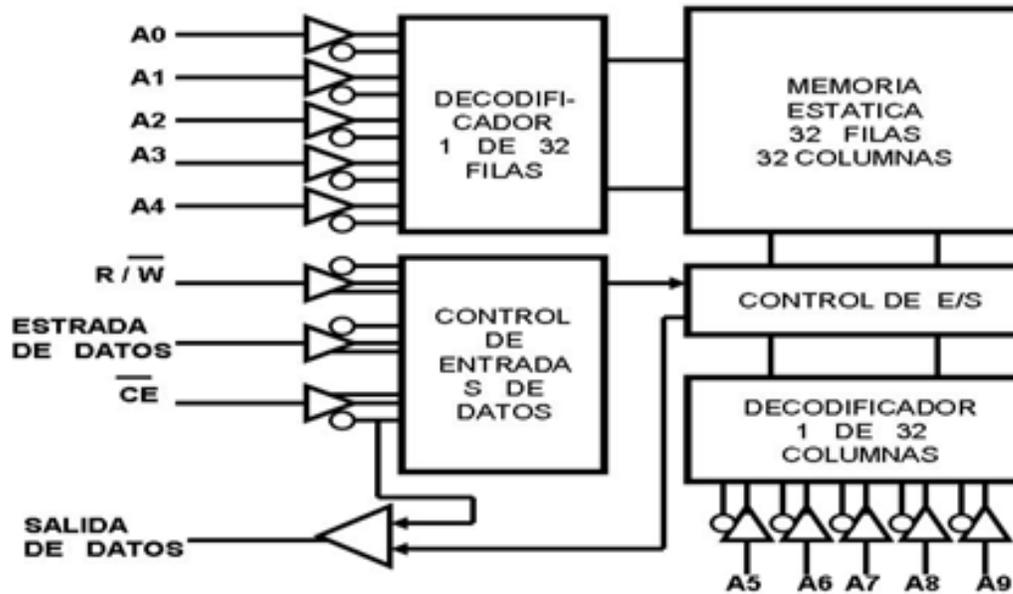
Organización de las memorias RAM

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

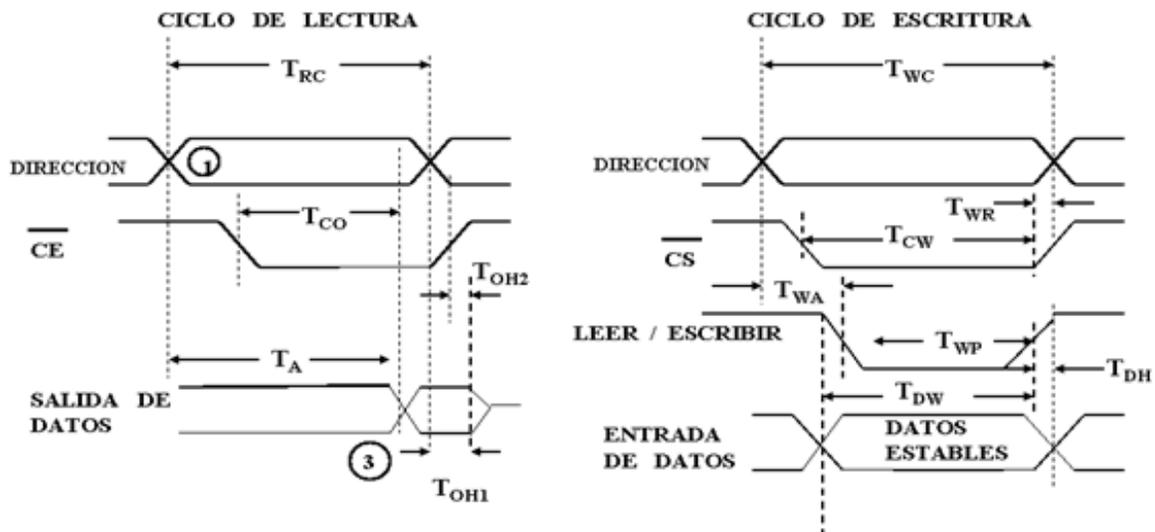
Una memoria RAM consta de dos bloques funcionales: Arreglo de localidades de memoria y los circuitos internos de interfaz, ver figuras 1a. y 2b. El arreglo de localidades de la memoria es generalmente una matriz cuadrada de localidades de uno o más bits arregladas en filas y columnas, tal es la razón 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.



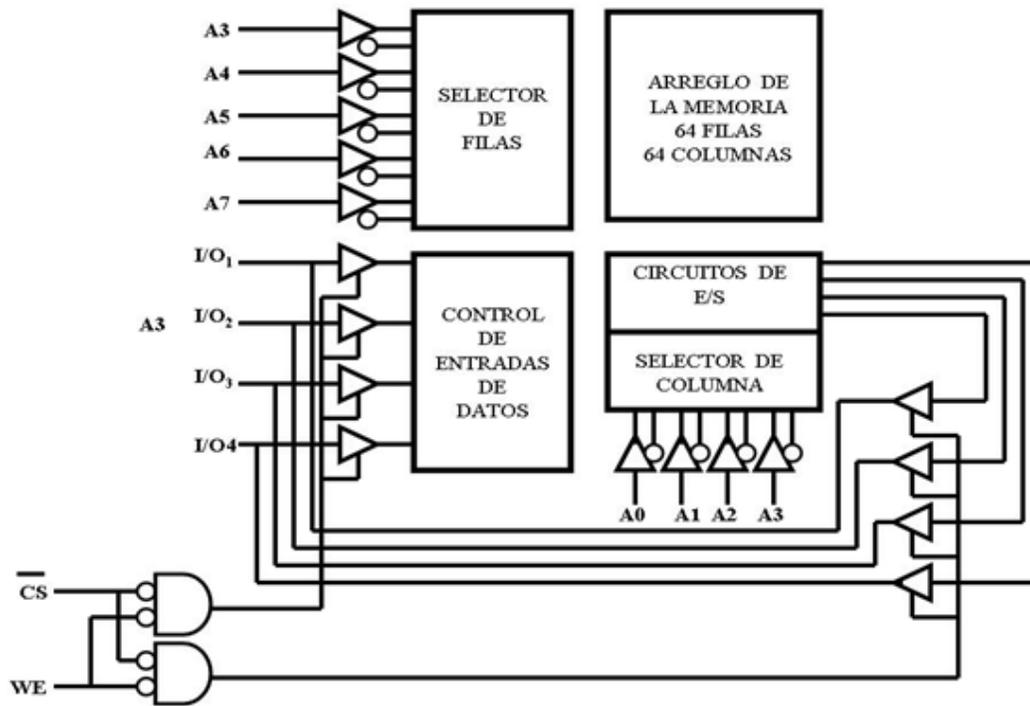
Los circuitos internos, una vez seleccionado el integrado y la localidad de la memoria, determinan si se va a recibir un dato (escribir) o si se va a enviar (leer).



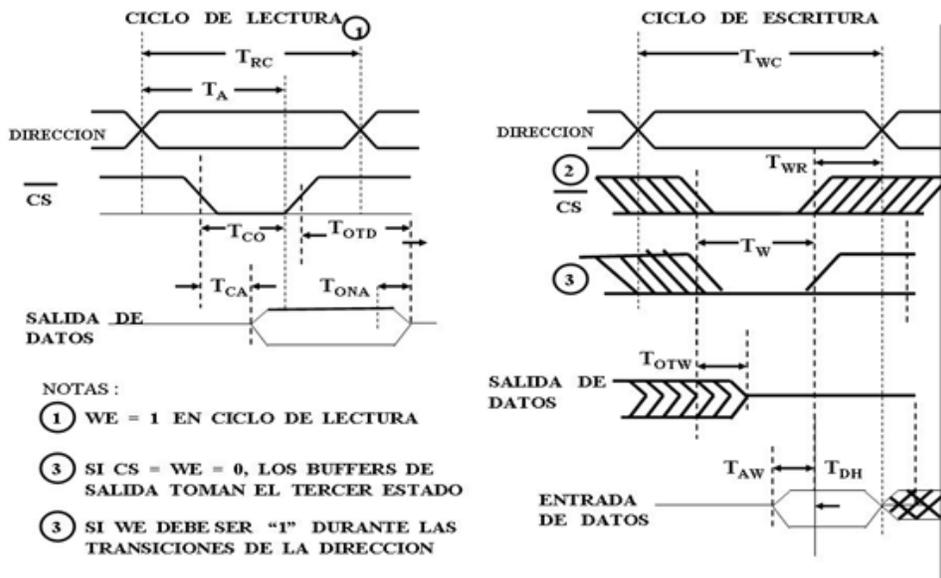
1a) Diagrama interno de la Memoria RAM 2102



1b Diagramas de tiempo de la memoria RAM 2102



2a) Memoria RAM 2114 - Diagrama a Bloques -



2b.) Diagramas de tiempo de la memoria RAM 2114

Figura 7.1 Fuente: Intel Corporation. (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.



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, T_a , 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, T_{co} , 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).

T_{oh2} (en el C.I. 2102) y T_{oha} (en el C.I. 2114) es el tiempo durante el cual el dato de salida sigue siendo 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. De nuestro microprocesador ejemplo del capítulo I, el 8085A (recordemos que se estudia este dispositivo por el nivel de detalle al que lo podemos revisar) 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

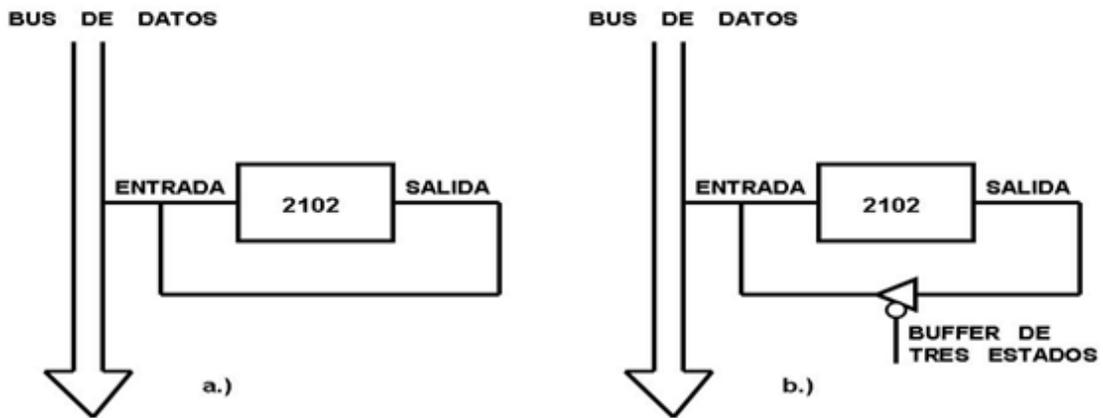
Existen cuatro configuraciones de Entrada/Salida de datos en las memorias RAM estáticas:

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* (reforzadores de señal) de tres estados (alto ("1"), bajo ("0") y desconectado o aislado) 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

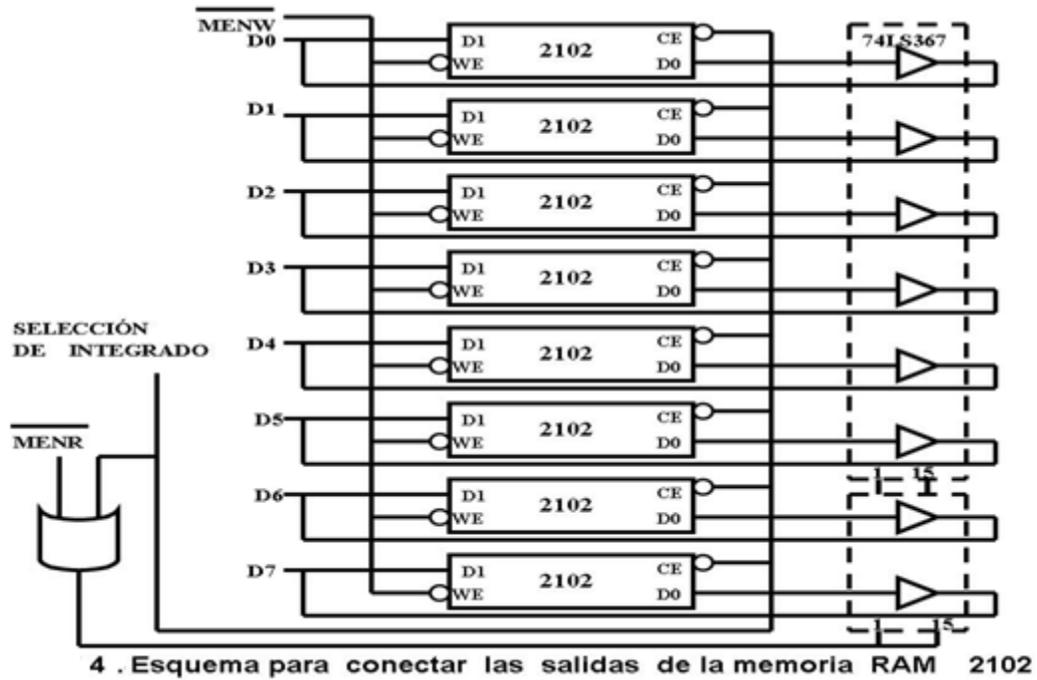


Figura 7.2 Fuente de esquemas anteriores: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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 modulo 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 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 tenga 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 de solo lectura" 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 permanentes como el BIOS (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 (ROM Programable), EPROM (ROM Eléctricamente Programable) y EEROM (ROM *Electrical* Borrable (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 solo 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^2 . 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 fuente externa de energía, tal como una fuente de 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 proteger con papel



adhesivo 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:

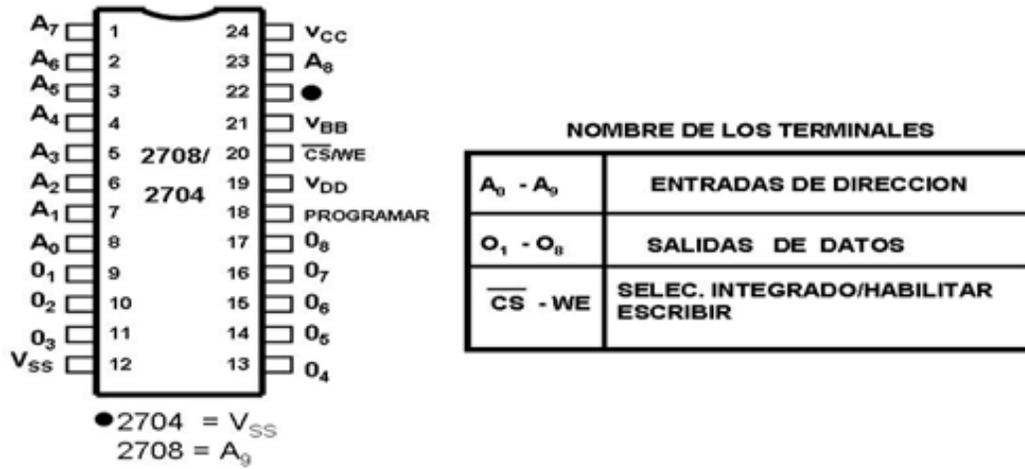
Nombre	Configuración	Alimentación	Tiempo de Acceso
1720A	256 x 8	+5, -9	650 -1700 nseg.
2708	1K x 8	+5, +12, -5	450 nseg.
2716	2K x 8	+5	450 nseg.
2732	4K x 8	+5	450 nseg.

Tabla 7.2 Características de memorias EPROM

Fuente: Con datos de Fairchild Corp. (1981). *Mos Memory Data Book*. Mountain View California, Estados Unidos de América: Fairchild.

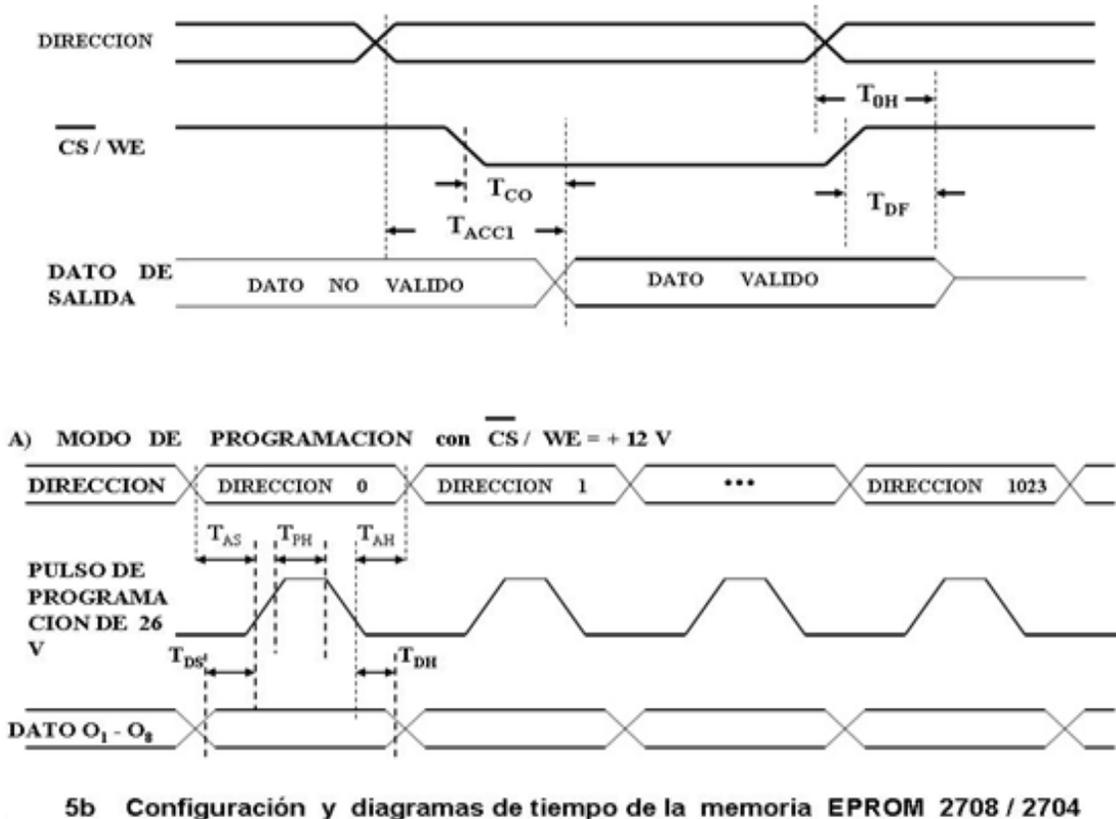
Los EEPROM son semejantes a los EPROM, pero en lugar de requerir luz ultravioleta para borrar el contenido de las localidades de memoria requieren de un voltaje aplicado en uno de sus contactos o “*pines*”.

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 la que se accede directamente el circuito interno.



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

Figura 7.3 Fuente de los esquemas: Con datos de Fairchild Corp. (1981). Mos Memory Data Book. Mountain View California, Estados Unidos de América: Fairchild.



5b Configuración y diagramas de tiempo de la memoria EPROM 2708 / 2704

Figura 7.4 Fuente: Con datos de Fairchild Corp. (1981). Mos Memory Data Book. Mountain View California, Estados Unidos de América: Fairchild.



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:

$$N \times T_{pw} \times 100 \text{ milisegundos}$$

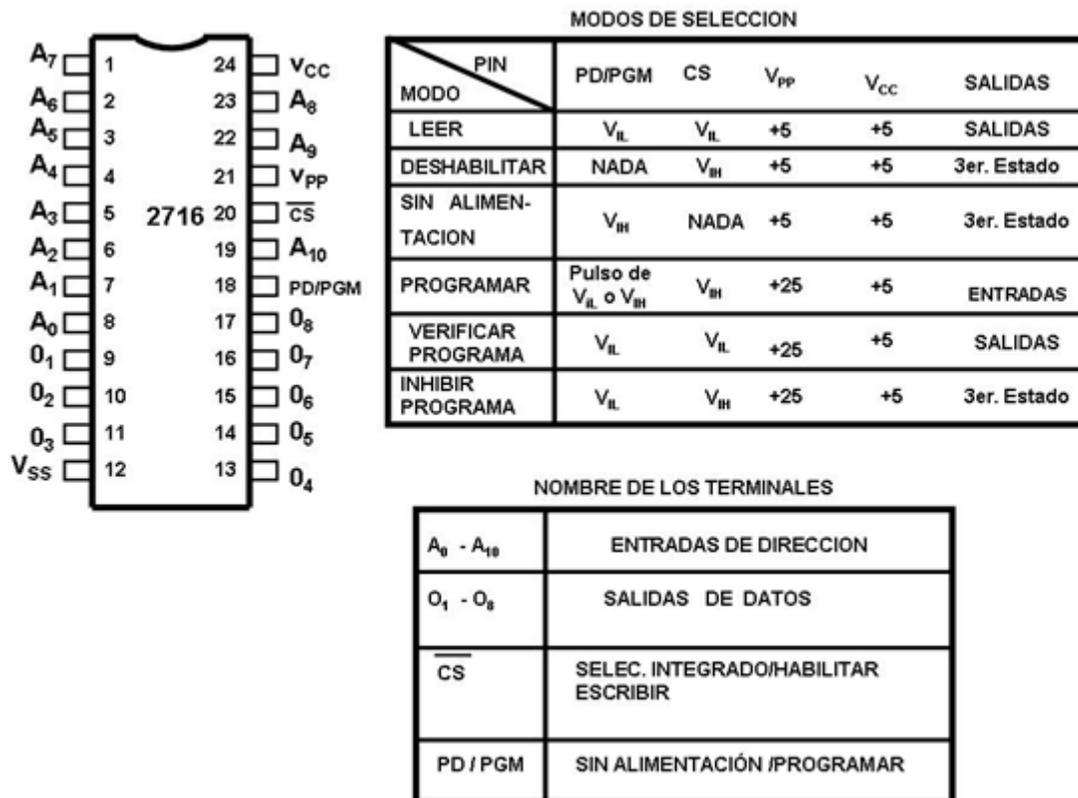
Donde

N es el número de lazo de programación requerido, y
 T_{pw} 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 T_{pw} usado. Independientemente del T_{pw} usado, siempre se debe ejecutar un lazo aplicando solo 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 V_{DD} 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.



6. Descripción de terminales del EPROM 2716

Figura 7.5 Fuente: Fairchild Corp. (1981). *Mos Memory Data Book*. Mountain View California, Estados Unidos de América: Fairchild.

La 2716 tiene seis modos de operación. Todas las entradas son nivel TTL. En la entrada V_{pp} 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. (T_{co}) 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 individual, secuencial o aleatoriamente. 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 $V_{pp} = 25V$ y $CS=0$. Al presentar una dirección, el contenido de la localidad se presenta en las líneas O0-O7.

Arquitectura de memorias ROM

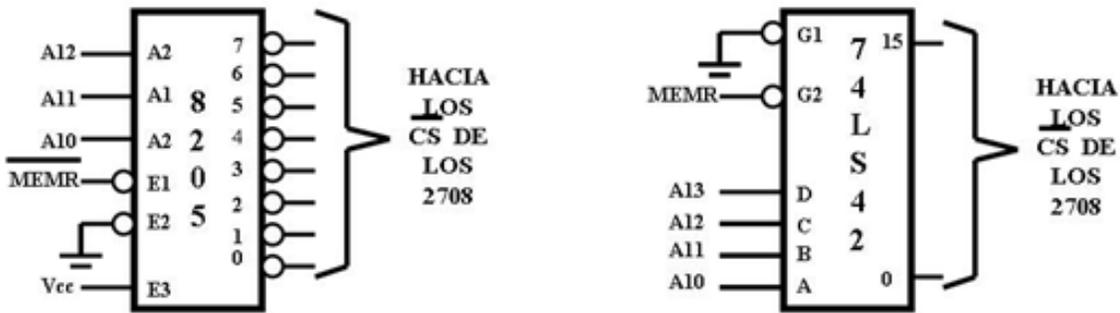
La arquitectura con memorias ROM es semejante a la arquitectura con memorias RAM. Las memorias ROM se pueden seleccionar con direccionamiento "absoluto" y "no-absoluto". Lo que se mencionará para las memorias ROM se aplica a las PROM, EPROM y EEROM.

Direccionamiento no-absoluto en memorias ROM

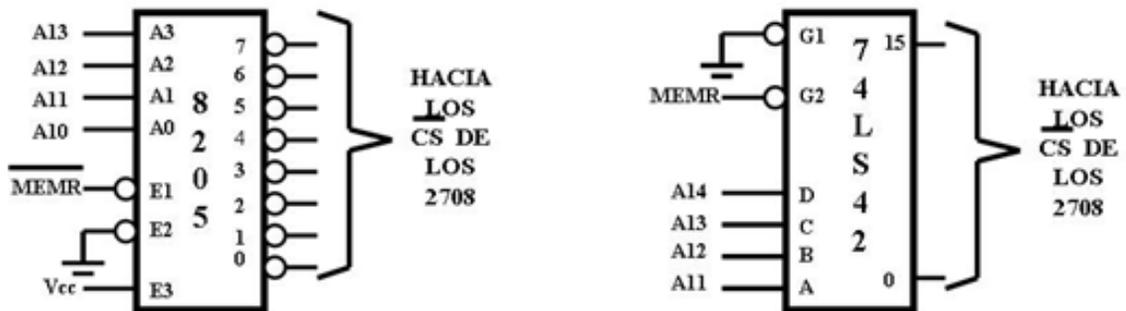
Las figuras 7a. y 7b. ilustran el uso de decodificadores para seleccionar las memorias ROM de 1K conectando las salidas de los decodificadores con las entradas CS de las ROM. Las salidas del 8205 permiten seleccionar hasta ocho memorias de 1K (2708) y las salidas del 74LS42 permiten seleccionar hasta 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

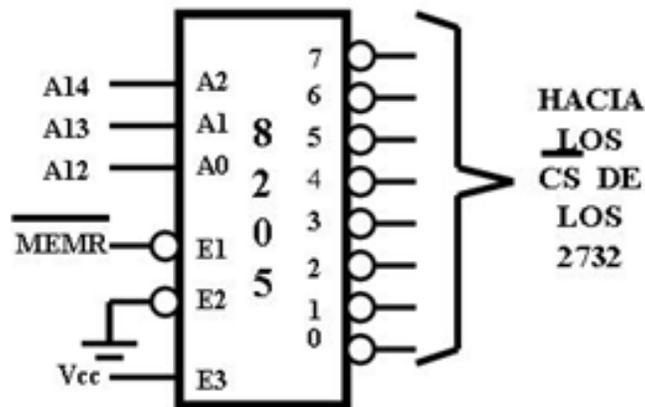
Los decodificadores de la figura 8. permiten seleccionar 8 (el 8205) y 16 (el 74LS42) memorias ROM de 2K (2716). El decodificador de la figura 9. permite seleccionar 8 memorias ROM de 4K (2732). Todos estos decodificadores proporcionan direccionamiento "no-absoluto".



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.

Figura 7.6 Fuente: Fairchild Corp. (1981). *Mos Memory Data Book*. Mountain View California, Estados Unidos de América: Fairchild.

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 tiene la característica de que la entrada G2 se puede conectar directamente por medio de un puente a la línea \overline{MEMR} ; para que las memorias conectadas a las salidas del 74LS154 se debe conectarlas a las entradas \overline{CS} de las memorias ROM. La memoria ROM conectada a la salida 0 del 74LS154 tendrá un rango de direcciones de 0000H a 03FFH, y la conectada a la salida 15 tiene el rango de direcciones de 3C00H a 3FFFH.

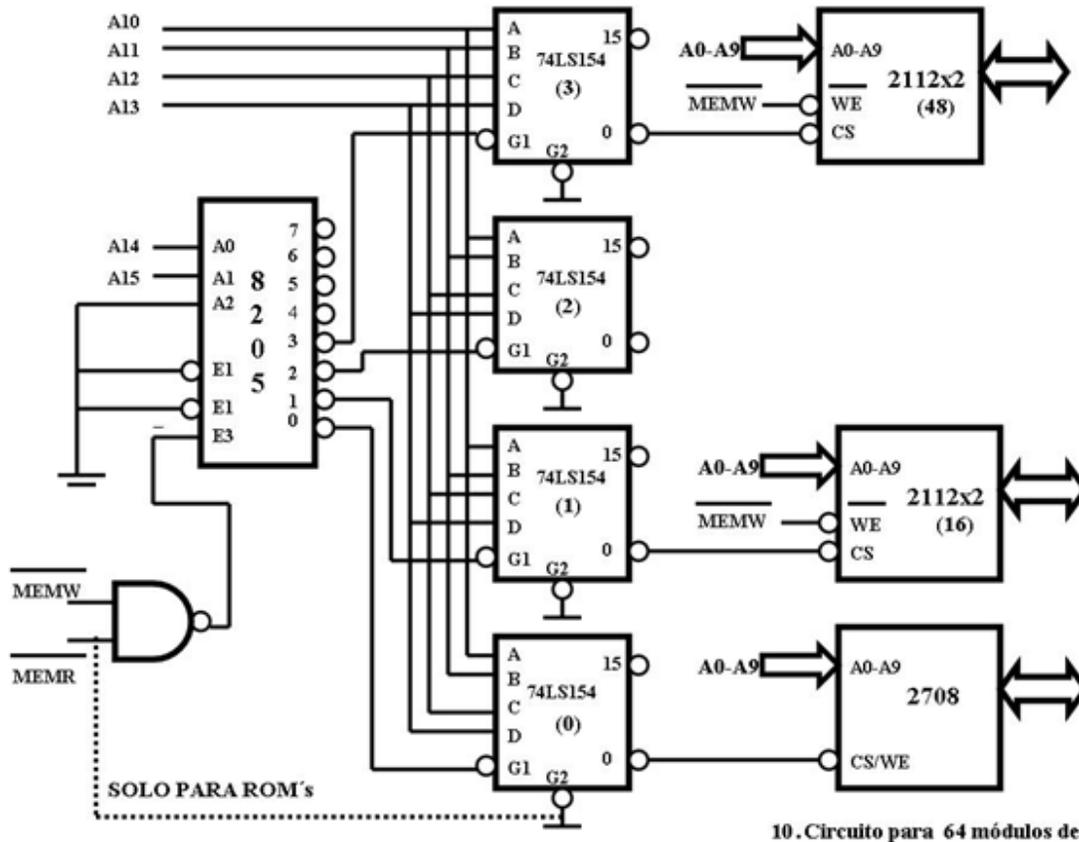
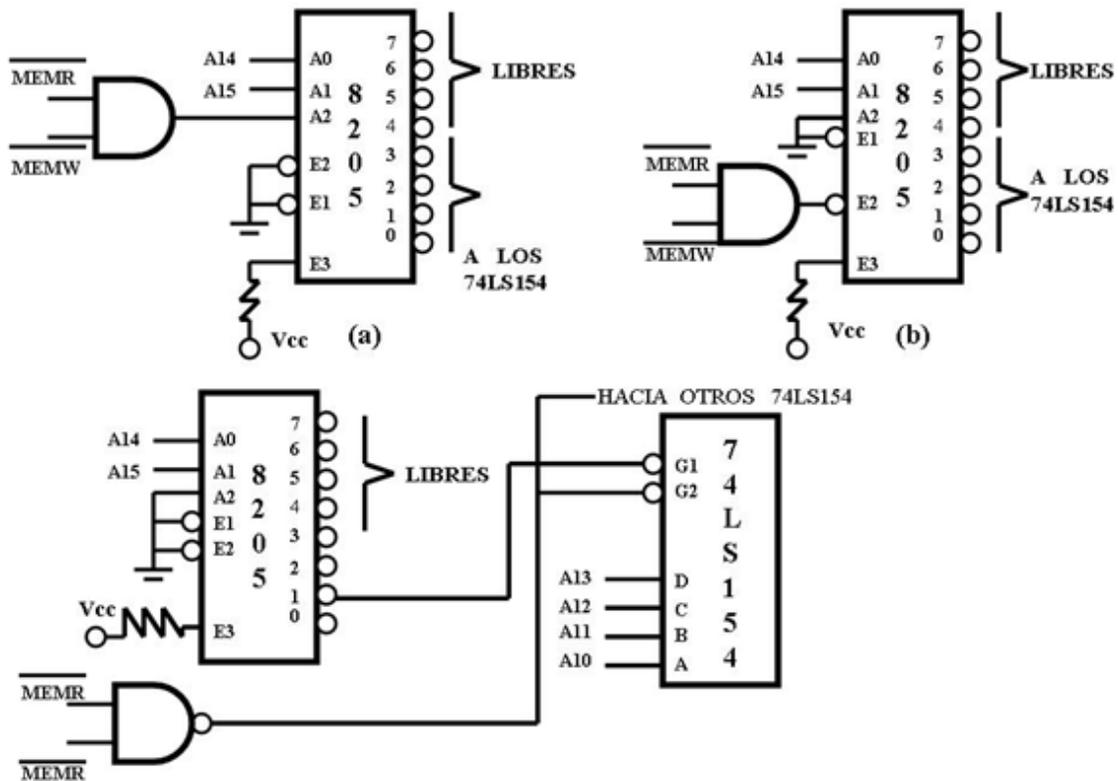


Figura 7.7 Fuente: Fairchild Corp. (1981). *Mos Memory Data Book*. Mountain View California, Estados Unidos de América: Fairchild.

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{MEMR} , para que los 74LS154 se habiliten en ciclos de leer memoria.



11. Variantes de los decodificadores de la figura 10.

Figura 7.8 Fuente: IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.

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.

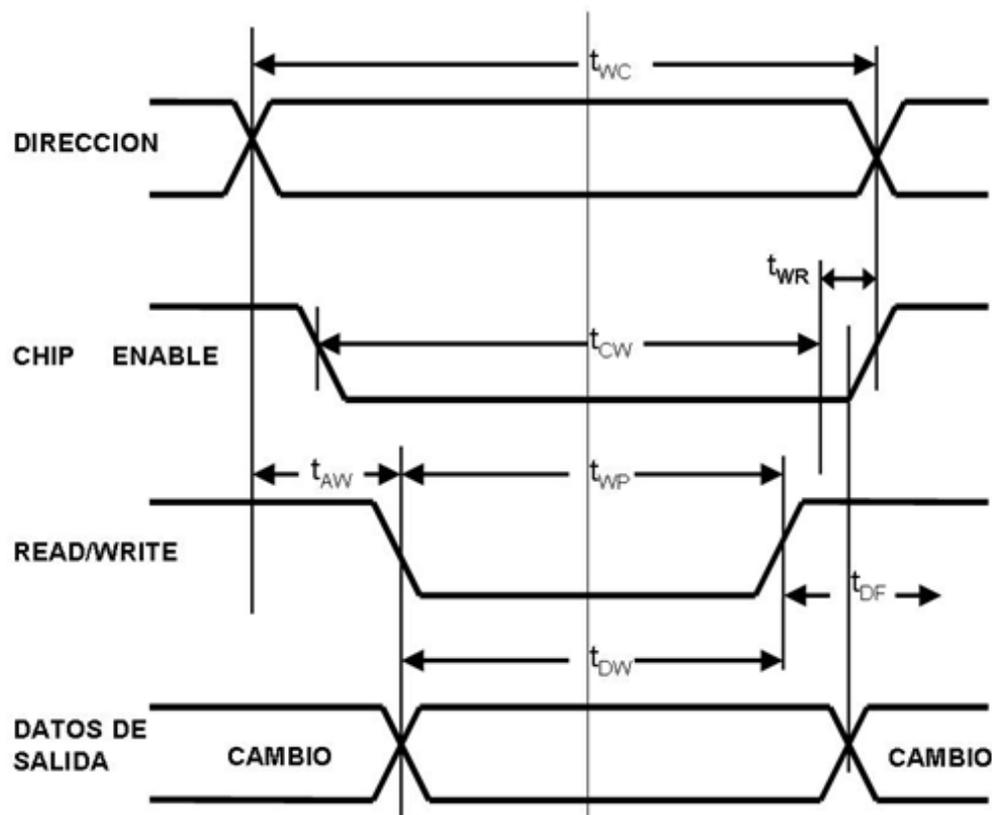
El tipo de memoria más simple, en términos de su operación, es la memoria ROM. Los pasos de operación básica de una memoria ROM son:



7.2.1. Lectura

Supón que un circuito de memoria es seleccionado con el nivel lógico apropiado en su entrada de selección (CS: Chip Select). 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 se activa para seleccionar la memoria ROM, el retraso entre la activación de las señales de selección (CS) apropiada y la estabilización del dato a la salida es t_{CO} o tiempo de acceso Chip Select. Los parámetros, t_A y t_{CO} , se muestran en el diagrama de tiempos de la figura 1. *Diagrama de tiempos para 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

Figura 7.9 Fuente: IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.



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

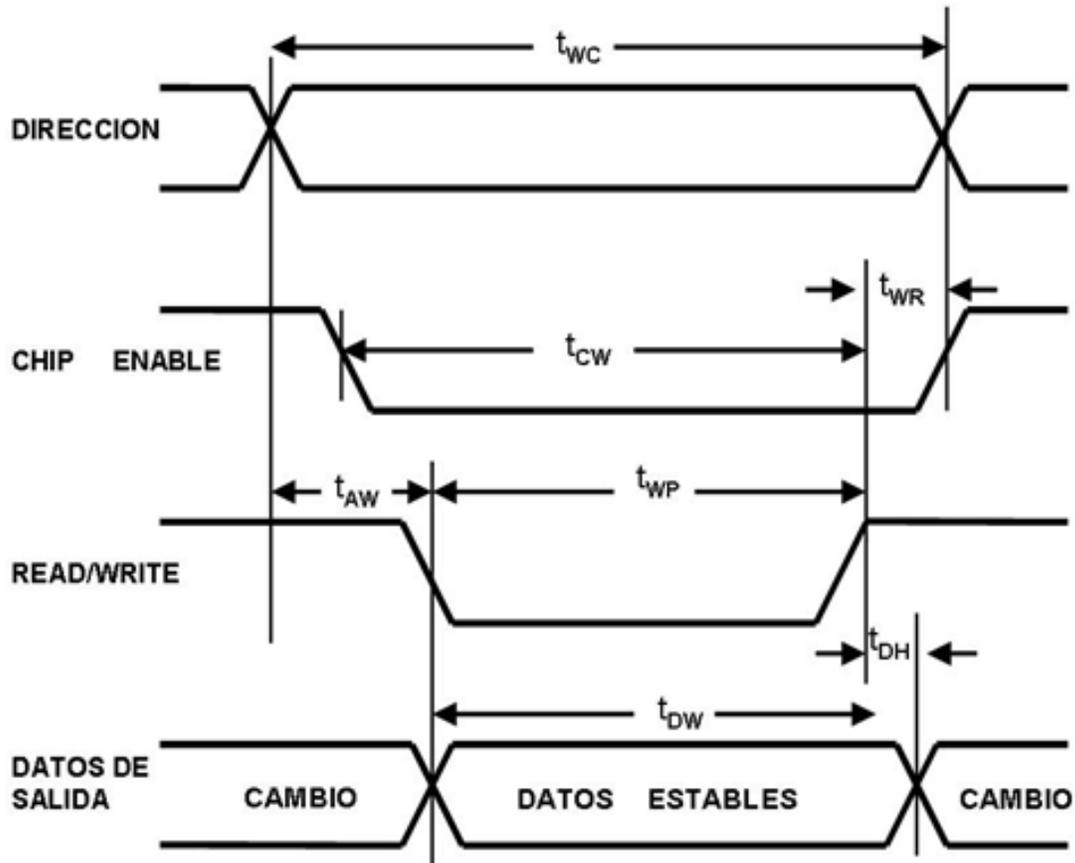
El ciclo de escritura se entiende mejor con una memoria RAM estática o dinámica. Las memorias RAM estáticas tienen los requisitos de operación más sencillos. Además de las líneas de dirección, habilitación de chip (*chip enable*) y datos de salida necesarios en las memorias ROM, las memorias RAM requieren 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 en 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 sino 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

Figura 7.10 Fuente: IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.

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 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 lectura de 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 o 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 cUSART a 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		MEMR	MEMW	I/OR	I/OW
Byte Alto	Byte Bajo				
01100000	10100000	0	1	1	1

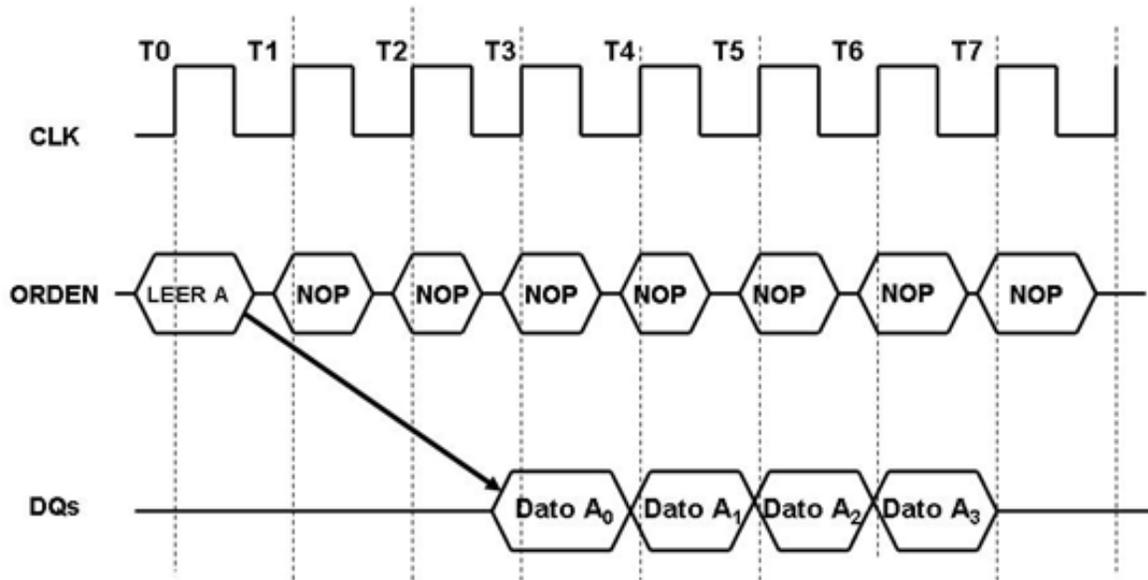


7.2.3. Refrescamiento

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. Los datos se almacenan como cargas eléctricas en condensadores, ya que los condensadores tienen una tendencia natural a descargarse, el contenido de la celda de memoria puede perderse. Por esta razón las memorias 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 de memoria.

La figura 3. *Temporizador de una lectura...* muestra un ejemplo del 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.



3 Temporización de una lectura de SDRAM (longitud de ráfaga =4,
latencia de CAS = 2)

Figura 7.11 Fuente: IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.

7.3. Mapa de memoria

Una computadora digital organiza su memoria RAM y memoria ROM en un Mapa de Memoria, para tener 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 y distribuido el espacio de memoria principal direccionada 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*. Este mapa muestra la distribución a 1MB (FFFFH, en hexadecimal).

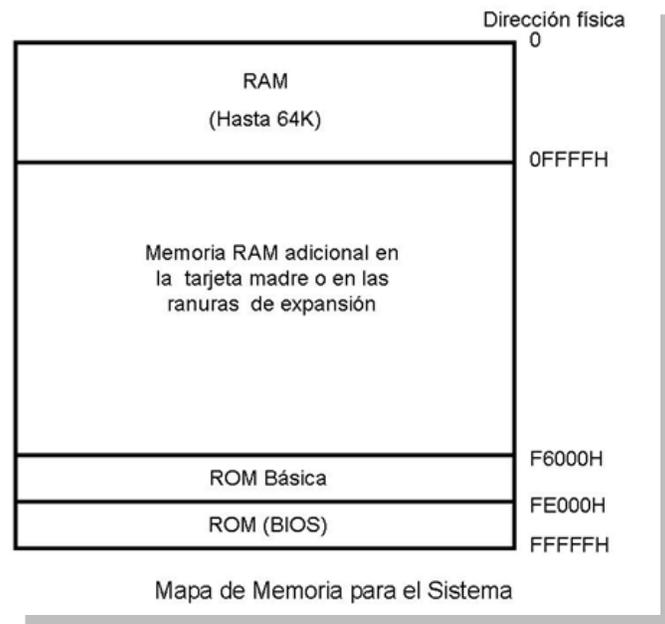


Figura 7.12 Fuente. IBM Corp. (1983). IBM Technical Reference. Boca Raton Flo., Estados Unidos de América: IBM.

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, para la primera versión de computadora personal *PC*. 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.

Figura 7.13 Observaciones en la primera versión de computadora personal. Elaboración propia.

Memoria expandida y extendida

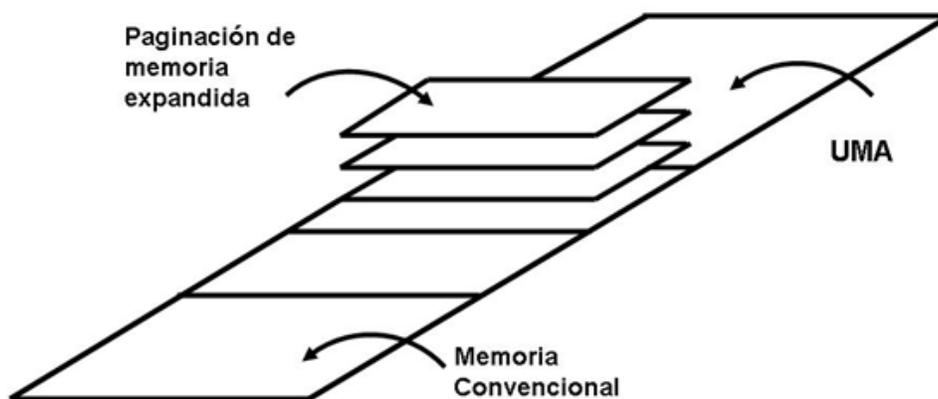
A medida que fue avanzando el desarrollo de programas y aplicaciones por parte de un usuario, los requerimientos de memoria aumentaron considerablemente. En las primeras computadoras basadas en microprocesadores de 8 bits (8085, Z80, 6800, etc.) y en algunos microprocesadores de 16 bits (8086, Z8000, 68000, etc.) se desarrollaron los conceptos de memoria expandida y memoria extendida, los cuales ayudaron a resolver cierta problemática con la demanda de memoria y para mantener la compatibilidad operativa. Imposición precisión visera

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 podía utilizarse en las máquinas que ejecutaban el sistema operativo original 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 ejecutaban 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. Solo el software compatible con **EMS** (Especificación de Memoria Expandida) puede utilizar la memoria expandida.

Como solamente podía trabajar con 64K de información a la vez, era 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

Figura 7.14 Fuente: Intel Corp. (1986). *Intel Technical Reference Manual*. Santa Clara California, Estados Unidos de América: Intel.

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 era capaz de operar en este modo, la mayoría de las aplicaciones sobre MS-DOS empleaban diversas técnicas para acceder a memoria extendida.

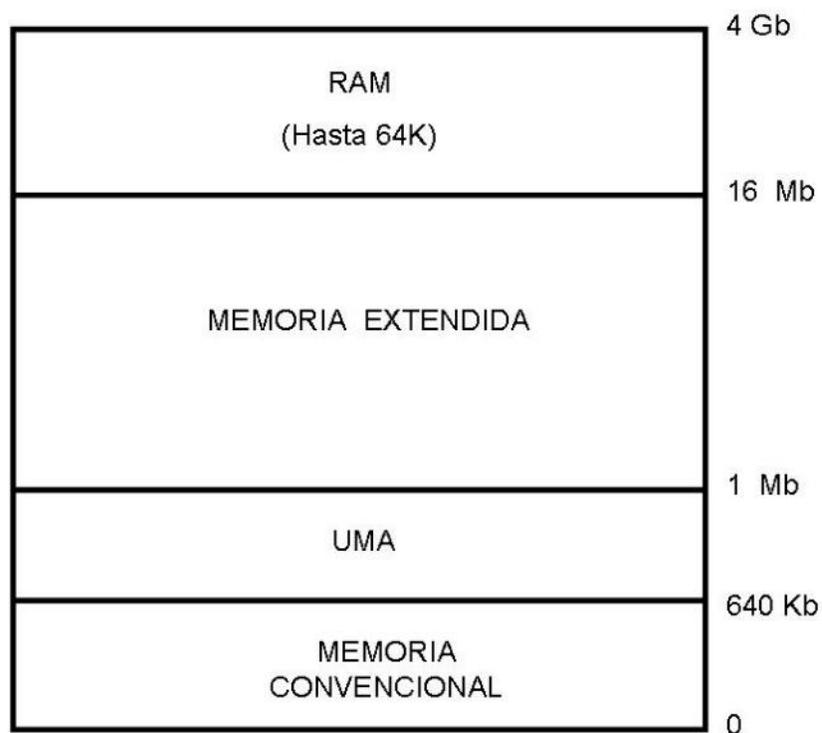


Figura 7.15. Memoria extendida. Fuente: IBM Corp. (1985). *IBM Personal Computer hardware reference library* /Technical Reference. Boca Ratón, Flo., Estados unidos de América: IBM.

En un IBM PC, compatible con un microprocesador 80286 o posterior, la memoria extendida se refería a la memoria por arriba del primer MByte de espacio de dirección. La memoria extendida estaba solamente disponible en computadoras personales basadas en el procesador 80286 de Intel o un procesador más alto. Solamente esos chips podían 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 era generalmente remapeada por arriba de 1MB, haciendo que toda ella quedara disponible a programas corriendo en modo protegido. Incluso sin este remapeo, las máquinas con más de 1MB de RAM pueden tener acceso a la memoria sobre el 1MB.

Solamente las aplicaciones ejecutándose en modo protegido podían 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 utilizaban 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, era usada por el *hardware* del sistema; 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

Corresponden a los primeros 640 Kb de memoria extendida. Esto es únicamente válido para Computadoras Personales.

Así como en la antigua PC el anterior era el mapa de distribución de la memoria, en la actualidad tenemos una reestructura acorde al crecimiento de los recursos en estos equipos, a continuación, se muestra un mapa actual de memoria.

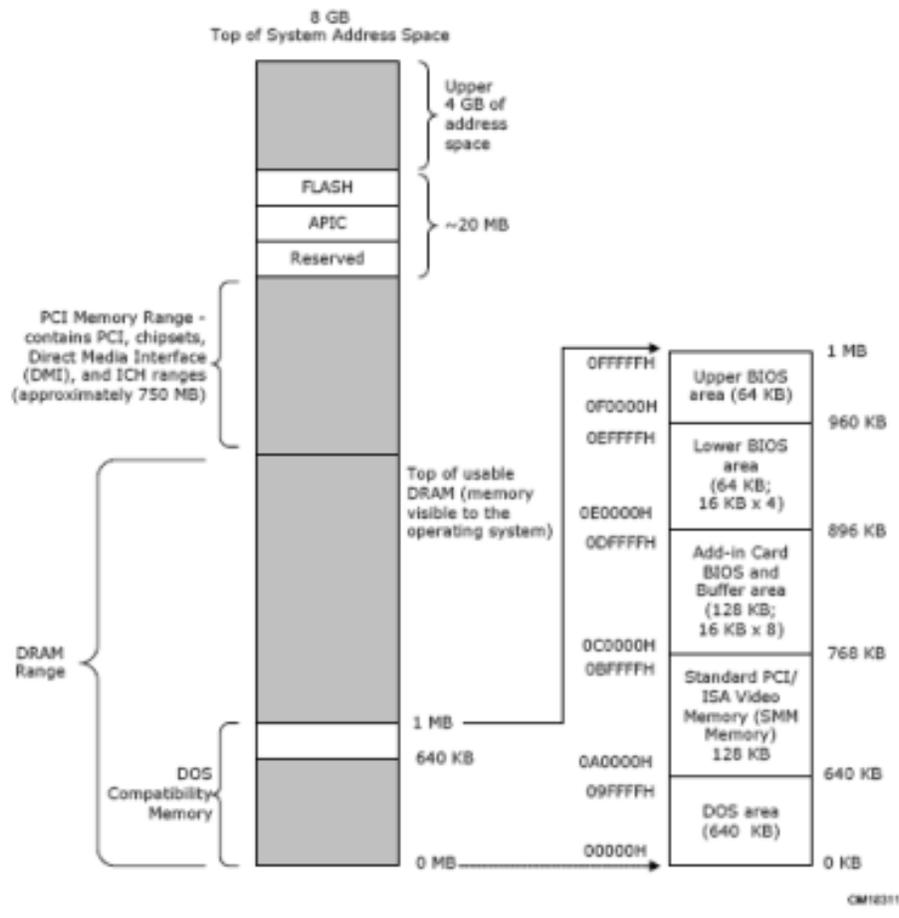


Figura 7.16 Mapa de memoria de un equipo PC actual.¹¹ Fuente: Gustavo Duarte. (2008). *Motherboard Chipsets and the Memory Map*. Recuperado 28 noviembre, 2018, de: <https://manybutfinite.com/post/motherboard-chipsets-memory-map/>

¹¹ Este mapa se asocia a una tarjeta madre para procesadores Intel dual Core™ que operan a 32 bits; observe el crecimiento a 8 GB en la cantidad de memoria y el espacio asignado para los nuevos dispositivos, con respecto a la primera configuración mostrada en la parte baja (1MB). Conforme aumentan las capacidades de los equipos, los procesadores y los sistemas operativos, se observan ajustes en su extensión y distribución.

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 el sistema de memoria. En éste la capacidad de almacenamiento se puede ampliar incrementando el número de palabras (uniendo varios bytes, así como en un principio las PC operaban con 8 y 16 bits, actualmente operan con longitudes de 64 bits) 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.

Se observa que el número de palabras en un sistema de memoria se aumenta

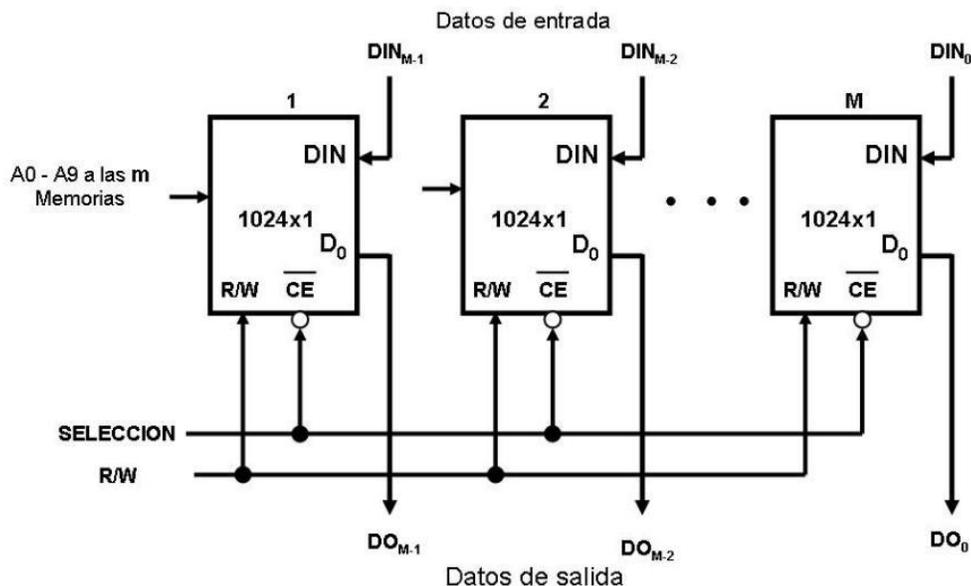


Figura 7.17 Memoria se 1024 x m bits formado a partir de circuitos de memoria de 1024 x 1
 Fuente: IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.



multiplexando 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 específicas 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-commerce* o de servicios donde se operan una gran cantidad de transacciones, el tamaño insuficiente de memoria puede redundar en un funcionamiento lento del sistema de cómputo, 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 muchas 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 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.

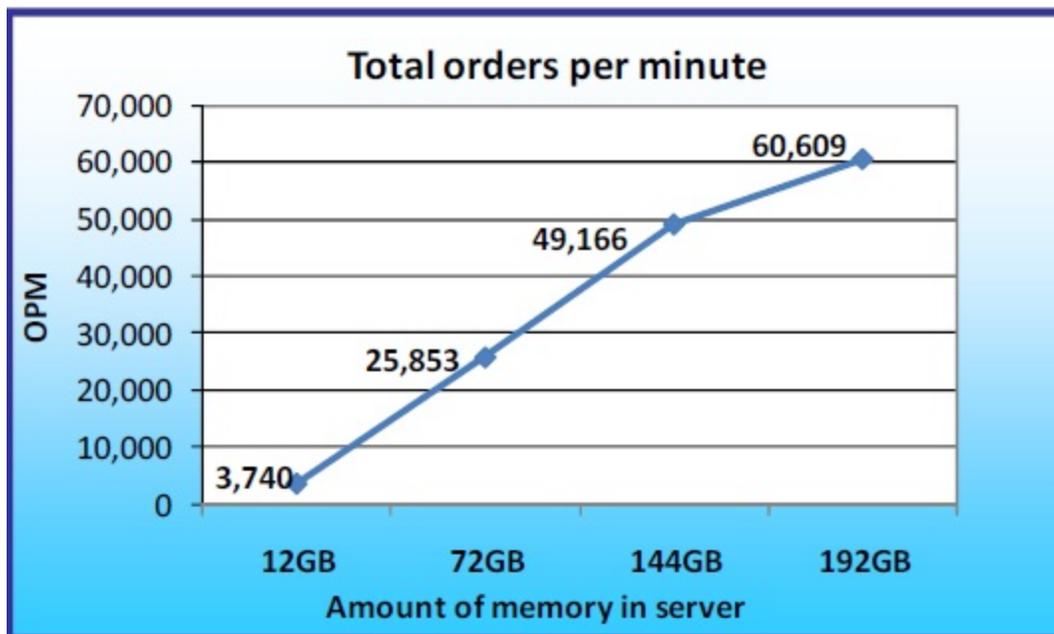


Figura 7.18 Cantidad de instrucciones por minuto (*Operations Per Minute*: OPM) para tamaño de memoria variable (www.kingston.com). Fuente: Kingston Tech. (2010). *Servers: assesing the value of adding more memory. A principled technology test summary report*, April (2010), 1–4. Recuperado el 25 de enero de 2019 de: www.kingston.com

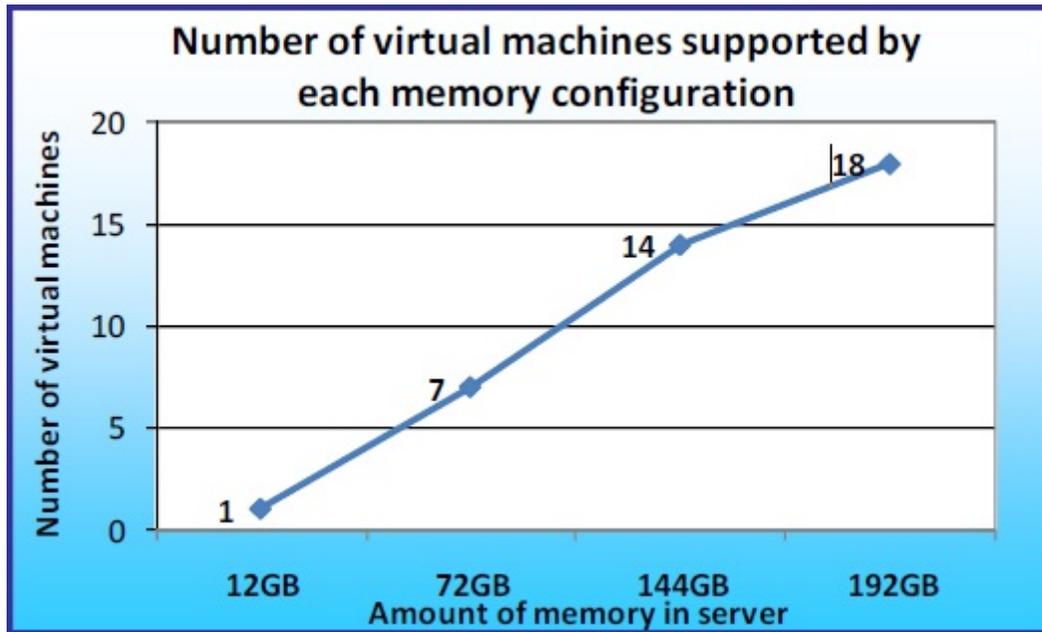


Figura 7.19 Cantidad de máquinas virtuales dependiendo de la cantidad de memoria (www.kingston.com). Fuente: Kingston Tech. (2010). *Servers: assesing the value of adding more memory. A principled technology test summary report*, April (2010), 1–4.

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.

2008. Aparece la memoria DDR3 con capacidad de hasta 8GB, velocidad de operación 800MHZ



2011. DDR4, con capacidades de hasta 16 GB, velocidad de operación 1600MHZ

2018. DDR5, con capacidades de hasta 128 GB, velocidad de operación 3200MHZ.

Sin embargo, la necesidad de cantidades mayores de memoria para aplicaciones gráficas sobre todo fotografías, audio y video también ocasiona que las memorias periféricas se incrementen en sus capacidades, 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 7.3 (Kinsgton, (2018), “Capacidad de almacenamiento de memorias”. Recuperado el 11/10/2018 de: 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 de los equipos.

7.4. Memoria caché

La memoria *caché* (del francés: oculta) es una memoria de menor tamaño, de rápido acceso y diseñada para resolver las diferencias de velocidad entre una CPU muy rápida y

una memoria principal lenta. Lo hace almacenando una copia de los datos e instrucciones de uso frecuente en el procesador. Un tamaño de memoria caché razonablemente pequeño (en cantidades cercanas a 1 MB) puede generar mejoras 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 dentro del microprocesador.



Funcionamiento

El objetivo de la memoria caché es lograr que la velocidad global de la memoria sea lo más rápida posible, permitiendo el uso de memorias semiconductoras menos costosas, ver figura *Memorias Caché y principal*. En general, 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.

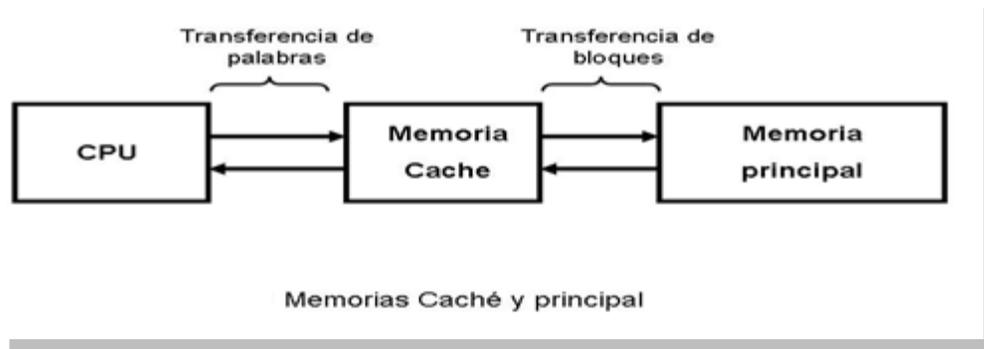


Figura 7.20 Fuente: Diana Ferreyra. (2016). *Reparación de PC*. Recuperado: 28 noviembre, 2018, de <http://reparacióndepc2016.blogspot.com/2016/05/memoria-cache-1-cual-es-el-objetivo-de.html>

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 respuesta asociado al ciclo de operación de la 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. Por cuestiones tecnológicas, 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 de computadoras 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 la mayor velocidad de operación posible al procesador y, al mismo tiempo, ofrecer la opción de operar con un tamaño de memoria principal grande pero menos costosa. 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

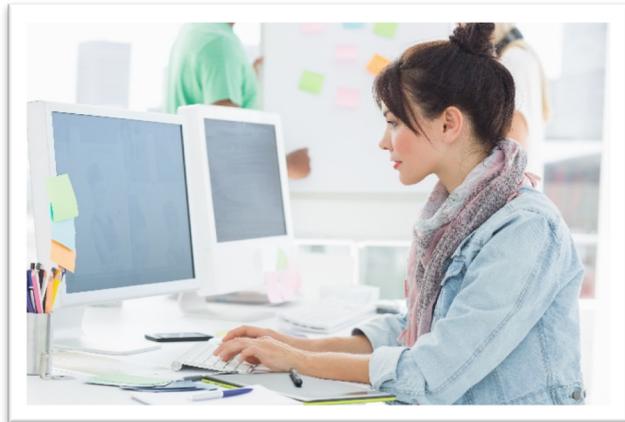
Un **tamaño de la memoria caché** razonablemente pequeño puede tener un impacto significativo en el rendimiento. Otro aspecto relacionado con la capacidad de la caché es el tamaño del bloque, el cual es la unidad de datos que se intercambia entre la caché y la memoria principal.

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.

Otro elemento de suma importancia, es el asociado a las políticas de ubicación, extracción y remplazo de los datos que se almacenan en el caché de una computadora, ya que a partir de ellos podemos optimizar el desempeño global de la computadora.

Asociado a estos elementos contamos con diferentes niveles de caches en un equipo de cómputo y se conocen por su cercanía al procesador, así tenemos caché nivel L1 que puede operar a la velocidad del procesador; caché nivel L2 que genera copias de L1 pero que también va obteniendo bloques de acceso posibles dentro del procesamiento realizado; finalmente se cuenta con una memoria caché nivel L3, que aunque con cierto desuso, puede encontrarse en sistemas que buscan optimizar el desempeño del procesador y la memoria. De esta forma podemos ver que si un procesador opera a velocidades cercanas a los 4 GHz, su ciclo de operación es de 0.25 ns , en tanto la memoria caché L1 puede operar en rangos similares, la memoria L2 en rangos de unos pocos ns ; que la memoria principal puede operar en con tiempos de acceso que van de las decenas a centenas de ns (en memorias antiguas) y los discos duros electromecánicos que tienen tiempos de acceso del orden de mili segundos (hasta un millón de veces más lentos que las memorias





caché más rápidas). De esta forma podemos darnos una mejor idea de la afectación que puede tener la mala selección y operación de una memoria caché en el desempeño de un sistema de cómputo.



7.5. Memoria virtual

La memoria virtual es la técnica que permite correr a los programas aun cuando estos no residan completamente en la memoria principal. La función de esta memoria es establecer un espacio de almacenamiento que medie entre el disco duro y la memoria principal (RAM), apoyando la transferencia de bloques de memoria entre el disco duro y la DRAM, para la paginación de aplicaciones y en situaciones de alta demanda de recursos en los programas. En adecuado funcionamiento, la memoria virtual da al usuario la percepción de que está disponible una gran cantidad de memoria principal.

El tamaño del almacenamiento virtual está limitado por el esquema de direccionamiento del sistema de computación y por la cantidad de almacenamiento externo disponible y no por el tamaño de memoria principal.

El espacio de almacenamiento direccional, es aquel en el cual las direcciones virtuales se traducen a direcciones reales.

La memoria virtual es una herramienta que permite a los programas direccionar la memoria desde un punto de vista lógico, sin importar la cantidad de memoria RAM disponible. La memoria virtual fue concebida como un método para tener múltiples trabajos de usuario residiendo en memoria principal de forma concurrente, disminuyendo el tiempo de espera entre la ejecución de procesos sucesivos, esto es, 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 completamente en memoria principal. De esta forma 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	<ul style="list-style-type: none"> • El tamaño de una tarea ya no queda sujeta al espacio libre dentro de la memoria principal. • La memoria se utiliza con más eficiencia porque las únicas secciones utilizadas del mapa de 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	<ul style="list-style-type: none"> • Tiempos del hardware y de procesador más altos. • Mayor carga general del sistema por el manejo de las interrupciones de paginación. • Incremento de la complejidad del software para evitar la híper paginación.

Tabla 7.4 Ventajas y desventajas de la memoria virtual. Elaboración propia.



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 combinación ales. Aquí se discutió como se integran los bancos de memorias a partir de estos arreglos de flip flops para el caso de memorias de semiconductores. Se describieron las diferencias entre las memorias dinámicas y estáticas y cómo afecta la arquitectura de las computadoras al tener que incluir, para las memorias dinámicas, circuitos de actualización de la información. Posteriormente se describieron los ciclos 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 constituye 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. Adicionalmente, se describió una memoria caché; los diferentes tipos y el uso que se les da en una computadora. Finalmente, se describió el uso de la memoria virtual en disco duro y su manejo a partir del concepto de paginación, lo que le proporciona a la computadora una memoria adicional a la memoria RAM instalada, con sus correspondientes limitaciones al estar utilizando espacios de almacenamiento en disco duro.

BIBLIOGRAFÍA DE LA UNIDAD



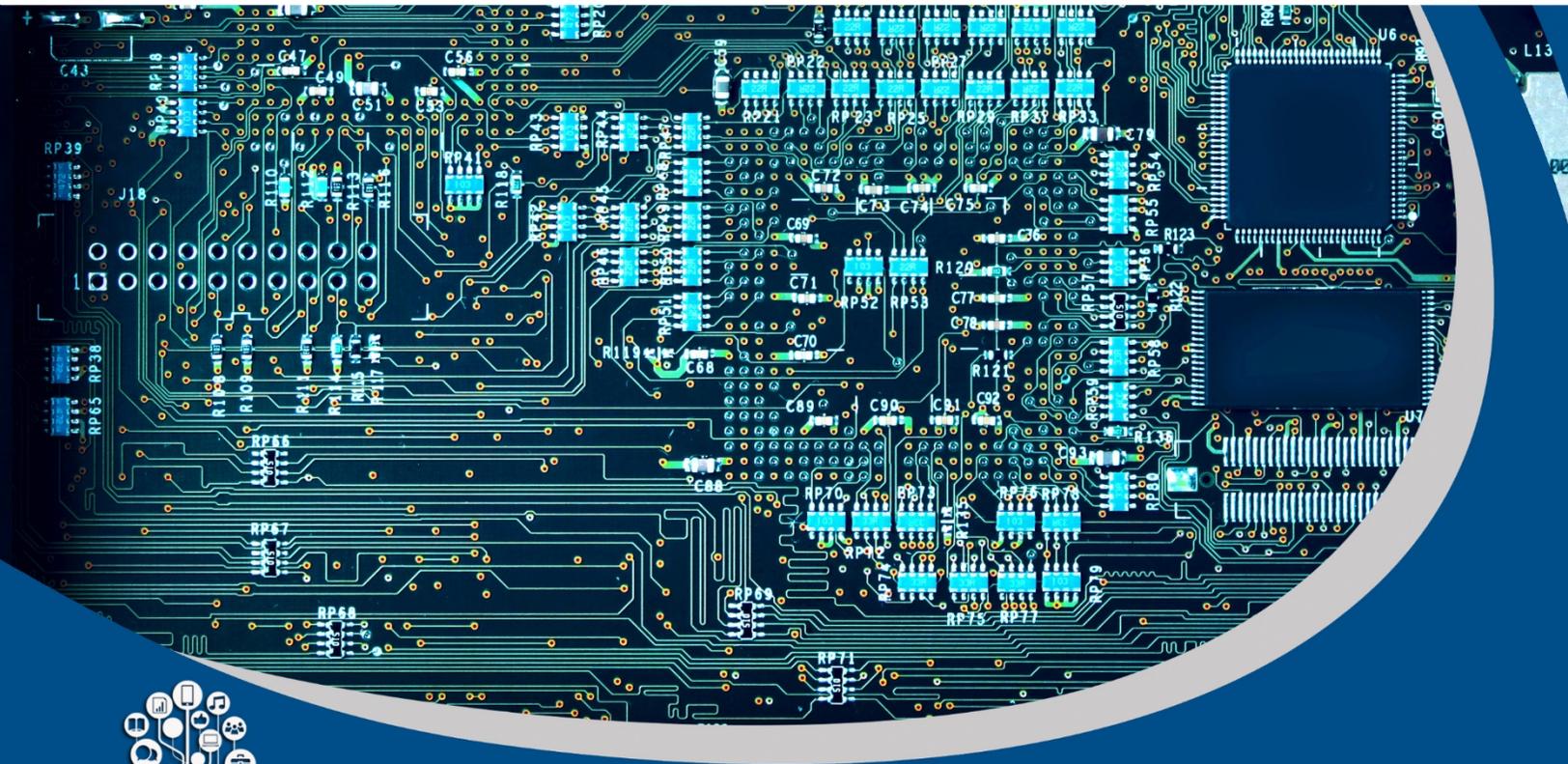
BIBLIOGRAFÍA SUGERIDA

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



UNIDAD 8

UNIDADES FUNCIONALES





OBJETIVO PARTICULAR

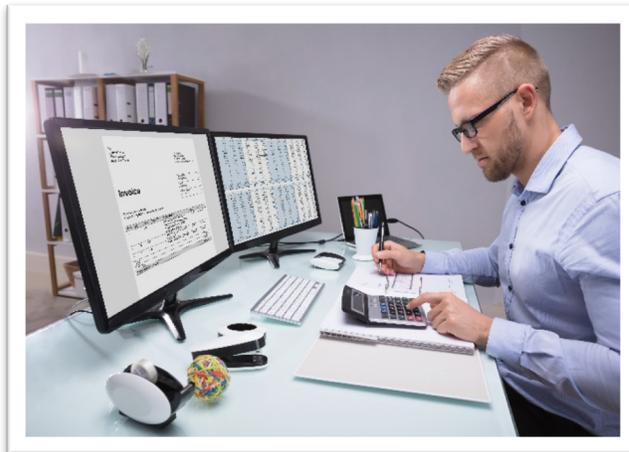
El alumno reconocerá el funcionamiento del sistema básico de entradas y salidas de una computadora, sus interrupciones, la configuración de sus puertos de comunicación y las características del almacenamiento de datos.

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. De acuerdo a la arquitectura de von Neumann la carga de procesamiento 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 las entregan procesadas, pudiendo dirigirlos a elementos de almacenamiento en la memoria principal o en la memoria secundaria que se ubica en los dispositivos de entrada y salida de esta arquitectura. Sin embargo, para poder ejecutar estas tareas se requiere de diversos

dispositivos y tecnologías que interconecten los procesos para el flujo de los datos, para el ingreso de los programas al sistema y la presentación de la información procesada. Del modelo de von Neumann sabemos que el flujo de los datos en una computadora se realiza por los buses, la unidad de control y los programas que los manejan.

Asimismo, revisaremos las formas en las que se puedan intercomunicar los diferentes dispositivos externos, entre ellos y con el sistema de cómputo, para lo

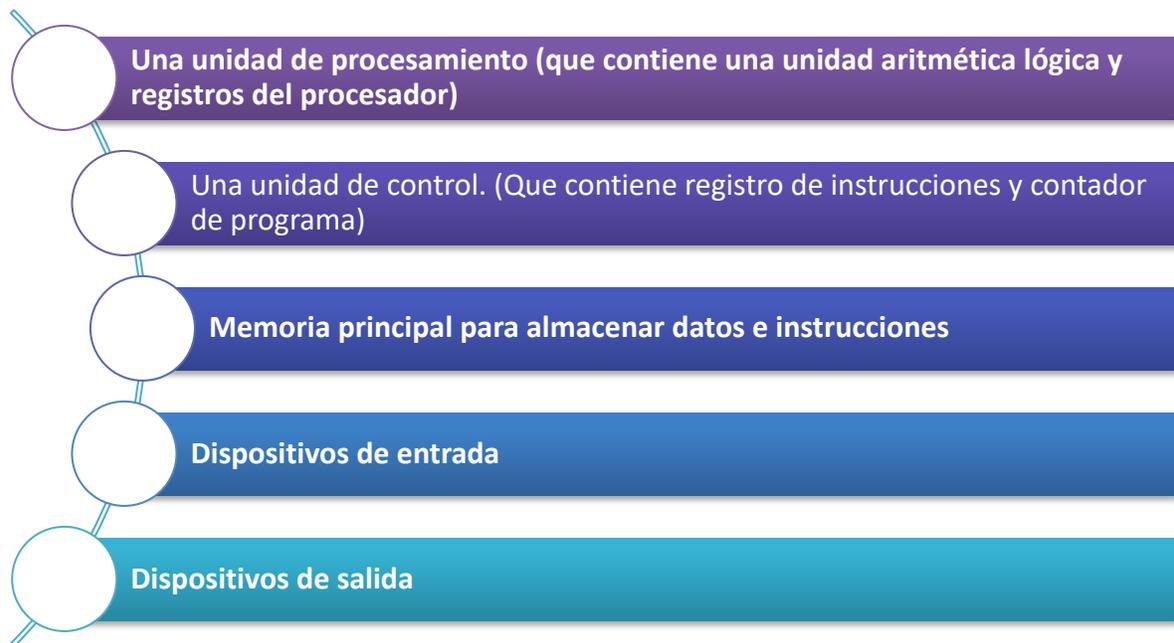


cual la arquitectura de computadoras ha desarrollado diversos recursos, como las interrupciones y los accesos DMA, que se estudiarán a lo largo de este tema.

Finalmente estudiaremos cómo es el proceso de encendido de una computadora, qué sucede desde que presionamos el botón de encendido de la computadora hasta que se despliega la interfaz de usuario en la pantalla, ya sean las amigables ventanas de control o el monótono 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: resolución explosión.



La arquitectura von Neumann es la estructura funcional que utilizan las computadoras modernas.

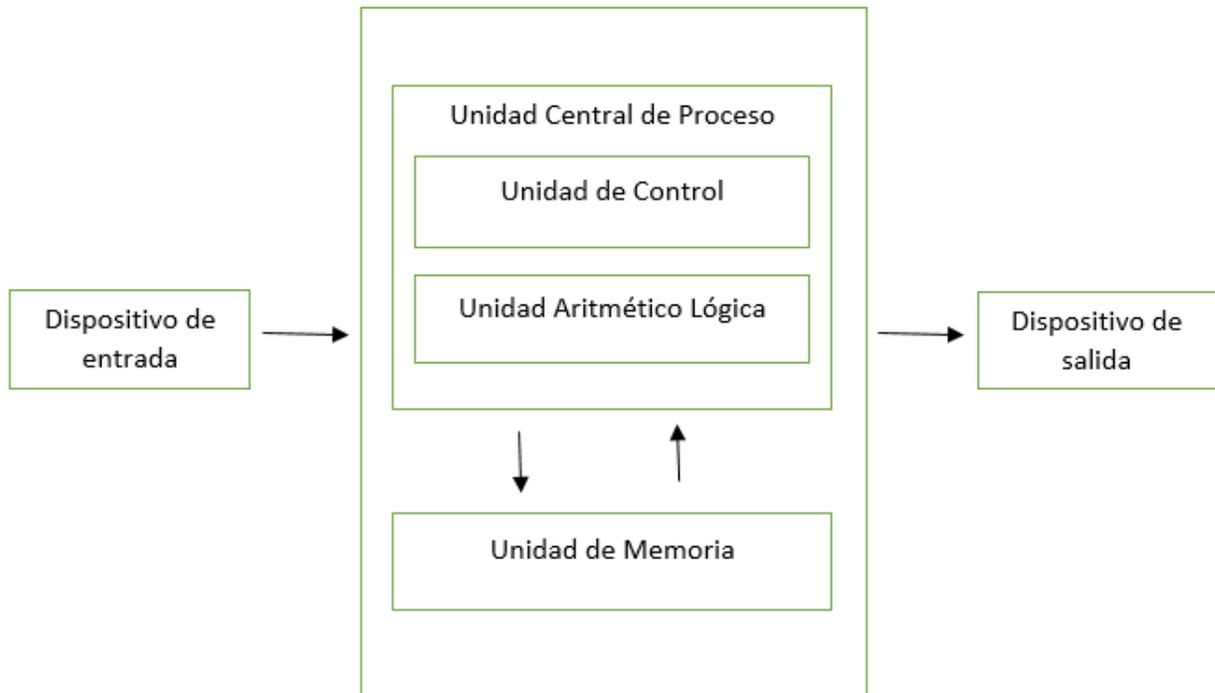


Figura 8.1 Arquitectura de von Neumann. Elaboración propia.

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

1. Procesador (UCP; unidad central de proceso), que ejecuta instrucciones, sobre la información almacenada.
2. Unidad de control (UC), consistente de algoritmos de decodificación de instrucciones y datos que provienen de la memoria del equipo. Por lo general, la UCP y la UC conforman en conjunto una estructura llamada Unidad Central de Proceso, la cual en una computadora actual se encuentra en el microprocesador.
3. Memoria del dispositivo, es un conjunto de localidades de almacenamiento, con identificadores únicos, que contienen instrucciones y datos, esto estaría representado por la actual memoria RAM.



4. Dispositivos de Entrada/Salida, que permiten la interacción del usuario y el exterior 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 impresora.

Fuente:

Gutiérrez Pedro (2013) ¿Cómo funciona la computadora actual?: Funcionamiento de la arquitectura de Von Neumann. Recuperado el 15 octubre de 2018, de: <https://www.genbetadev.com/actualidad/como-funciona-la-computación-actual-funcionamiento-de-la-arquitectura-de-von-neumann>.



8.1.1. EL BIOS

Considerando un esquema secuencial lógico iniciaremos con la descripción de los elementos básicos de una computadora. La Computadora Personal (PC, del inglés *Personal Computer*) integra diversos elementos que se han ido incorporando a la arquitectura original y que tienen una razón propia de ser, como se verá posteriormente, entre ellos podemos mencionar: el *BIOS* (*Basic Input Output System*), los canales de acceso directo a memoria (DMA: Direct Memory Access), los puertos de comunicación, etcétera.

El BIOS es un conjunto de programas que están almacenados de forma permanente en una memoria tipo ROM de PC. Cuando se inicia una Computadora Personal, el BIOS se carga en memoria y almacena el sistema operativo en memoria RAM para su ejecución. Las funciones principales del BIOS son:

- 1) Proporcionar una comunicación de *bajo nivel*, esto es, con el *hardware* (al nivel de la máquina, considerando los circuitos y dispositivos que la integran).
- 2) Configurar las características y parámetros de operación del hardware que se encuentra registrado para el funcionamiento de la computadora.
- 3) Poner en funcionamiento una interfaz mínima de usuario (Programa Monitor), que permite manejar la entrada de instrucciones por medio del teclado, la bocina (como elemento básico de salida, que emite códigos de diagnóstico audibles, tanto para condición de fallas crítica como para un funcionamiento correcto) y el monitor o pantalla que estará disponible en caso de que el sistema se encuentre en condiciones operativas, situaciones determinadas por un conjunto de pruebas mínimas diseñadas por el fabricante del equipo.

En resumen, el BIOS proporciona un programa mínimo que controla el hardware de la computadora y permite su interacción con el sistema operativo que administra el funcionamiento global de una Computadora Personal. El BIOS se almacena en una memoria ROM, la cual forma parte del Mapa de Memoria, como se ilustra a continuación.

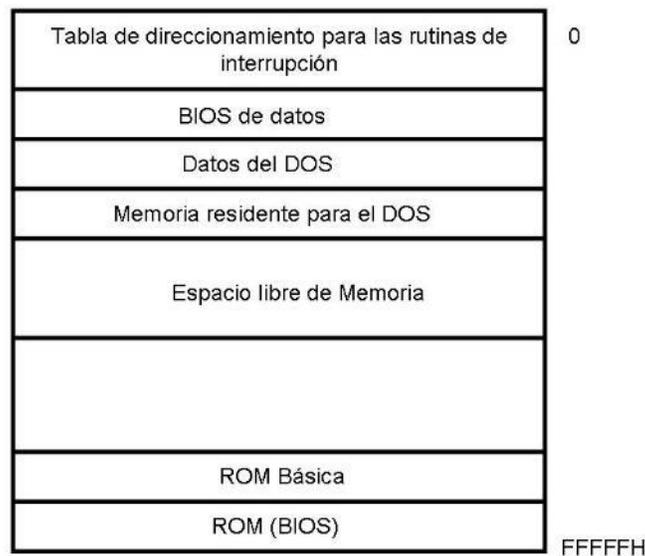


Figura 8.2 Ubicación del BIOS en el Mapa de Memoria. IBM. (1985). *IBM Personal Computer hardware reference library /Technical Reference*. Boca Ratón, Flo., Estados unidos de América: IBM.

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, incorporando internamente lo que se conoce como las rutinas o programas de control de dispositivos, almacenados en el 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.). El direccionamiento de la salida estándar (Monitor), la unidad de disco flexible, el puerto serie y el teclado de una computadora personal original basada en el microprocesador 8088 se ilustra en la tabla siguiente, observe como aún configura los modos de operación de los antiguos monitores monocromáticos (BW) y a color (RGB), todos estos dispositivos actualmente en desuso (a excepción del teclado).

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		

Figura 8.3 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

Figura 8.4 Direcciones de entrada/salida (Continuación)

Fuente: IBM. (1985). *IBM Personal Computer hardware reference library /Technical Reference*. Boca Ratón, Flo., Estados unidos de América: IBM.

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 dispositivos externos. Dicho controlador tenía dos registros ubicados en las direcciones siguientes:

20H	Registro de Comando de Interrupción
21H	Registro de Máscara de Interrupción

Tabla 8.1 Direcciones del C. I. 8259 Controlador de Interrupción

Otro dispositivo visible en una computadora personal es la impresora. Esta se puede conectar aún a la PC por medio del *Puerto Paralelo* que se encarga de enviar datos hacia el exterior en forma de palabras o Bytes (ocho bits simultáneamente). En la



arquitectura original de la PC esta unidad funcional posee 4 registros en el BIOS, 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. Cada uno de estos registros cuenta con una dirección, las cuales mostramos en seguida:

60H	Puerto de entrada "PA"
61H	Puerto de entrada/salida "PB"
62H	Puerto de entrada "PC"
63H	Registro de Comando del 8255

Tabla 8.2 Direcciones del C. I. 8255 Periférico Programable de E/S Paralelo, circuito que se integraba en las primeras computadoras personales. Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

8.1.3. Niveles de interrupción (IRQ'S)

¿Cómo comunicarse con una computadora?, es una tarea de suma importancia en su operación cotidiana y para iniciarla debemos pensar en la forma en como nosotros como personas lo hacemos; así para entablar una conversación alguien llama nuestra atención para poder iniciarla, de forma similar las interrupciones son señales recibidas por el microprocesador de una computadora personal, indicándole que debe "interrumpir momentáneamente" el curso de ejecución actual y pasar a atender una llamada a una subrutina (la realización de un proceso intermedio) para establecer comunicación con el dispositivo generador de la interrupción. Esta petición es de tipo asíncrono, esto es, puede ocurrir en cualquier momento durante la ejecución del programa en curso. Para atender las interrupciones existen dos técnicas: una utiliza al microprocesador directamente y la otra utiliza una unidad funcional conocida como controlador de interrupciones.



Atención de una interrupción utilizando un microprocesador

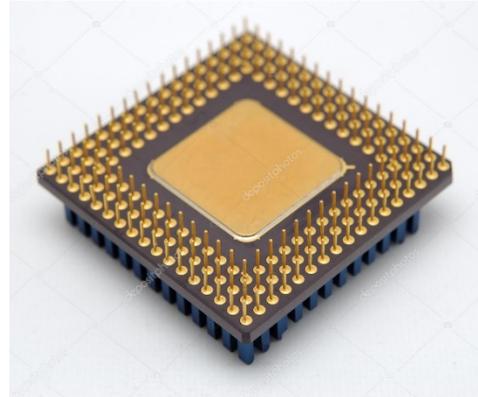
Esta primera técnica emplea al propio microprocesador, el cual se encarga de sondear (*polling*) el dispositivo cada cierto tiempo para averiguar si tiene pendiente alguna comunicación directa. El principal inconveniente de este tipo de interrupción es que es muy ineficiente, ya que el procesador constantemente 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 delegar a un dispositivo dedicado la responsabilidad de comunicarse cuando se requiera. El microprocesador, en este caso, no sondea a ningún dispositivo, sino que queda a la espera de que estos le notifiquen (le "interrumpan", por medio de una señal física *ad hoc*) cuando exista un evento como una transferencia de información, una condición de error, etc. Las computadoras personales utilizan este tipo de técnica.

Este mecanismo de interrupción consiste en cada dispositivo que requiere comunicarse con el microprocesador, debe tener asignada una línea única capaz de notificar al controlador dedicado que se le requiere para una operación particular. 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 gestionar estos eventos y puede o no estar integrado en el microprocesador (en las computadoras personales se encuentra embebido en el *chipset* correspondiente). Este dispositivo debe ser capaz de habilitar líneas de interrupción y establecer prioridades entre las distintas interrupciones habilitadas (operación llamada "enmascarar"). Cuando varias líneas de petición de interrupción se activan a la vez, el controlador de interrupciones utilizará estas prioridades para escoger la primera interrupción sobre la que informará al microprocesador.

Un microprocesador (con el controlador de interrupciones integrado) puede tener una única línea de interrupción llamada habitualmente INT. Esta línea es activada por el controlador de interrupciones interno cuando tiene una interrupción que servir. Al activarse esta línea, el microprocesador consulta los registros del controlador de interrupciones para averiguar



qué IRQ ha de atender. A partir de éste busca en el vector de interrupciones preestablecido qué rutina debe llamar para atender una petición del dispositivo asociado al evento.

En lo general, se busca que las rutinas de interrupción tomen un pequeño tiempo de ejecución y la mayoría de ellas no pueden ser interrumpidas por otras llamadas cuando se están atendiendo, esto se debe a que al entrar en cada subrutina el estado del procesador se almacena en los registros de estado en una pila dedicada y si se interrumpe muchas veces, la pila se puede desbordar por todas estas peticiones.

Pasos para el procesamiento de una IRQ

1. Terminar la ejecución de la instrucción de 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 recuperarla y continuar ejecutando el programa a partir de la última instrucción.
3. La UCP salta a la dirección donde está almacenada la rutina de servicio de interrupción (ISR, *Interrupt Service Routine*), programa que inicia, ejecuta y termina la petición del dispositivo que la generó.



4. Una vez que la rutina de la interrupción termina, el microprocesador restaura el estado que había guardado en la pila en el paso 2 y retorna el programa que se estaba usando anteriormente.¹²

Retomando nuestro ejemplo inicial a partir del microprocesador 8085^a, debemos indicar que éste cuenta con 5 líneas de entrada para señales de control por medio de las cuales los dispositivos (o circuitos) externos pueden solicitar la atención inmediata de la microcomputadora, estas señales se conocen como "**solicitudes de interrupción**". Las líneas de solicitudes de interrupción indican a la microcomputadora pausar el trabajo que está realizando en el momento de la solicitud para atender las llamadas de los dispositivos externos. La transferencia de datos por medio de un controlador de interrupciones hace más eficiente el uso del tiempo de procesamiento de las microcomputadoras.

Las interrupciones permiten que eventos como alarmas, fallas de energía y periféricos obtengan una atención inmediata de la CPU; así el programador no necesita tener al microprocesador verificando continuamente si ocurren estos eventos.

Tipos de interrupciones

Existen tres modos básicos de interrupción: por línea, multinivel y vector.

¹² Altamira Aldo. (s.f.). "Arquitectura de computadoras". Recuperado 28 noviembre, 2018, de <http://compuarqui.blogspot.com/2015/02/interrupciones.html>



Por línea

Una señal de interrupción a la entrada **INTR** del controlador correspondiente originará que inicie una serie de acciones bien definidas. Varios dispositivos pueden utilizar concurrentemente esta entrada a través de una compuerta OR. La interfaz de un dispositivo originador debe tener una bandera (señal de interrupción) que se activa cuando el dispositivo solicita una interrupción. La computadora puede determinar por programa qué dispositivo solicita atención, una vez que determina cuál es el dispositivo solicitante, la computadora le dará atención por medio de la rutina de servicio específica.

Multinivel

La microcomputadora proporciona varias líneas independientes de interrupción y cada una originará diferentes acciones específicas. En este tipo de interrupción, se pueden conectar uno o varios dispositivos, ya que generalmente cada línea tiene pre asignada una acción en el programa de atención correspondiente.

Vector

El dispositivo solicitante debe proporcionar un vector (dirección) de interrupción a la computadora después de que realiza la solicitud de interrupción y ésta se acepta por la computadora. Este vector de interrupción le indica a la microcomputadora en qué localidad se encuentra el programa de atención correspondiente. debe continuar para darle atención al dispositivo que se encuentra solicitando la interrupción.

En nuestro caso de ejemplo, 8085A, éste utiliza la técnica de interrupción por vector y multinivel; para entender cómo funcionan estas técnicas de interrupción, revisaremos la instrucción *Restart* (**RST**), la cual es muy importante para el procesamiento de las interrupciones.



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**, el 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**, como se muestra a continuación:

	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 obtienen con la instrucción **RST** tienen entre sí una diferencia de 8H localidades.



Instrucción	Código	Dirección
RST 0	C7H	0000H
RST 1	CFH	0008H
RST 2	D7H	0010H
RST 3	DFH	0018H
RST 4	E7H	0020H
RST 5	EFH	0028H
RST 6	F7H	0030H
RST 7	FFH	0038H

Tabla 8.3 Instrucciones RST. Fuente: Con datos de Héctor Miranda Lohse (s.f.). *El Microprocesador 8085*. Recuperado 28 noviembre, 2018, de: <https://es.slideshare.net/mirandafhector/el-microprocesador-8085>

Líneas de interrupción del 8085A

La 8085A tiene 5 entradas por medio de las cuales los dispositivos externos pueden. Interrumpir al procesador, 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 dispositivo externo por el Bus de Datos. El 8085A tiene un registro (flip-flop INTE) que controla las solicitudes de interrupción que recibe por esta línea, tiene la característica de poder ser condicionada su atención, por lo que se dice que este tipo de interrupción es de tipo "*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 del CPU. La línea TRAP genera solicitudes de interrupción "no mascarables", es decir, no hay máscara o bandera (flip-flop) que lo controle, por lo que no se puede deshabilitar.



RST 5.5

Ordena al 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. El 8085A tiene una máscara que controla las solicitudes de interrupción que se reciben por esta línea, acepta o no las solicitudes. Esta línea también es "mascarable".

RST 6.5

Al igual que la línea RST 5.5 pero carga el Contador del Programa con la dirección 0034H.

RST 7.5

Funciona de la misma forma que la línea RST 5.5 pero carga el Contador del Programa con la dirección 0003CH.

El microprocesador 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 X.X. 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). Esta instrucción habilita o deshabilita por programa la atención de las interrupciones mascarables.

El procedimiento de aceptación de interrupciones implica que cuando el 8085A "acepta" una solicitud de interrupción externa su lógica interna deshabilita en forma automática el flip-flop INTE, con lo que se retira durante el evento la posibilidad de que den más solicitudes de interrupción "mascarables" hasta que se ejecute por



programa la instrucción EI. Al activarse la línea RESET IN (se reinicia completamente el procesador) 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 y atiende una solicitud y al activarse la línea RESET IN.

El microprocesador 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 la dirección correspondiente), siempre y cuando estén habilitadas las interrupciones (flip-flop INTE = 1) y no tengan activas 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) INTÉR



Las interrupciones están configuradas de tal forma que el 8085A acepta primero la solicitud de interrupción de la línea de mayor prioridad. Este esquema no toma en cuenta la prioridad de la línea que está siendo atendida.



8.1.4. Canales DMA

En la operación de una computadora puede haber interrupciones de diversa índole, de esta forma podemos notar que cuando introducimos datos por un teclado se introduce un carácter a la vez, sin embargo, en condiciones diferentes, podemos requerir introducir completo un archivo de datos de gran extensión en bytes. El esquema anteriormente mostrado puede ser muy lento para introducir un byte a la vez, situación por la que se han definido otras estrategias, de esta manera se establecieron los canales de acceso denominados DMA (Direct Memory Access), que permiten establecer un canal de comunicación directo entre el dispositivo externo y la memoria principal de la computadora. De esta forma el acceso a un archivo desde las memorias USB y una transferencia de video desde el controlador de red pueden ser realizadas de esta manera.

En las primeras computadoras personales, el procesador era la parte principal de la máquina, es decir, el procesador hacía prácticamente todo. Aparte de procesar las instrucciones de los programas, también era responsable de transferir datos a los periféricos. Sin embargo, esta forma de operación demeritaba sustancialmente el desempeño global de la computadora y a causa de ello se han ido añadiendo dispositivos y funciones especializadas en su arquitectura que mejoran el rendimiento de las computadoras.

La invención del concepto de DMA permitió liberar tiempo de proceso del procesador y que los periféricos transfieran los datos sin su intervención, con la consiguiente mejora del rendimiento. Algunos controladores especiales fueron creados permitiendo la transferencia de información sin que el procesador controlara estas transferencias.

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.¹³

¹³ (s/a) (s/f) ¿Qué son los canales DMA? Recuperado el 12/03/2019 de: <http://www.ordenadores-y-portatiles.com/dma.html>



Las computadoras personales incluyen en su arquitectura el controlador DMA en la tarjeta madre, con hasta ocho canales para la transmisión de datos.

Un inconveniente de los canales de DMA es que monopolizan el bus de datos y si dos dispositivos tratan de usar el mismo mecanismo DMA al mismo tiempo, la información se podría mezclar entre los dos equipos que pudieran buscar utilizarlo. Por esta razón es que la tecnología asociada ha evolucionado en el arbitraje de estos posibles conflictos, y actualmente hay tecnologías, como la PCI, donde un dispositivo se apropia del canal de manera controlada dependiendo de sus diversas prioridades, por ejemplo, video, audio, archivos, etc.

Arquitectura del controlador de DMA

Regresando a nuestra arquitectura ejemplo, mostramos el Diagrama a bloques del circuito 8237, circuito perteneciente al *chipset* MCS85 del microprocesador Intel 8085A.

EL circuito permite hacer transferencias por DMA; el CPU tiene 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 $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ y $\overline{\text{CS}}$ (chip select), y las líneas de datos DB0-DB7. Cuando se hace una transferencia por DMA, el C.I. 8237 toma las líneas de control y dirección del CPU y genera a partir de las líneas CLOCK y READY, las señales de control $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ e $\overline{\text{IOW}}$.

En esta condición, las líneas de datos DB0- DB7 se convierten en las líneas de dirección A8-A15 durante la transferencia por DMA.

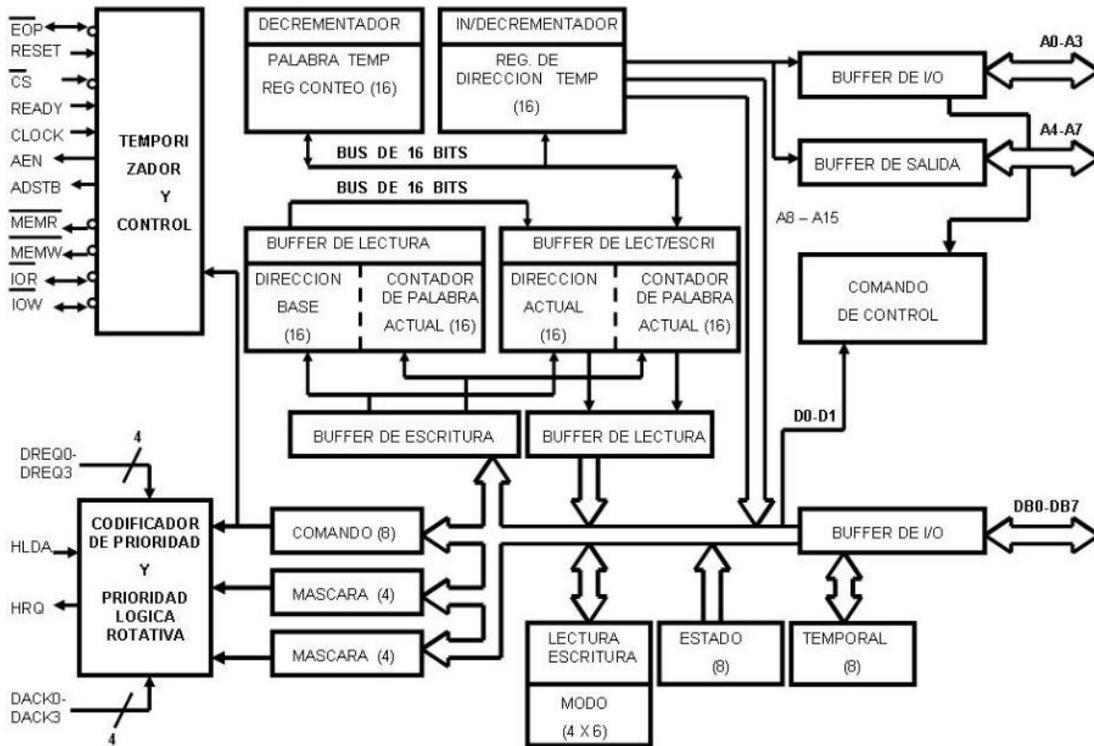


Figura 8.5 Diagrama a bloques del 8237 de Intel

Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: 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 solo 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 de E/S seleccionados. 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 (esto es que se desconecte del bus de datos



y direcciones y se mantenga en pausa). El CPU le indica al 8237 que puede empezar a controlar los canales de datos y direcciones activando la línea HLDA. En esta condición el 8237A empieza su transferencia DMA. La línea $\overline{\text{EOP}}$ (*fin de proceso*) del 8237 es bidireccional. Cuando se usa como salida, $\overline{\text{EOP}}$ 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, $\overline{\text{EOP}}$ 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 activar. En la computadora, los canales tienen una prioridad fija, el canal 0 tiene la prioridad más alta y el canal 7 la más baja.

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 o 1), y la cuenta actual disminuye. La transferencia continua hasta que la cuenta actual llega a cero (cuenta FINAL), o hasta que el "pin" $\overline{\text{EOP}}$ 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 tasas de transferencia substancialmente menores que un byte por ciclo de máquina, 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 ser deshabilitados individualmente, 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 usaba en las computadoras personales tenía cuatro canales DMA. Dos de los cuales se utilizaban en la operación primaria de la computadora: el canal 0 se usaba en la función de actualización de memoria dinámica de la unidad del sistema, y el canal 2 se usaba para transferir datos entre el controlador de disco flexible (sistema de almacenamiento de datos que ya está en desuso) y memoria. Los canales 1, 2 y 3 estaban disponibles en el bus para uso de interfaces instaladas en las ranuras de expansión. 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 solo los contenidos de registros 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 8.4. Direcciones del DMA. Fuente: Intel Corporation. (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.

2 8237 DMA Control Unit (DMAU)

2.1 Block Diagram

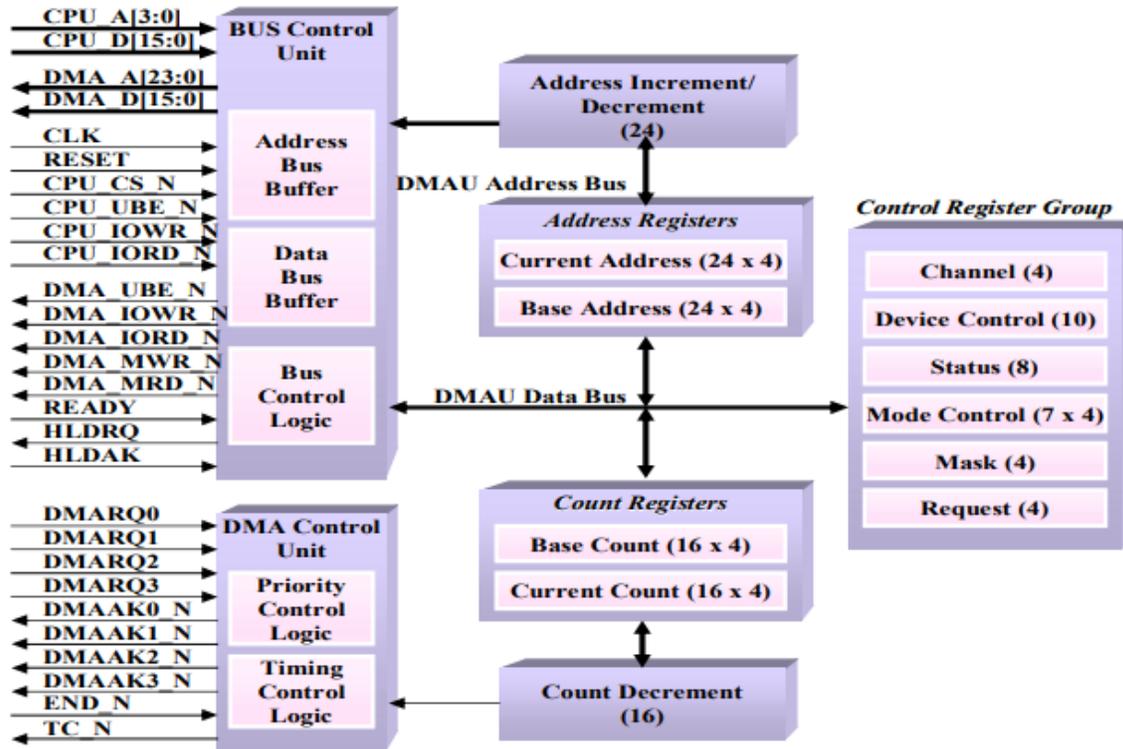


Figura 8.6 Diagrama de la unidad de control del circuito integrado Intel 8237, controlador de DMA, tomado de la hoja de especificaciones del fabricante. Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.



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 exactamente el puerto paralelo 8255 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*.

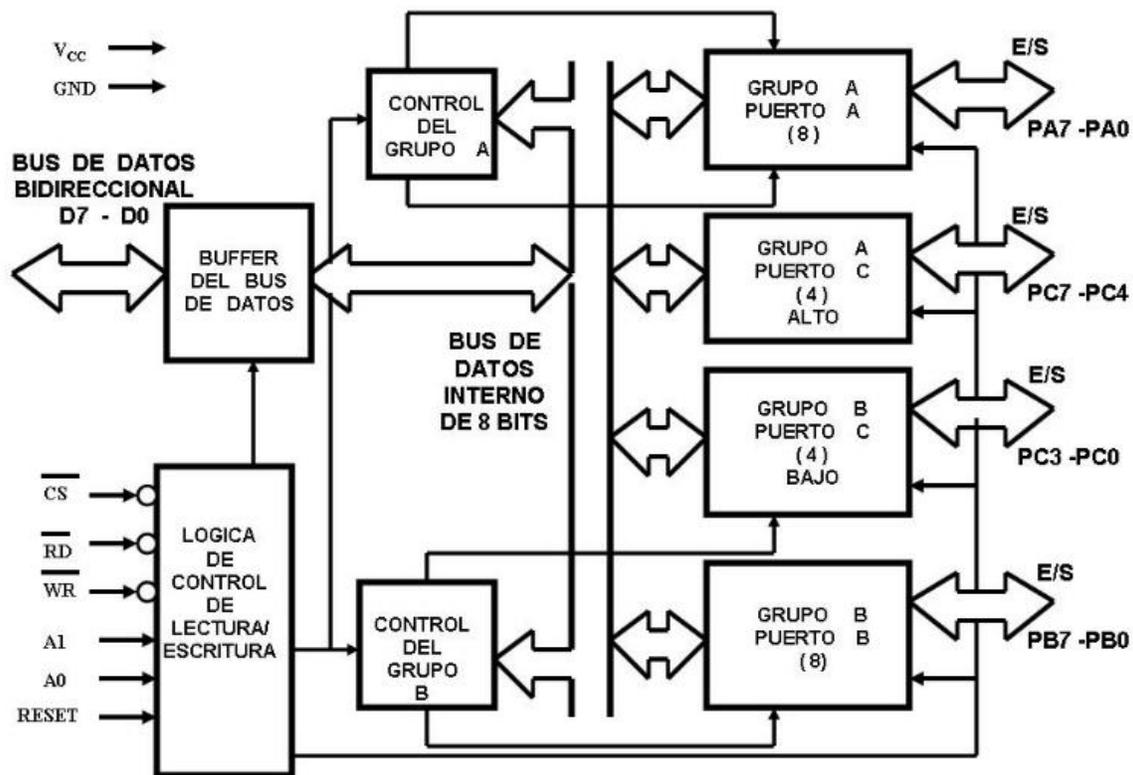


Figura 8.7 Diagrama lógico de la 8255

Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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 8.5. Operación Básica del C. I. 8255.

Elaboración propia con base en: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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)

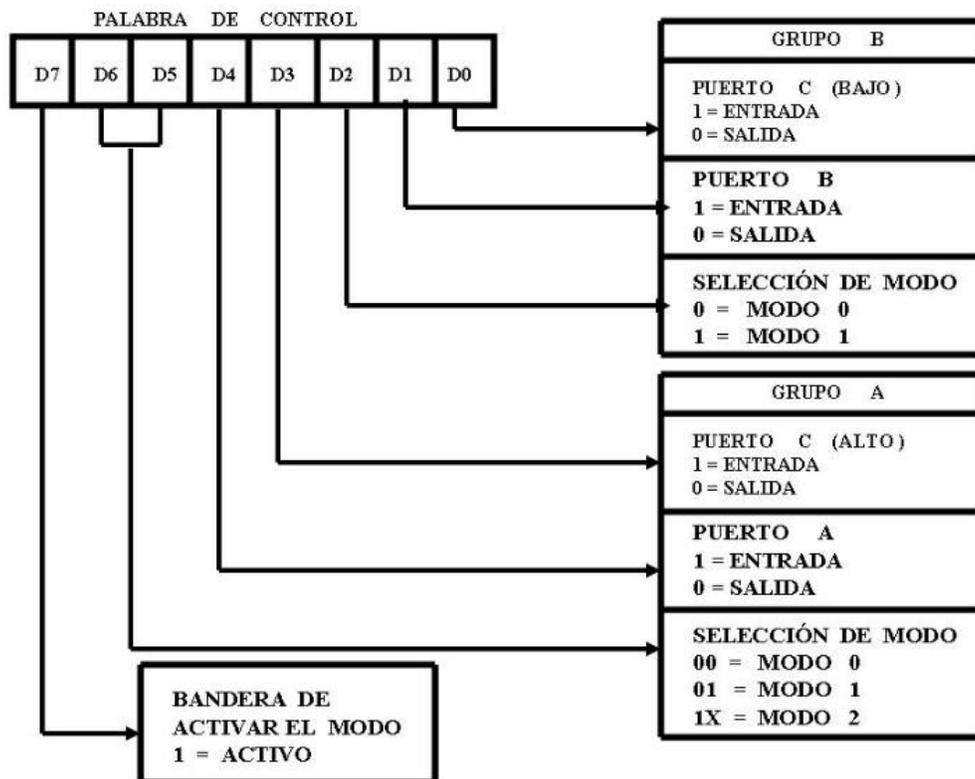


Figura 8.8 Formato de definición del modo. Fuente: Intel Corporation. (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.



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.

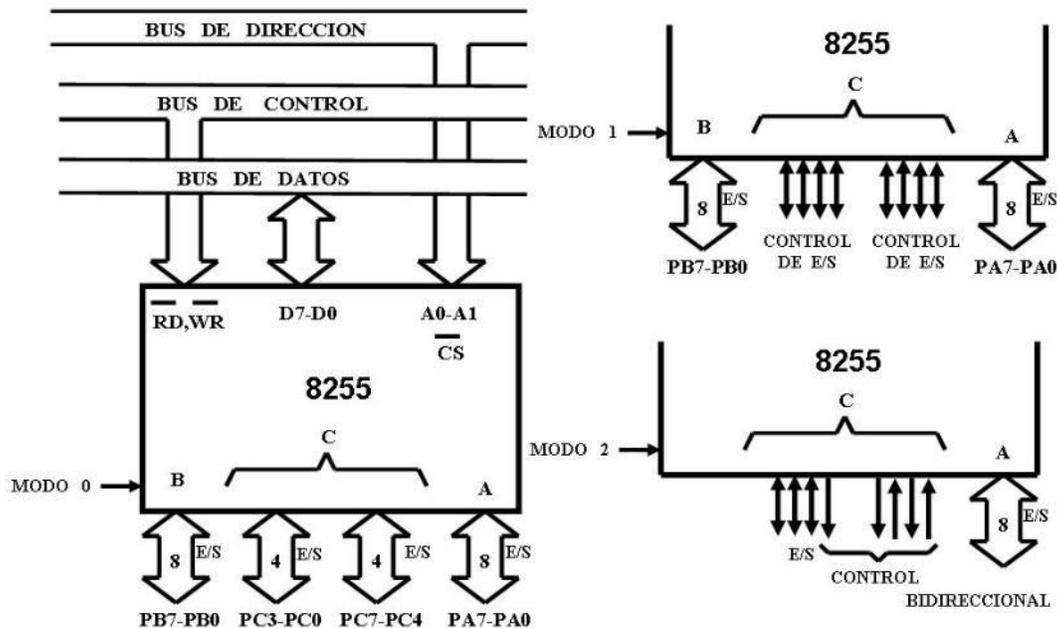


Figura 8.9 Definiciones de modo básico. Fuente: Intel Corporation (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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.

Conector paralelo tipo Centronics de doble conector con 25/36 contactos (izquierda) para un dispositivo externo, utilizado anteriormente en las impresoras, actualmente sustituido por cables tipo USB doble conector con 4/4 contactos (derecha, USB 2.0) que da prestaciones de protocolo de comunicación versátil, mayor velocidad de transmisión de datos y menor tasa de errores de comunicación.¹⁴



Tipo de puerto	SPP	PS/2	EPP	ECP
Fecha de Introducción	1981	1987	1994	1994
Fabricante	IBM	IBM	Intel, Xircom y Zenith Data Systems	Hewlett Packard y Microsoft
Bidireccional	No	Si	Si	Si
DMA	No	No	No	Si
Velocidad	150 Kbyte/seg.	150 Kbytes/seg.	2 Mbytes/seg.	2 Mbytes/seg.

Tabla 8.6 Héctor Miranda Lohse. (s.f.-b). *El puerto paralelo*. Recuperado de <http://etitudela.com/fpm/comind/downloads/elpuertoparalelo.pdf> p. 2.

¹⁴ Para fines educativos. La imagen puede extraerse de la siguiente fuente:
<https://www.cables2u.co.uk/parallel-ieee1284-centronics-printer-cable-p-229.html>



	Velocidad de transferencia	Voltaje soportado	Configuración del cable	Disponibilidad
USB 1.1	12 Mbps	2.5V, 500mA	Tipo A - Tipo B	1988
USB 2.0	480 Mbps	2.5V, 1.8A	Tipo A - Tipo B	2000
USB 3.0	5 Gbps	5V, 1.8A	Tipo A - Tipo B	2008

Tabla 8.7 Tipos de puerto y características para el estándar Paralelo (vía Centronics) y USB, obsérvese cómo la velocidad tope bajo “ECP” justifica la desaparición de la interfaz Paralela (año 2000:16 Mbps (paralelo) vs 480 Mps (USB)). SPP: Standard Paralell Port; EPP: Enhanced Paralell Port, puerto paralelo mejorado; ECP: Extended Compatibility Port, puerto con compatibilidad extendida. USB: Universal Serial Port.

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.

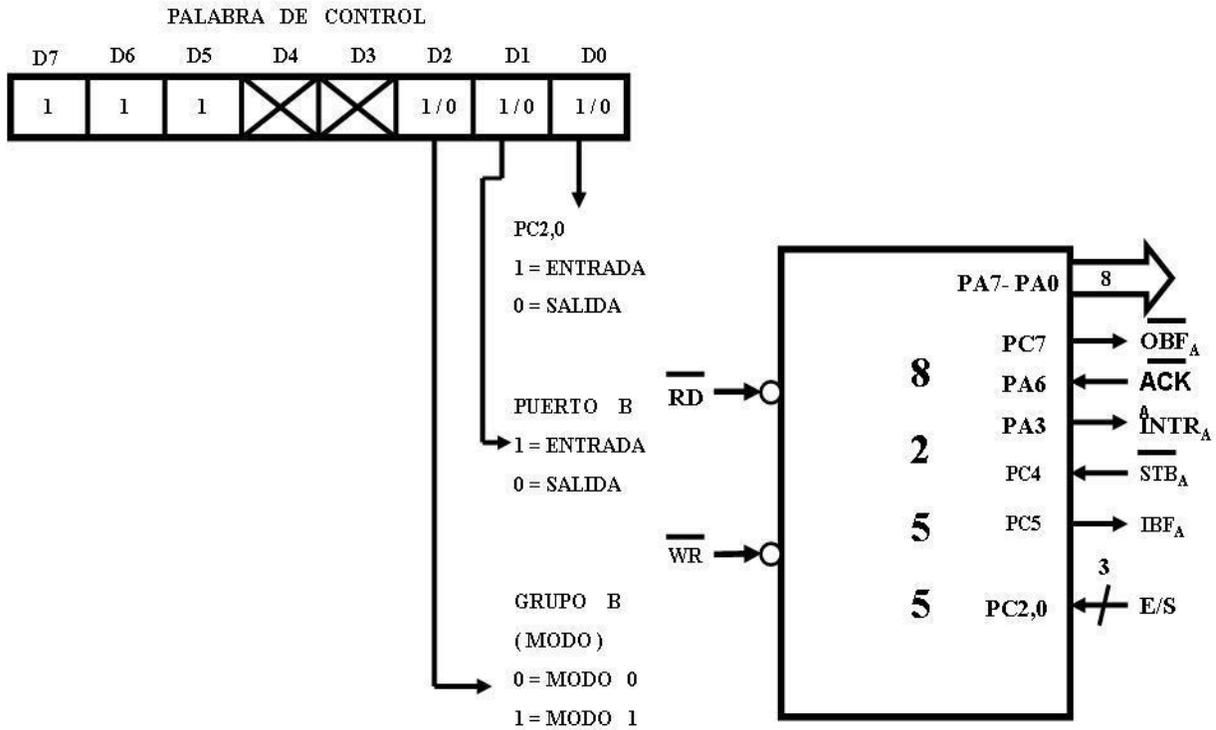


Figura8.10. Palabra de control y configuración en el Modo 2 Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel. El modo 2 no utiliza las terminales PC0, PC1 y PC2, por lo que el puerto B se puede programar en el modo 0 o en el modo 1, la tabla *Combinaciones con el modo 2* ilustra las combinaciones con el modo 2.

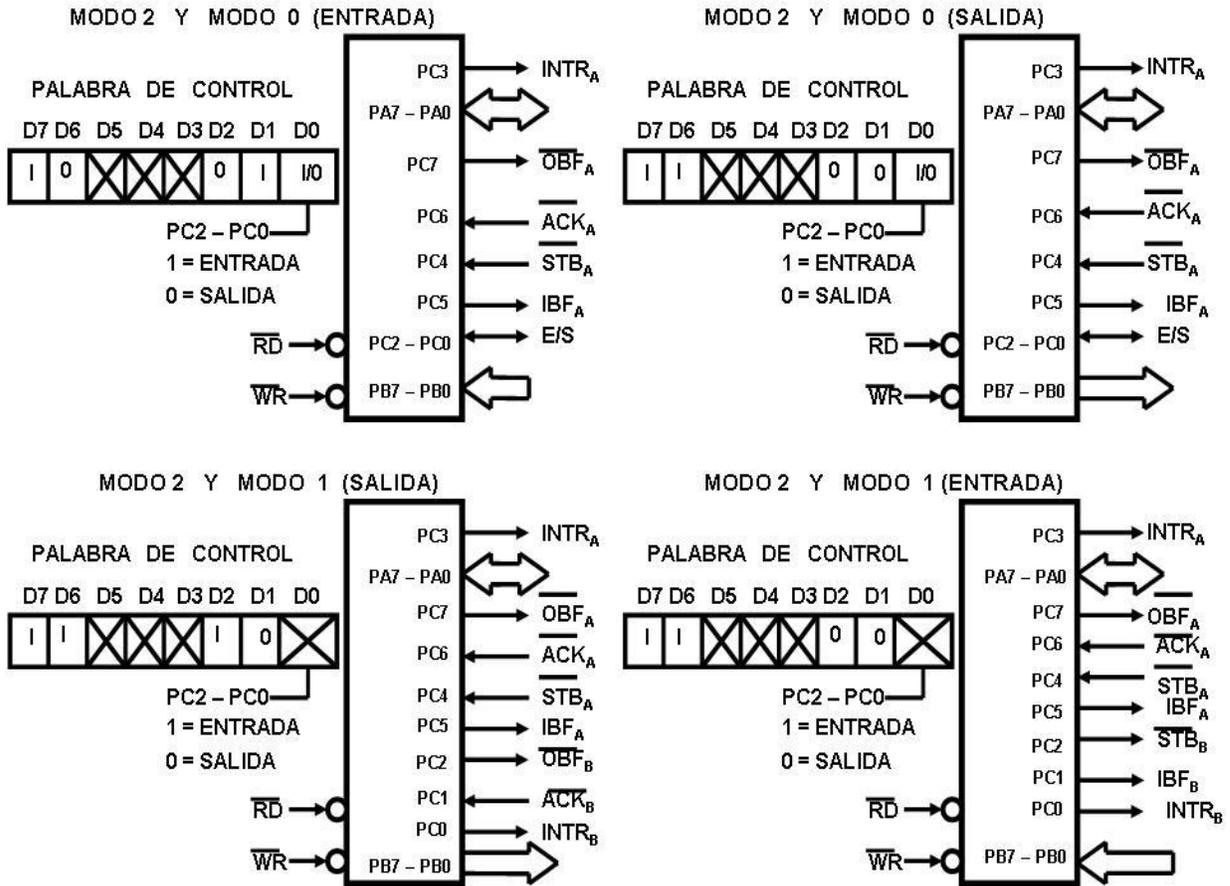


Figura 8.11 Combinaciones con el modo 2. Fuente: Intel Corporation. (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.

Puerto serial (Protocolo de Entrada y Salida Serie)

Una computadora también permite la comunicación en forma serial (secuencias de bits transmitidas un bit detrás de otro) 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.

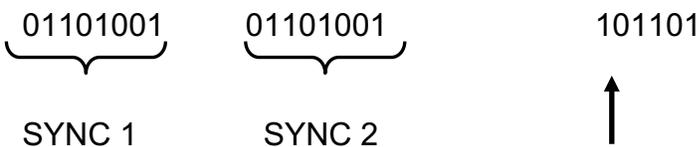
Transferencia síncrona

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



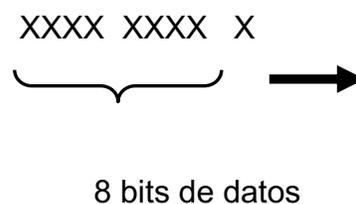
velocidad de *baudios* (bits por segundo) de transferencia de datos serie, el dispositivo transmisor debe transmitir *un bit de datos en cada pulso de reloj*, y, por lo tanto, el dispositivo receptor debe tener la capacidad de interpretar las señales de los datos serie.

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 o 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 (aquí podemos observar una aplicación de los bits de paridad, anteriormente revisados en el capítulo tres de estos apuntes):





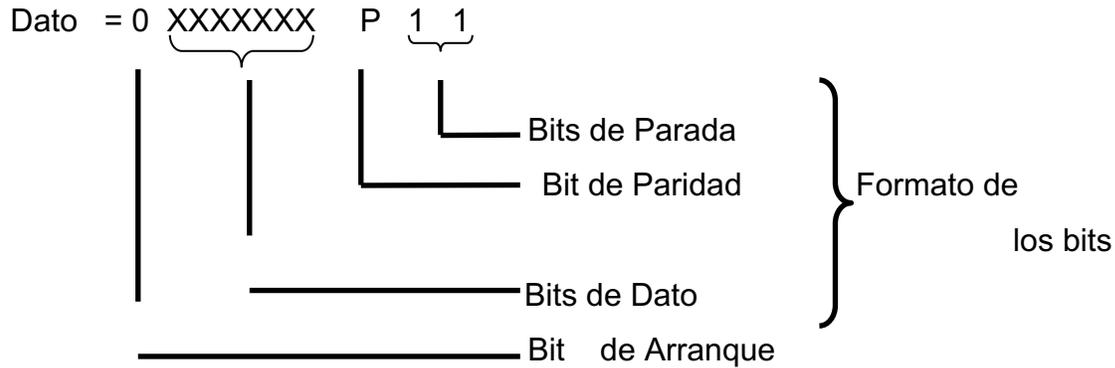
Un dispositivo receptor entra en el modo "*Hunt*" (cacería) mientras espera a que comiencen a llegar los Datos Serie Síncronos. Durante la espera el receptor lee continuamente los datos serie que recibe tratando de encontrar en el flujo de datos serie el patrón estándar SYNC. Si su protocolo se llama 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 asíncrono, 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 termino 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/Asynchronous 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.

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 (datos en bytes u octetos, que entran a la interfaz para su transmisión serie de salida; datos en serie que llegaron y se forman a partir de ellos octetos en paralelo para su transmisión a la computadora). 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.

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{CS} y $\overline{C/D}$ (Control/Dato). La señal \overline{CS} permite seleccionar al dispositivo y la señal $\overline{C/D}$ permite indicar el tipo de dato durante la transferencia. La dirección de la transferencia se indica al dispositivo por medio de las entradas \overline{RD} y \overline{WR} . La combinación de estas dos entradas con la entrada $\overline{C/D}$ 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.

Original	RS232 on board baud rate
50	50
300	300
600	600
1200	1200
2400	2400
4800	4800
9600	9600
19.2K	19.2K
38.4K	38.4K
57.6K	57.6K
115.2K	115.2K

Tabla 8.8 Velocidad de transmisión de un puerto USART de computadora, en bits/segundo (k= multiplicador 1000), observe la lentitud relativa en la transmisión de datos de estos dispositivos, comparados con la velocidad de transmisión de las redes de comunicación actuales.
Fuente: PCCOM PCI 4 PORT HYBRID CARD - CHAPTER 5. (s.f.). Recuperado el 28 noviembre, 2018, de: <https://www.pccompqi.com/manual/PCI-4PORT/3in1/chapter5.html>

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 o 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 o 2	No se Aplica
Caracteres SYNC	No se aplica	1, 2 o SYNC Externo

Tabla 8.9 Parámetros de E/S serie. Elaboración propia.

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 en paralelo de la CPU y los convierte en un flujo de datos en serie para transmisión. Simultáneamente puede recibir un flujo de datos en serie y convertirlos en caracteres de datos en paralelo para la CPU. Las terminales y señales de control del C.I. 8251 se muestran en las siguientes figuras:

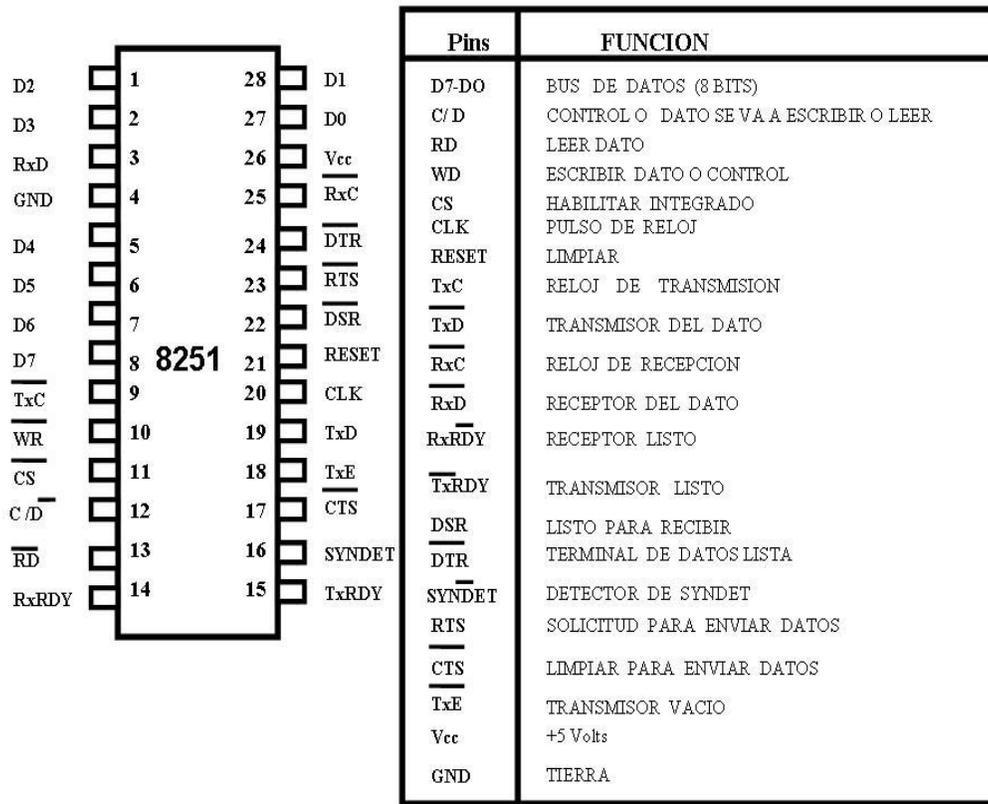


Figura 8.12 Configuración de CI 8251

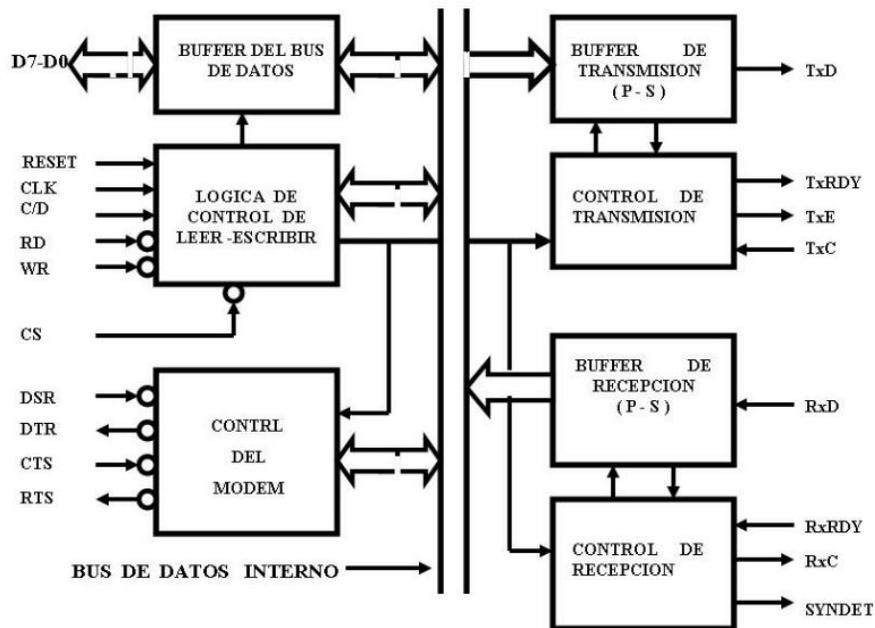


Figura 8.13 Configuración del CI 8251 (Continuación)

Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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* (dispositivo de comunicación vía línea telefónica fija)

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 considerarán primero las señales de control e interfaz con la CPU.

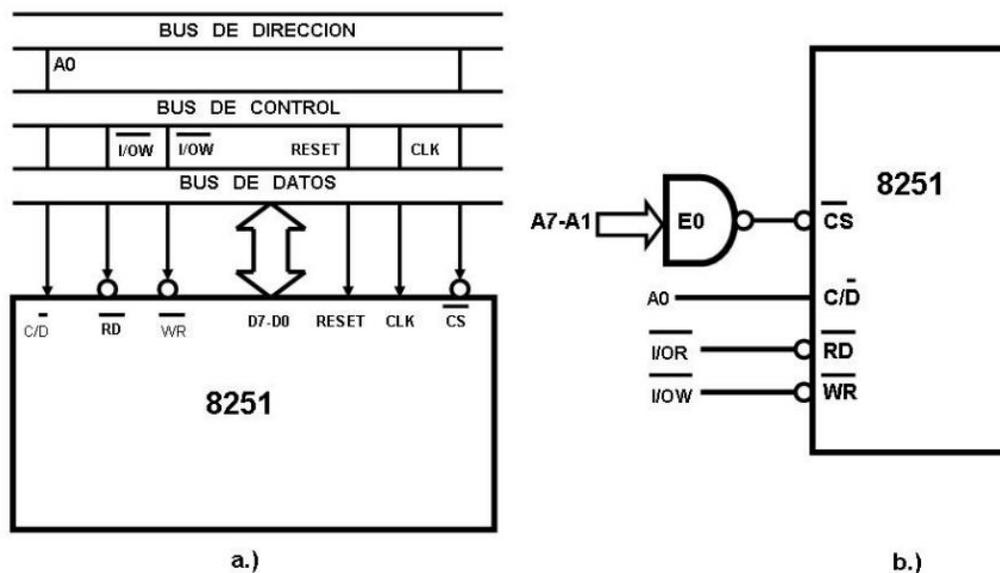


Figura 8.14 Interfaz del Puerto Programable Serial. a) Conexión del 8251 al Bus del Sistema 8085 . b) Señales de Control y Selección. Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.



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{CS} y $\overline{C/D}$

Estas señales se combinan con los valores de las entradas \overline{RD} y \overline{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{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}	$\overline{C/D}$	\overline{RD}	\overline{WR}	TRANSFERENCIA
0	0	0	1	8251 → Bus de Datos
0	0	1	0	Bus de Datos → 8251
0	1	0	1	Estado → Bus de Datos
0	1	1	0	Bus de Datos → Control
1	X	X	X	Bus de datos → Tercer Estado

Tabla 8.9. Selección del tipo de transferencia. Elaboración propia.



Cuando la señal $\overline{\text{I/OR}}$ tiene nivel bajo, la CPU está leyendo un dato o la palabra de estado; y cuando la señal $\overline{\text{I/OW}}$ 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, $\overline{\text{CS}}$ y C/D(=0). Mientras la CPU lee el contenido del buffer RA, el próximo dato serie se puede estar ensamblado en el buffer RB. Por "ensamblar" se entiende al hecho de recibir 8 bits en serie por medio de una línea y almacenarlos en un registro para tenerlos disponibles en paralelo.

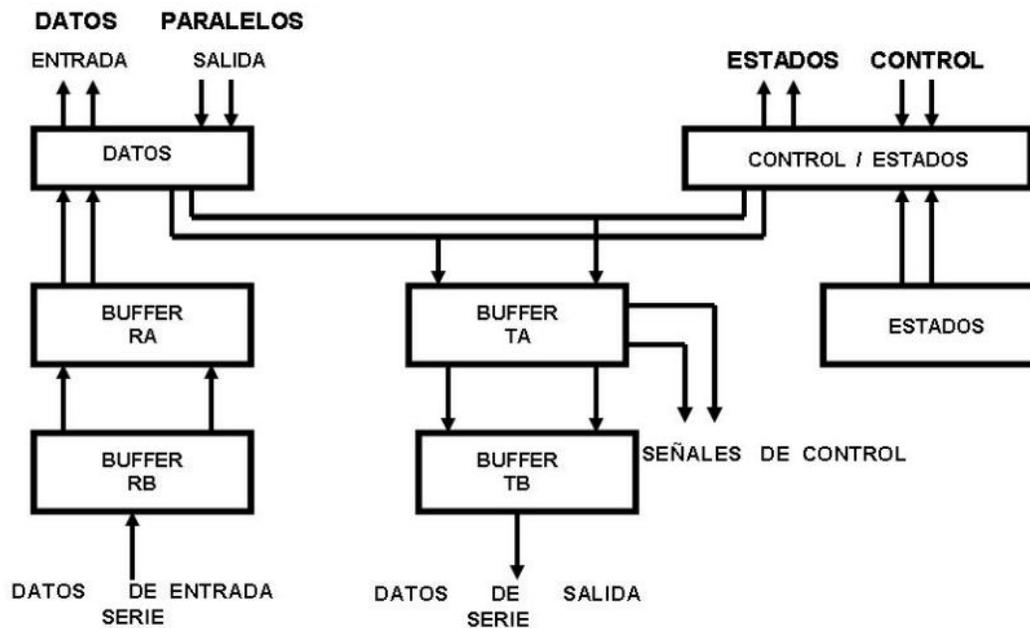


Figura 8.15. Diagrama del flujo de datos. Fuente: Intel Corporation. (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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* o desbordamiento).

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 vacío 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 $\overline{\text{CTS}}$ 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.

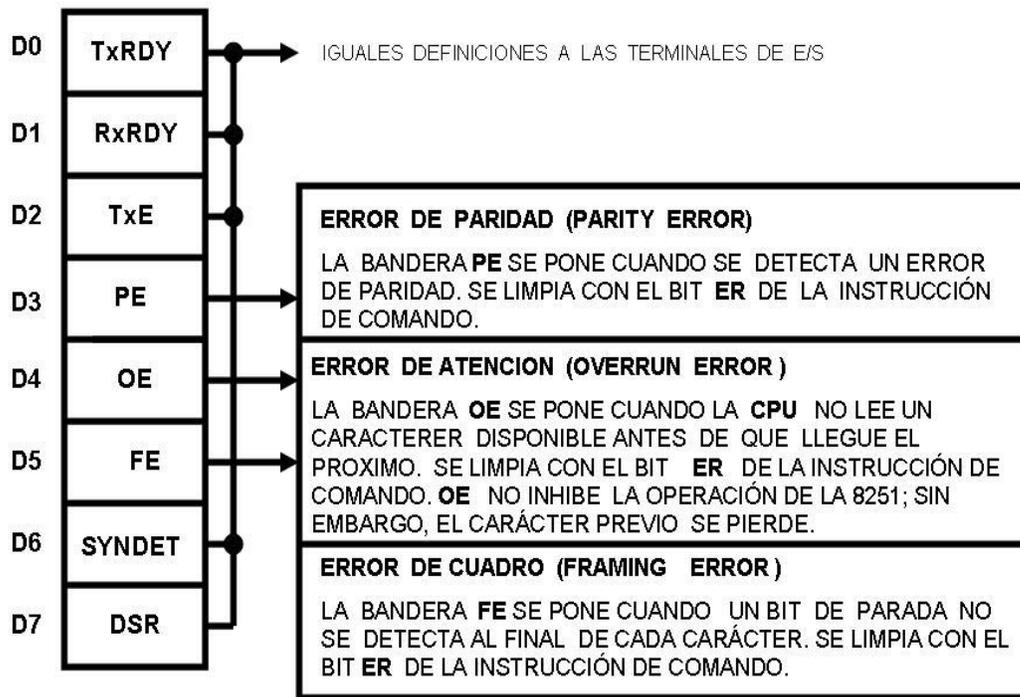


Figura 8.16 Formato para Lectura de los Estados

Fuente: Intel Corporation (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.

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.

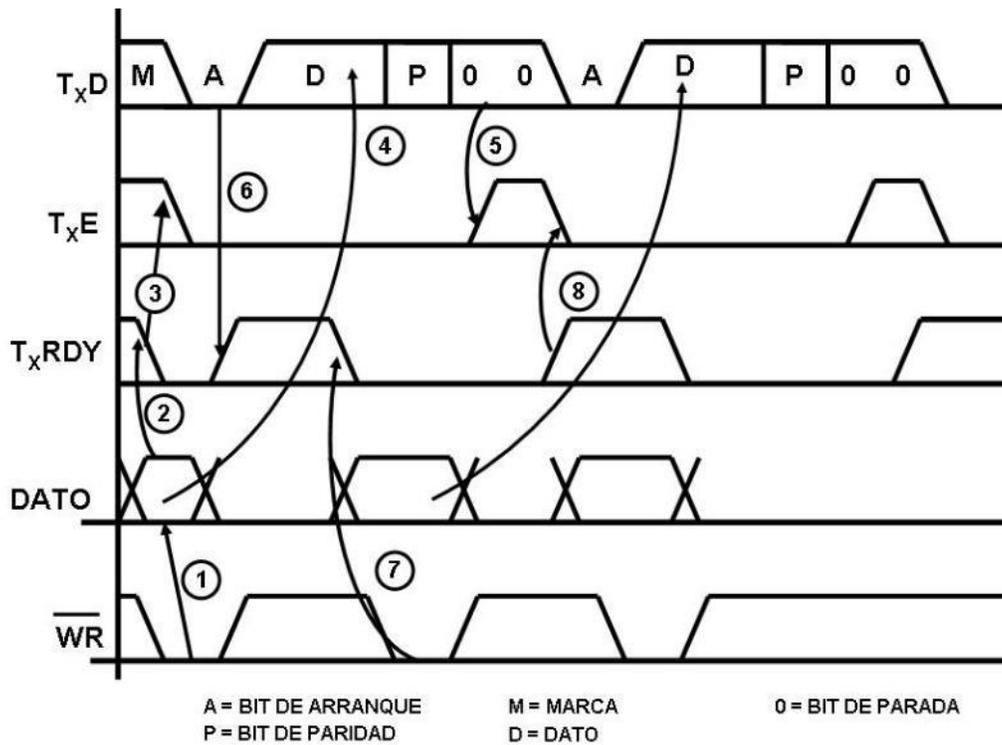


Figura 8.17. Diagrama de tiempos de transmisión asíncrona

Fuente: Intel Corporation (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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{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 $\overline{\text{WR}}$.
8. Cuando TxE pasa a nivel alto, TA se encuentra esperando un dato, una vez recibido dicho dato, se transfiere inmediatamente a TB, ocasionando que TxE pase a un nivel bajo y TxRDY a un nivel alto.
9. El dato nuevo es ahora enviado 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 $\overline{\text{RxD}}$. Los bits de los datos son muestreados por las señales de reloj $\overline{\text{RxC}}$, las cuales en forma semejante a TxC



usualmente se derivan del reloj del sistema de microcomputadora. Las transiciones bajo-alto del reloj \overline{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 que 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.

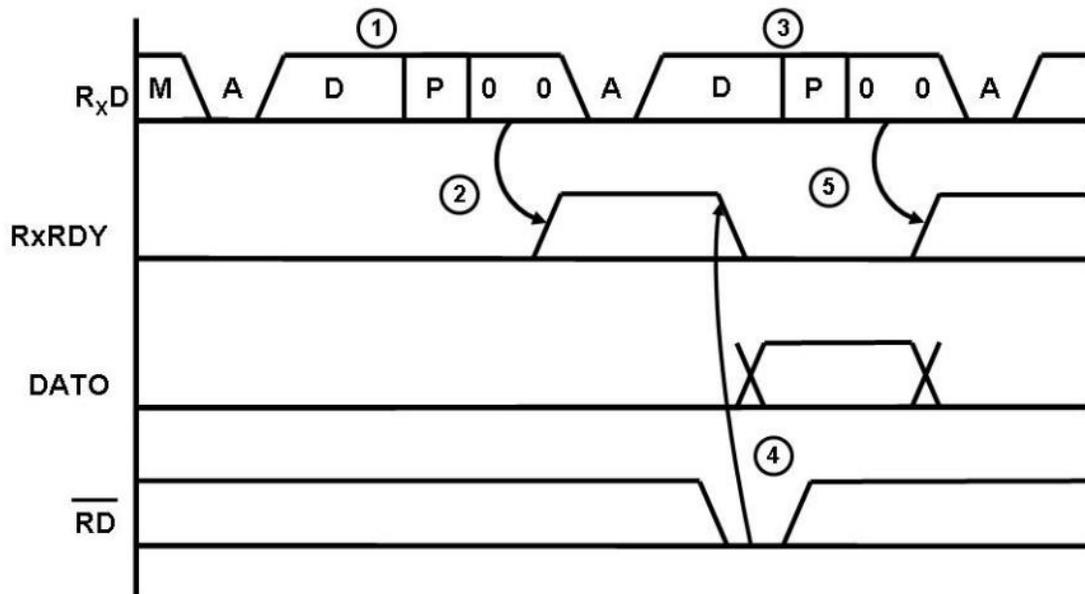


Figura 8.18 Diagrama de tiempos de recepción asíncrona

Fuente: Intel Corporation (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.

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 8.19 Formato de Instrucción de Comando

Fuente: Intel Corporation (1987). Hoja de especificaciones. Santa Clara California, Estados Unidos de América: Intel.

Control del *módem* (MODulador-DEModulador)



Las señales de control del módem son estándar para estos dispositivos que transmiten por líneas telefónicas fijas o dedicadas; para esto el *hardware* realiza una etapa de preparación para enviar los datos con las características requeridas por el canal de comunicación, también llamado proceso de **modulación**, en el lado contrario (receptor) los datos son bajados del canal y adaptados para su envío al equipo destino, o proceso de **demodulación**. El C.I. 8251 utiliza la salida $\overline{\text{DTR}}$ para indicar que se encuentra listo, y la entrada $\overline{\text{DSR}}$ 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 $\overline{\text{RTS}}$ y la entrada $\overline{\text{CTS}}$ 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 o velocidad 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:

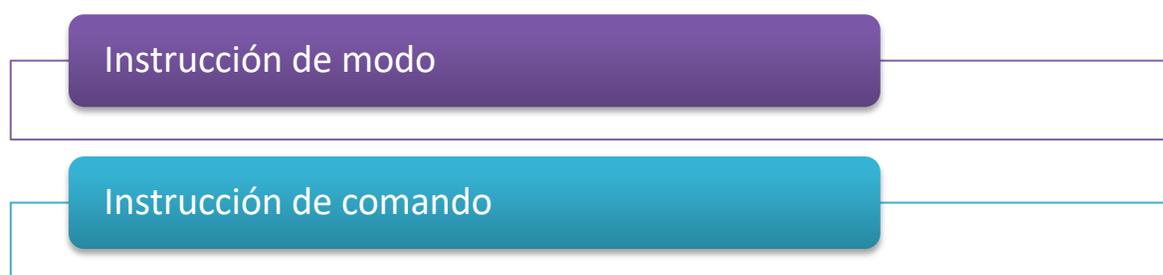


Figura 8.20 Formatos de palabras de control. Elaboración propia.

La siguiente figura Secuencia de la programación de las instrucciones ilustra la secuencia de la instrucción de Modo y de Comando.

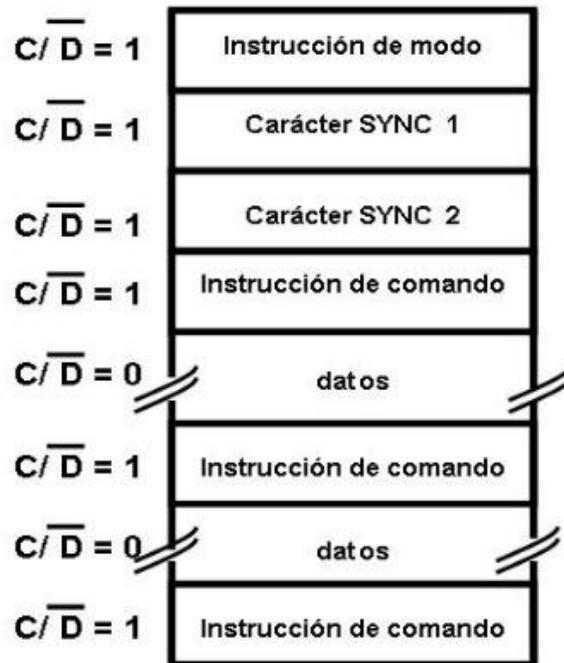


Figura 8.21 Secuencia de la programación de las instrucciones.

Fuente: Intel Corporation (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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, 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 el 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, o 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 $\overline{\text{TxC}}$ y $\overline{\text{RxC}}$.

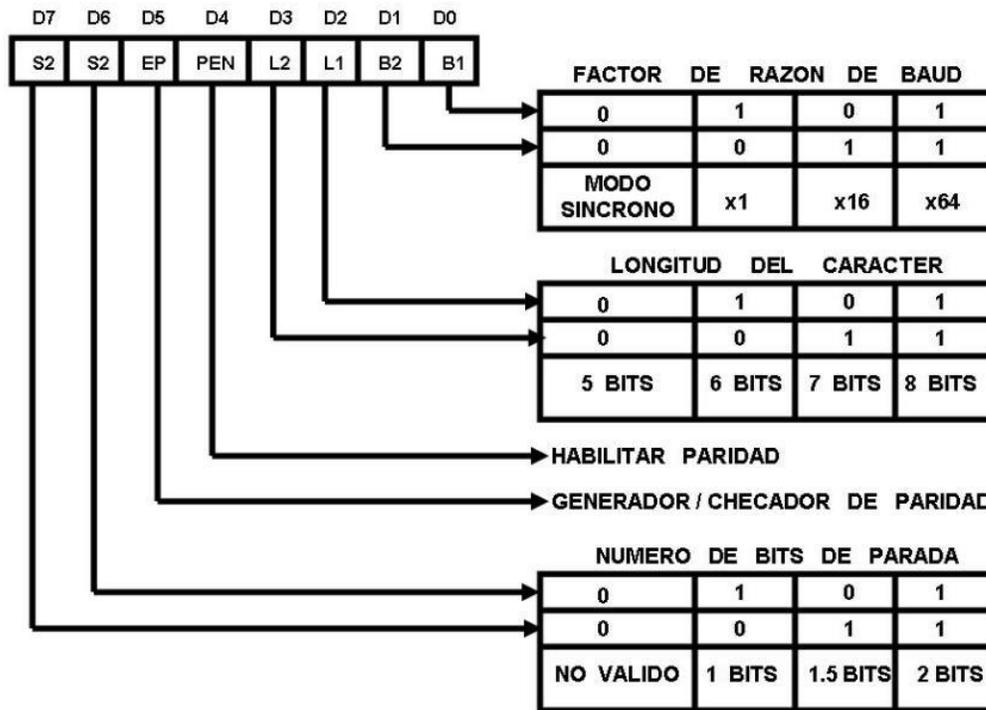


Figura 8.22 Formato de la Instrucción de Modo Asíncrono

Fuente: Intel Corporation (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

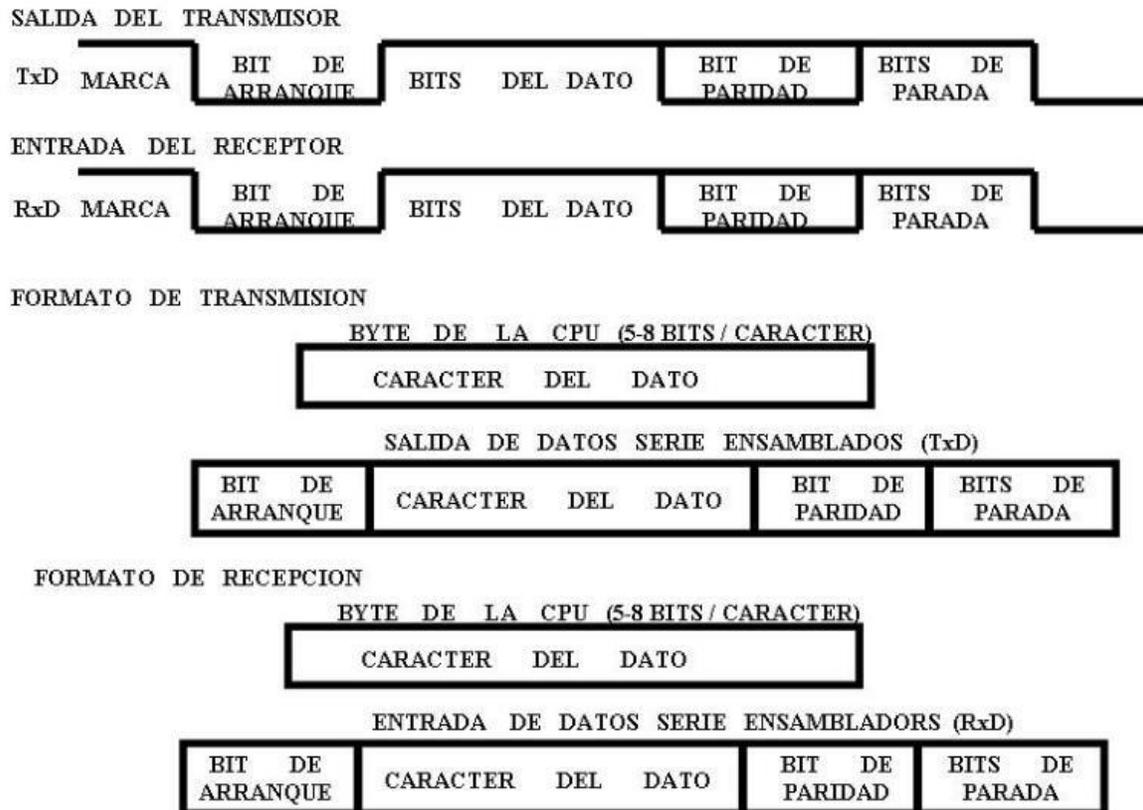
Por ejemplo, si la frecuencia de las entradas de reloj $\overline{\text{TxC}}$ y $\overline{\text{RxC}}$ es 38,400 y se escoge un factor de razón de baud de x64 (bits D0-D1=11), la frecuencia transmisión y recepción será de 600 bits por segundo.

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 o 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.



NOTA 1: SI LA LONGITUD DEL CARÁCTER SE DEFINE COMO DE 5, 6 ó 7. LOS BITS QUE NO SE USAN SE PONEN A CERO.

Figura 8.23 Modo Asíncrono

Fuente: Intel Corporation (1987). *Hoja de especificaciones*. Santa Clara California, Estados Unidos de América: Intel.

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 $C/\overline{D} = 1$ 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{DTR} a nivel 0.

La señal \overline{DTR} se utiliza en sistemas de comunicación de datos que operan automáticamente o bajo control de programa. La salida \overline{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:

Habilitado para transmitir, operación normal, enviar las señales $\overline{\text{DTR}}$ y $\overline{\text{DTS}}$ a 0 y no entrar en estado Hunt.

Solución:

$$\text{Control} = 00100111 = 27\text{H}$$

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 / \bar{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.

Valores para programar la velocidad de transmisión de la UART

Velocidad (BPS)	Divisor (Dec)	Divisor Latch Byte Alto (Programado en la dirección BASE+1 con DLAB=1)	Divisor Latch Byte bajo (Programado en la dirección BASE con DLAB=1)
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

Tabla 8.24 Velocidades de transmisión de la USART 8251.

Fuente: (s/a) (s/f) "Programación básica de la interface RS232 C" Recuperado el 8 de febrero de 2019 de: <https://slideplayer.es/slide/1758199/>



De la tabla podemos observar que la velocidad de transmisión de una USART es baja comparada con dispositivos de tipo USB y conexiones Ethernet por medios alámbrico o inalámbrico (WiFi), sin embargo, para comunicaciones en localidades remotas donde la infraestructura de red es limitada y solo se cuenta con líneas telefónicas, en procesos industriales de baja velocidad o en ambientes de ruido eléctrico, la comunicación vía un USART aún se encuentra en uso. Un elemento de interés es observar la complejidad asociada al proceso de transmisión en formato serie, no tan solo por la forma de separar los bytes en bits, sino la tarea de recuperar íntegra la información, ya sea de manera síncrona o asíncrona, las implicaciones con el protocolo de comunicación y la adaptación de la información para su transmisión a través del medio utilizado. También es de resaltar que los protocolos de comunicación actuales son cada vez más versátiles y complejos y que todas las funciones requeridas son llevadas a cabo por “pequeños” *chips*, pero que con su alto nivel de integración pueden desarrollar eficientemente los protocolos de comunicación de alta velocidad y con un alto nivel de confiabilidad, todo ello logrado a través del avance en el diseño de arquitecturas digitales, como las que hemos revisado en estos apuntes.



8.1.6. Sistemas de almacenamiento de información

En muchas ocasiones 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 unidades ópticas reescribibles, discos duros, etc. Actualmente existen dispositivos (de entrada/salida) para almacenar información que va desde casi un GigaByte (GB: 1, 073, 741, 824 bytes) hasta varios TeraBytes (TB: 1, 099, 511, 627, 776 bytes).

Una primera clasificación de los sistemas de almacenamiento se puede realizar en función de la tecnología utilizada para ello. Actualmente existen tres tipos de tecnologías: la óptica (CD, DVD, Bluray), la magnética (discos duros) y la de semiconductores (memorias tipo USB (Universal Serial Bus)), discos duros SSD (Solid State Disc: discos de estado sólido), sus diferencias estriban como lo indica su nombre, en el fenómeno físico que las soporta; pero también en su capacidad de almacenamiento (actualmente los medios de mayor capacidad son de tipo magnético); su velocidad (los líderes en este campo son los de semiconductores); su bajo consumo de energía en operación (los semiconductores son líderes); su tecnología de construcción electromecánica (tecnologías óptica y magnética) o sin partes móviles (semiconductores).



RESUMEN

En esta unidad se revisaron las principales partes que integran tanto el 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 DE LA UNIDAD



BIBLIOGRAFÍA SUGERIDA

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



REFERENCIAS BIBLIOGRÁFICAS

Bibliografía sugerida

Mano M. Morris, (2013). *Diseño Digital*. México: Pearson Education.

_____ (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

Bibliografía básica:

Baer, J. (2010). *Microprocessor architecture: from simple pipelines to chip
multiprocessors*. Cambridge: U.S.A: Cambridge University Press.

Beltrán, M., & Guzmán, A. (2010). *Diseño y evaluación de arquitecturas de
computadoras*. Madrid: Pearson.

Castro, M., Mora, C., Peire, F., Mur, F., et al. (2013). *Estructura y tecnología de
computadores I: (gestión y sistemas)*. Madrid: Universidad Nacional de Educación a
Distancia.

Lara, J. A. (2014). *Arquitectura de computadores y sistemas*. Madrid: Centro de
Estudios Financieros.

Quiroga, I. P. (2010). *Arquitectura de computadoras*. Buenos Aires; Ciudad de
México: Alfaomega /Grupo Editor Argentino.



Rocha, F. E., Martins, R. F., Lourenço, N. C., & Horta, N. G. (2014). *Electronic design automation of analog ICs combining gradient models with multi-objective evolutionary algorithms*. Cham: Springer.

Vázquez, J. B. (2012). *Arquitectura de computadoras I*. México: Red Tercer Milenio.

Bibliografía complementaria

Lee, R. B. (2013). *Security basics for computer architects*. San Rafael: Morgan & Claypool Publishers.

Luisi, J. V. (2014). *Pragmatic enterprise architecture: strategies to transform information systems in the era of big data*. Massachusetts: Morgan Kaufmann.

Molina, F. J. (2014). *Implantación de los elementos de la red local*. Madrid: Starbook.

Moreno, J. C. (2014). *Montaje de equipos microinformáticos*. Madrid: Starbook.

Xu, L. D. (2015). *Enterprise integration and information architecture: a systems perspective on industrial information integration*. Florida, USA: CRC Press.

Otras fuentes de consulta.

Altamira Aldo. (s.f.). "Arquitectura de computadoras". Recuperado 28 noviembre, 2018, de <http://compuarqui.blogspot.com/2015/02/interrupciones.html>

Fairchild Corp. (1981). *Mos Memory Data Book*. Mountain View California, Estados Unidos de América: Fairchild.

Gutiérrez Pedro (2013) ¿Cómo funciona la computadora actual?: Funcionamiento de la arquitectura de Von Neumann. Recuperado el 15 octubre de 2018, de: <https://www.genbetadev.com/actualidad/como-funciona-la-computacion-actual-funcionamiento-de-la-arquitectura-de-von-neumann>

IBM Corp. (1983). *IBM Technical Reference*. Boca Raton, Flo., Estados Unidos de América: IBM.



IBM Corp. (1985). *IBM Personal Computer hardware reference library /Technical Reference*. Boca Raton, Flo., Estados unidos de América: IBM.

IntelCorp. (1984). *Memory Component Handbook*. Santa Clara California, Estados Unidos de América: Intel.

Robert F. Coughlin y Frederick Driscoll. (1987). *Circuitos integrados lineales y amplificadores operacionales*. México: Prentice Hall.

Sitios electrónicos

Vigentes al 27 de febrero de 2019.

Sitio	Descripción
http://platea.pntic.mec.es/~lgonza/le/tic/binarios/aritmetica.html	Sitio que describe como se realiza la gramática binaria.
http://www.asifunción.com/informatica/af_binario/af_binario_5.htm	Página que muestra ejemplos de cómo funciona el sistema numérico binario.
https://www.researchgate.net/publication/287646222_Representación_de_numeros_en_binario	Sitio que presenta un documento que muestra la representación de números en binario.
https://ascii.cl/es/conversión.htm	Sitio que muestra la tabla de conversión Decimal, Hexadecimal, Octal y binario.
https://www.w3.org/International/articles/definitions-characters/index.es	Página que muestra la codificación de caracteres.
http://www.esacademic.com/dic.nsf/eswiki/278151	Sitio que muestra información acerca del código biquinario.
http://arquitectura.blogspot.com/2013/06/una-bestia-de-la-potencia.html	Sitio que presenta información sobre microprocesadores.



http://platea.pntic.mec.es/~lgonza/le/tic/binarios/logica.html	Página que informa sobre la lógica binaria.
http://www.kingston.com/es/usb/storage_chart	Página que muestra información sobre memorias y sus capacidades de almacenamiento
http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/Tflipflop.html	Página que muestra el circuito Flip-Flop T
https://manybutfinite.com/post/motherboard-chipsets-memory-map/	Página que muestra las características de una tarjeta madre.
http://reparacióndepc2016.blogspot.com/2016/05/memoria-cache-1-cual-es-el-objetivo-de.html	Página que muestra información sobre reparación de PC.
https://www.genbetadev.com/actualidad/como-funciona-la-computación-actual-funcionamiento-de-la-arquitectura-de-von-neumann	Página que muestra el funcionamiento de la computadora actual de acuerdo a la arquitectura de von Neumann
http://compuarqui.blogspot.com/2015/02/interrupciones.html	Página que muestra información sobre arquitectura de computadoras.
https://es.slideshare.net/mirandafector/el-microprocesador-8085	Presentación que muestra información sobre microprocesadores.
https://www.pccompci.com/manual/PCI-4PORT/3in1/chapter5.html	Página que muestra información sobre la configuración del Switch y Jumper.
https://slideplayer.es/slide/1758199/	Página que muestra las velocidades de transmisión del a USART 8251).

Plan 2012 **2016**
actualizado

