Instituto de Formación Técnica Superior N° 18

Mansilla 3643 (C1425BBW), Ciudad Autónoma de Buenos Aires

Técnico Superior en Análisis de Sistemas

Arquitectura de Computadoras

Asignatura:
Arquitectura de Computadoras

Autor:
Leandro E. Colombo Viña

última modificación:
27 de marzo de 2014

Resumen

Este apunte aborda los temas principales que profundizaremos a lo largo del cuatrimestre. Es una breve introducción a la materia.

En el capítulo 1 abordaremos los conceptos básicos que tenemos que trabajar durante la asignatura. Establece un punto de partida común de terminologia y conceptos que debemos empezar a familiarizarnos con.

En el capítulo 2 se trabajan los conceptos de cómo la computadora utiliza los números para la representación de información. En particular, este tema sera abordado con mayor profundidad en la materia Lógica Computacional.

En el capítulo 3 introducimos los conceptos de lógica digital que son básicos para el entendimiento del funcionamiento de una computadora. Este tema se verá en mayor profundidad en la materia Lógica Computacional y luego casi al final del cuatrimestre trabajaremos sobre los circuitos involucrados.

Finalmente, en el capítulo 4 introducimos los conceptos básicos de la arquitectura de von Neumann. En él veremos las principales unidades funcionales de una computadora y será nuestro punto de partida. A lo largo del cuatrimestre trabajaremos conceptualmente con estas unidades funcionales en mayor profundidad.

Índice general

Capítulo 1

Conceptos Básicos

1.1. La computadora en la vida diaria

En la vida moderna las computadoras constituyen un componente esencial y, aunque no lo notemos, están en todas partes y son determinantes en nuestro modo de vida. Aún más, a veces sólo nos damos cuenta de esto cuando dejan de funcionar.

Pensemos por un momento en qué cosas está presente alguna forma de computadora: en el reloj despertador digital, en la radio, la TV, en el reproductor de CD, en una agenda electrónica, en la cafetera automática, en el horno a microondas, en el encendido electrónico del auto, en el portón eléctrico de la cochera, en nuestro teléfono celular, en el cajero automático, en el lector de tarjeta de ingreso al trabajo, en los ascensores automáticos, en los controles de seguridad del edificio, en los lavarropas automáticos, en las cámaras fotográficas, en las máquinas de juegos, en las expendedoras de comida, en el control de los semáforos, en las centrales telefónicas, en los aviones, en los aeropuertos, en ... ¡casi todo!

Es difícil imaginarse un día en el cual no utilicemos alguno de estos elementos. ¿Qué pasaría si todos ellos dejaran de funcionar simultáneamente? Nuestra vida está relacionada con las computadoras, tanto por su operación como por su falta de funcionamiento. Y lo más sorprendente es que se hayan infiltrado tanto en la vida diaria en un tiempo tan corto.

1.1.1. La "idea" de la computadora

En 1823, el excéntrico genio matemático inglés Charles Babbage, profesor en Cambridge, comenzó a trabajar sobre la idea de un dispositivo mecánico para efectuar sumas repetidas. Esta idea se enriqueció al conocer que Jacquard, un fabricante de tejidos francés, había ideado un telar que permitía reproducir automáticamente patrones de tejidos leyendo la información codificada en patrones de agujeros perforados. Babbage se embarcó entonces en el ambicioso proyecto de crear una máquina analítica, que pretendía evolucionar el telar programable en una máquina capaz de realizar cualquier cálculo que se le programara mediante tarjetas perforadas, con una precisión de 20 dígitos.

A esta idea adhirió Ada Lovelace, hija del poeta Lord Byron y con aptitudes matemáticas. Publicó un artículo sobre la máquina analítica que incluía el

primer programa para computadora. Se asoció a Babbage aportando mayores alcances a su idea y corrigiendo errores de su trabajo.

> *"La máquina analítica no es capaz de crear nada, sin embargo puede hacer cualquier cosa que sepamos ordenarle."*
>
> **Ada Lovelace.**

Pero la tecnología de la época no bastaba para hacer realidad la máquina. El mundo aún no estaba listo para las computadoras, y no lo estaría por cien años más.

1.1.2. De la calculadora a la computadora... la gran diferencia

Si bien las computadoras nos acompañan desde hace apenas medio siglo, sus raíces van mucho más allá de la máquina analítica concebida por Babbage, y son producto de siglos de meditación y esfuerzo intelectual. Durante años el esfuerzo tecnológico estuvo en calcular: ábacos, calculadores mecánicos, circuitos electromecánicos, circuitos electrónicos. El objetivo era obtener la mayor velocidad posible para alguna combinación de las operaciones matemáticas básicas. Las primitivas computadoras y las primeras aplicaciones industriales fueron de cálculo fijo que debía hacerse a la mayor velocidad posible. Los componentes electrónicos más "famosos" eran las Unidades Aritmético-Lógicas que realizaban cálculos simples a gran velocidad.

El salto conceptual de las "máquinas de calcular" a la computadora fue comprender que el cálculo era sólo uno de los elementos de interés para la computación. Aún más, representaba tal vez la línea tecnológica más "fácil". El verdadero desarrollo estaba en poder generalizar la utilización de "la máquina" para cualquier aplicación que se pudiera "programar", tal como lo había escrito Ada Lovelace 120 años antes.

1.1.3. Una primera definición

Una **computadora** es una máquina digital y sincrónica, con cierta capacidad de cálculo numérico y lógico, controlada por un programa almacenado, y con posibilidad de comunicación con el mundo exterior. ¿Qué significa esto?

- Es digital porque dentro de la computadora las señales eléctricas que se manejan y la información que se procesa se representa en forma discreta, por medio de valores binarios (0 y 1).
- Además es sincrónica, ya que realiza las operaciones coordinada por un reloj central que envía pulsos de sincronismo a todos los elementos que componen la computadora. Esto significa que todas las operaciones internas se realizan en instantes de tiempo predefinidos y coordinados con el reloj.
- Internamente posee una capacidad de cálculo numérico y lógico, en un subsistema conocido como unidad aritmético-lógica (ALU, por sus siglas en inglés). Normalmente las operaciones que pueden realizarse en ella son muy simples (suma, disyunción, conjunción, comparaciones).

- El hecho de que sea controlada por programa es quizás el punto más importante que diferencia a una computadora de una calculadora. Significa que internamente se tienen órdenes o instrucciones almacenadas, que la computadora podrá leer, interpretar y ejecutar ordenadamente.
- Además, está comunicada con el mundo real, que es analógico. Esto significa que puede realizar operaciones de entrada y salida con el mundo real, a través de dispositivos periféricos (por ejemplo el teclado o el mouse para la entrada de información, y la pantalla para la salida).

La computadora es una máquina que cambia información de una forma a otra: recibe información (entrada), la transforma, y proporciona información (salida). Esta información puede presentarse de muchas formas, lo que convierte a la computadora en una máquina sumamente versátil, que es capaz desde liquidar impuestos hasta guiar el recorrido de una nave espacial. En cada caso las entradas y salidas son totalmente distintas, y en esto radica lo sorprendente de poder usar una computadora para ambas actividades.

Esta versatilidad está dada en que la máquina está controlada por un programa, que establece las instrucciones que le indican a las partes físicas qué deben hacer para transformar los datos de entrada en la salida requerida. El programa controla todo el proceso, de principio a fin: podemos modificar su funcionamiento con solo cambiar el programa. Con el advenimiento de la computadora, gran parte de la tecnología pasó del mundo analógico al digital.

1.2. Un poco de historia

"Considera el pasado y conocerás el futuro"

Proverbio Chino

La evolución en la tecnología electrónica en los últimos 60 años tuvo un impacto notable en la ciencia informática. En la primera generación de computadoras, las máquinas estaban construidas con *tubos de vacío* (válvulas), que eran tubos de vidrio del tamaño de una bombilla que albergaban circuitos eléctricos. Eran máquinas muy grandes, costosas y de difícil operación. A pesar de esto, rápidamente se convirtieron en herramientas indispensables para los científicos e ingenieros.

El *transistor*, inventado en 1948, podía cumplir la misma función que un tubo de vacío, ya que podía transferir la electricidad a través de una pequeña resistencia. Esto dio lugar, a partir de 1956, a la segunda generación de computadoras, donde las máquinas ya eran más pequeñas, confiables y económicas que las anteriores. En forma paralela hubo un avance en la programación y la forma de manejo de estas computadoras, lo que produjo un mayor uso de las mismas.

A mediados de los ´60 las computadoras basadas en transistores fueron sustituidas por las máquinas más pequeñas y potentes de la tercera generación, construidas con base en los nuevos circuitos integrados (que empaquetaban cientos de transistores en una pequeña placa de silicio). Su éxito estuvo basado en la mayor confiabilidad, velocidad y eficiencia, y menor tamaño y costo.

La invención del tubo de vacío, el transistor y los circuitos integrados (IC, del inglés, *Integrated Circuits*) tuvieron un impacto notable en la sociedad, y por

eso muchos historiadores señalan estos acontecimientos como fronteras generacionales. Pero ninguno de ellos tuvo un efecto más profundo que la invención en 1969 del primer microprocesador, que es una computadora completa empaquetada en un diminuto IC de silicio. Esto fue considerado el inicio de la cuarta generación, que trajo aparejados cambios en la capacidad y la disponibilidad de las máquinas en todo el planeta.

Datos (y velocidad) de la evolución:

- En el siglo IX un texto budista es el primer libro impreso conocido.
- En el siglo XV aparece la imprenta de Gutenberg.
- En el siglo XVIII aparece la revolución industrial.
- A principios del siglo XX la producción industrial automatizada.
- En el siglo XIX la radio.
- En el siglo XX la TV y el cine.

    - **1940 a 1950:** Aparecen las primeras computadoras. Con programa fijo y programa variable. En 1945 John von Neumann propone almacenar programas en forma de datos. Surge el transistor y con él la electrónica moderna.
    - **1950 a 1960:** Computadoras transistorizadas. Banca computarizada. Circuitos integrados. Láser. En 1959 la Unión Soviética lanza el Sputnik.
    - **1960 a 1970:** Sistemas operativos de tiempo compartido. El software como producto industrial. Lenguajes de programación. La primera red de computadoras. En 1969 el hombre llega a la Luna.
    - **1970 a 1980:** Aparecen los microprocesadores. Microcomputadoras. Robots industriales controlados por computadora. Supercomputadoras. Primeros videojuegos. Planilla de Cálculo. Interfaz gráfica. Apple. En 1979 nace el PacMan.
    - **1980 a 1990:** IBM presenta la primera computadora personal (PC). Surgen publicaciones electrónicas. Nace Internet. Aparecen las primeras computadoras masivamente paralelas. Aparecen los virus y los hackers.
    - **1990 a 2000:** En 1990 Microsoft introduce Windows 3.0. Aparecen otros elementos como la interfaz hablada, multimedia, robots móviles, realidad virtual, videoconferencia, visión por computadora, etc.
    - **2000 en adelante:** Adquiere fuerte impulso la Inteligencia Artificial. La realidad virtual cada vez es más real. La interfaz hombre-máquina sigue evolucionando. Las comunicaciones por Internet dan origen a nuevos mecanismos como el *e-commerce*.

Estos datos reflejan la diferencia en la velocidad de evolución de la informática con respecto a cualquiera de las otras industrias. Notar que el avance desde el primer libro impreso a la imprenta tomó 6 siglos, mientras que desde los tubos de vacío al primer microprocesador sólo pasaron una veintena de años.

El complejo electrónico-informático ha desplazado a la industria automotriz, a la industria pesada, a la industria militar y a la industria petrolera en la facturación mundial.

1.2.1. Los grande ejes de la evolución

"La experiencia histórica muestra que los cambios tecnológicos transforman notablemente las relaciones políticas y sociales"

John von Neumann

Podemos ver gráficamente cuáles han sido los grandes ejes de la impresionante evolución de las computadoras:

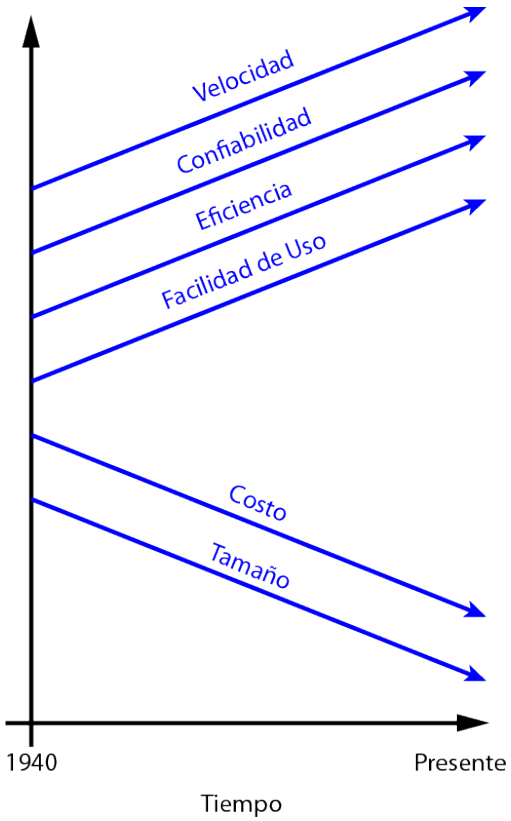

Figura 1.1: Evolución de las computadoras desde 1940 a la fecha

1.3. ¿Qué es la Informática?

La informática nace de la idea de ayudar al hombre en aquellos trabajos rutinarios y repetitivos, generalmente de cálculo y gestión, donde es frecuente la repetición de tareas. La idea es que una máquina puede realizarlos mejor, aunque siempre bajo la supervisión del hombre.

El término Informática se creó en Francia en 1962 bajo la denominación *Informatique*, y procede de la contracción de las palabras *Information* y *automatique*. Posteriormente fue reconocido por el resto de los países, siendo adoptado por España en 1968 bajo el nombre de **Informática**, que como puede deducirse fácilmente, viene de la contracción de las palabras *información* y *automática*. En los países anglosajones se conoce con el nombre de *Computer Science*.

La informática se puede definir de diversas formas si bien todas ellas giran en torno a la misma idea. Dos de las más difundidas son:

- es la ciencia que estudia el tratamiento automático y racional de la información.
- es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.

La palabra **ciencia** se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas.

La **resolución de problemas** utilizando las herramientas informáticas puede tener aplicaciones en áreas muy diferentes tales como biología, comercio, control industrial, administración, robótica, educación, arquitectura, diseño, etc.

Los temas propios de la ciencia Informática abarcan aspectos tales como la arquitectura física y lógica de las computadoras, las metodologías de análisis y diseño de sistemas de software, los lenguajes de programación, los sistemas operativos, la inteligencia artificial, los sistemas de tiempo real, el diseño y aplicación de bases de datos, etc.

1.3.1. Aplicaciones de la Informática

"El grado de inteligencia que atribuimos al comportamiento de algo está determinado tanto por nuestra propia capacidad y comprensión como por las propiedades del objeto que analizamos".

Alan Turing

"La computadora es, por mucho, la más extraordinaria de las vestimentas electrónicas creadas por el hombre, ya que es una extensión de nuestro sistema nervioso central. Junto a ella, la rueda no es más que un juguete..."

Marshall McLuhan

El universo de las aplicaciones informáticas es esencialmente multidisciplinario. Las aplicaciones que pueden desarrollarse con una computadora van desde un sistema de gestión comercial, administrativo, hasta sistemas expertos que ayudan en la toma de decisiones, diseño asistido, controladores de vuelo automáticos, máquinas jugadoras de ajedrez, etc.

En esta tarea están involucradas personas de distintas disciplinas: matemáticos, ingenieros e informáticos. Los matemáticos brindan las herramientas básicas para que tanto ingenieros como informáticos puedan desarrollar su labor.

Por otro lado se encuentran los usuarios de las aplicaciones, que van desde especialistas que utilizan una determinada herramienta (economistas, docentes, músicos, médicos, arquitectos, etc.) hasta entusiastas que navegan por Internet o juegan con un simulador de vuelo.

1.4. Componentes y funcionamiento básico de una computadora

Recordemos la definición que dimos antes de computadora: *"Una **computadora** es una máquina digital y sincrónica, con cierta capacidad de cálculo*

numérico y lógico, controlada por un programa almacenado, y con posibilidad de comunicación con el mundo exterior."

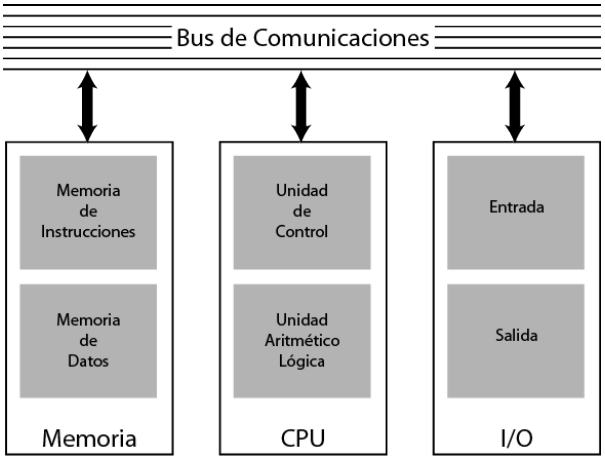

Figura 1.2: Esquema de bloques de una computadora

La mayoría de las computadoras actuales de propósito general presentan una estructura interna basada en la arquitectura definida por John von Neumann. Podemos esquematizarla de la siguiente manera:

- En el gráfico se ha dividido conceptualmente la **memoria**, en memoria de instrucciones donde residen las órdenes que la computadora debe interpretar y ejecutar, y en memoria de datos donde se almacena la información con la cual la computadora realizará los procesos (cálculos, decisiones, actualizaciones) que sean necesarios para la resolución del problema.

- El bloque rotulado como **I/O** representa los dispositivos que permiten la comunicación con el mundo real. Por ejemplo, el controlador de video que vincula el procesador central de la computadora con la pantalla o el circuito controlador de multimedia que permite tener salida por un parlante o entrada por un micrófono.

- Las líneas de comunicación indicadas como **bus de comunicaciones** normalmente permiten el paso de tres grandes categorías de información: direcciones, datos y control. En el esquema simplificado se acepta que estas líneas permiten la comunicación interna y externa de datos, direcciones y señales de control.

- Por último, tradicionalmente la combinación de la unidad de control (UC) y la unidad de cálculo (ALU) se la llama **unidad central de procesamiento** (CPU), que en las computadoras personales está representada por el microprocesador (por ejemplo: 486, 586, Pentium, Athlon, Sempron, etc.).

El funcionamiento de una computadora representado esquemáticamente por el modelo anterior, se puede sintetizar en el siguiente diagrama de flujo:

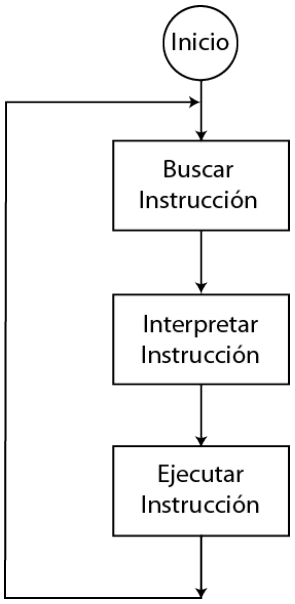

Figura 1.3: Funcionamiento del CPU

Esto representa una secuencia infinita de pasos:

- Buscar la próxima instrucción a ejecutar I_i de la memoria de instrucciones M_i.
- Interpretar qué hacer con I_i en la Unidad de Control (UC).
- Ejecutar las operaciones interpretadas por la UC, utilizando la ALU de ser necesario. Estas operaciones pueden comprender lectura/escritura de la memoria de datos M_d o entrada/salida por los periféricos P_e o P_s.

En clases posteriores trataremos más en detalle sobre la estructura interna y el funcionamiento de las computadoras. Para finalizar, damos algunos conceptos:

- El *hardware* se refiere a las componentes físicas de la computadora.
- El *software* comprende los programas y la información que se ejecutan sobre la computadora.
- Un bit (dígito binario o *binary digit*) es la unidad de información más pequeña. Sólo puede tener uno de dos valores: encendido o apagado (0 o 1, si o no, blanco o negro, etc.).
- La Unidad Central de Procesamiento (CPU) es la encargada de interpretar y llevar a cabo las instrucciones de los programas. Efectúa cálculos aritméticos y lógicos con los datos, y se comunica con las demás partes del sistema de cómputo.

Capítulo 2

Representación numérica.

2.1. Introducción

Desde hace mucho tiempo, el hombre en su vida diaria se expresa, comunica, almacena información, la manipula, mediante letras y números. Para la representación numérica utiliza el sistema de representación decimal, en tanto que, dependiendo del idioma, dispone de un alfabeto que representa estas letras. Siguiendo el mismo principio que guía al hombre, las computadoras tienen su propio sistema de representación. Debido a su construcción basada fundamentalmente en circuitos electrónicos digitales, utiliza un sistema binario. Esto obliga a transformar la representación de nuestra información, tanto numérica como alfanumérica, a una representación binaria para que la máquina sea capaz de procesarlos.

Como veremos más adelante, tanto el sistema decimal como el binario están basados en los mismos principios. En ambos, la representación de un número se efectúa por medio de cadenas de símbolos, los cuales representan una determinada cantidad dependiendo de cada símbolo y la posición que ocupa dentro de la cadena con respecto al denominado punto (o coma) decimal.

Por cuestiones de índole técnica, los circuitos electrónicos que conforman una computadora suelen estar capacitados para reconocer señales eléctricas de tipo digital; por lo tanto, se hace necesario que los métodos de codificación internos tengan su origen en el sistema binario, y con ellos se pueda representar todo tipo de informaciones y órdenes que sean manejadas por la computadora.

En los circuitos electrónicos suele representarse la presencia de tensión (electricidad) en un punto de un circuito por medio de un 1, en tanto que un 0 representa la ausencia de dicha tensión.

2.2. Sistemas de numeración

Se denomina sistema de numeración al conjunto de símbolos y reglas que se utilizan para la representación de datos numéricos o cantidades. Un sistema de numeración se caracteriza fundamentalmente por su base, que es el número de símbolos distintos que utiliza, y además es el coeficiente que determina cuál es el valor de cada símbolo dependiendo de la posición que ocupe.

Los sistemas de numeración actuales son sistemas posicionales, en los que el valor relativo que representa cada símbolo o cifra de una determinada cantidad depende de su valor absoluto y de la posición relativa que ocupa dicha cifra con respecto a la coma decimal.

2.2.1. Teorema fundamental de la numeración

Se trata de un teorema que relaciona una cantidad expresada en cualquier sistema de numeración posicional con la misma cantidad expresada en el sistema decimal. Supongamos una cantidad expresada en un sistema cuya base es B y representamos por x_i cada uno de los dígitos que contiene dicha cantidad, donde el subíndice i indica la posición del dígito con respecto a la coma fraccionaria, la posición se numera en forma creciente hacia la izquierda y decreciente hacia la derecha de la coma (posición 0), en ambos casos de a 1.

El Teorema Fundamental de la Numeración dice que el valor decimal de una cantidad expresada en otro sistema de numeración, está dado por la fórmula:

$$Número = \sum_{i=-m}^{n} (dígito_i) \times (base)^i \tag{2.1}$$

donde el número en base B es $\ldots x_4 x_3 x_2 x_1 x_0 x_{-1} x_{-2} \ldots$, o sea:

$$\begin{aligned} Número = {} & x_{-m} \times B^{-m} + x_{-m+1} \times B^{-m+1} + \cdots + \\ & + x_{-2} \times B^{-2} + x_{-1} \times B^{-1} + x_0 \times B^0 + x_1 \times B^1 + x_2 \times B^2 + \cdots + \\ & + x_{n-1} \times B^{n-1} + x_n \times B^n 1 \end{aligned} \tag{2.2}$$

2.2.2. Sistemas decimal, binario y hexadecimal

El sistema que ha usado el hombre para contar desde hace bastante tiempo es el denominado sistema decimal, adoptado por contar con los diez dedos de la mano. El sistema decimal es uno de los denominados posicionales, que utiliza un conjunto de 10 símbolos, $x_i \epsilon 0, 1, ..., 9$. Un valor determinado o cantidad, que se denomina número decimal, se puede expresar por la fórmula del Teorema 2.1, donde la base es 10.

¿Cuál es la interpretación de la representación de la cantidad 3,1416?

Siguiendo el Teorema Fundamental de la Numeración, utilizando la base 10, resulta:

$$3,1416 = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 6 \times 10^{-4} \tag{2.3}$$

El sistema binario es el sistema de numeración que utiliza internamente el hardware de las computadoras actuales. La base o número de símbolos que utiliza el sistema binario es 2, siendo los símbolos 0 y 1, los utilizados para la representación de cantidades.

¿Qué número decimal representa el número binario 1001,1?

Utilizando el Teorema Fundamental de la Numeración:

$$1001,1_{(2} = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} \tag{2.4}$$

Al igual que los anteriores, el sistema hexadecimal es un sistema posicional pero que utiliza dieciséis símbolos para la representación de cantidades. Estos símbolos son los siguientes: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E$ y F; donde las letras A, B, C, D, E y F equivalen a $10, 11, 12, 13, 14$ y 15 del sitema decimal respectivamente.

¿Qué número decimal representa el número hexadecimal 2CA?

Siguiendo el Teorema Fundamental de la Numeración:

$$2CA_{(16} = 2 \times 16^2 + C \times 16^1 + A \times 16^0 \tag{2.5}$$

$$2CA_{(16} = 2 \times 16^2 + 12 \times 16^1 + 10 \times 16^0 \tag{2.6}$$

$$2CA_{(16} = 512 + 192 + 10 = 714 \tag{2.7}$$

2.2.3. Operaciones de Suma y Resta Binaria

Las operaciones aritméticas son similares a las del sistema decimal, con la diferencia que se manejan sólo los dígitos 0 y 1. Al realizar la suma parcial de dos dígitos, si el resultado excede el valor del máximo dígito (el 1) se debe pasar el sobrante (denominado acarreo) a la suma parcial siguiente hacia la izquierda.

Sumemos los números binarios 100100 y 10110

				1			$\longrightarrow$ carry
	1	0	0	1	0	0	
+		1	0	1	1	0	
	1	1	1	0	1	0	

En la resta binaria hay que tener en cuenta que al realizar las restas parciales entre dos dígitos de idénticas posiciones, uno del minuendo y otro del sustraendo, si el segundo excede al primero, se sustrae una unidad del dígito de más a la izquierda en el minuendo –*pedir prestado*-. Si el dígito siguiente de la izquierda es 0, se busca en los sucesivos teniendo en cuenta que su valor se multiplica por dos a cada desplazamiento sucesivo a derecha.

Restemos los números binarios 111100 y 101010

					10		$\longrightarrow$ borrow
	1	1	1	$\not{1}$	$\not{0}$	0	
-	1	0	1	0	1	0	
	0	1	0	0	1	0	

2.2.4. Conversiones entre los sistemas de numeración

Se denomina conversión entre números representados en distintos sistemas de numeración a la transformación de una determinada cantidad expresada en uno de dichos sistemas de numeración, a su representación equivalente en el otro sistema.

Conversión decimal-binario

El método de conversión de un número decimal a un número binario consiste en efectuar, sobre la parte entera del número decimal, divisiones sucesivas de los cocientes por el número 2, hasta que el cociente entre una de las divisiones tome el valor 0. La unión de todos los restos obtenidos, escritos en orden inverso, nos proporciona el número inicial expresado en sistema binario.

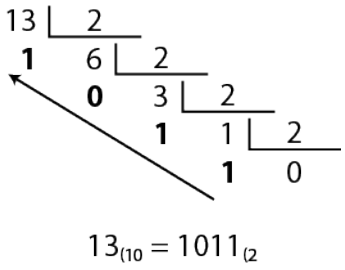

Figura 2.1: Convirtiendo el número decimal 15 a binario

Para convertir una fracción decimal a su equivalente binario se debe multiplicar dicha fracción por dos, obteniendo en la parte entera del resultado el primero de los dígitos binarios de la fracción que buscamos. A continuación, se repite el proceso con la parte fraccionaria del resultado anterior, obteniendo en la parte entera del nuevo resultado el segundo de los dígitos buscados. El proceso se repite hasta que desaparezca la parte fraccionaria de los resultados parciales (se haga 0) o hasta que tengamos los suficientes dígitos binarios.

$$\begin{aligned} 0,828125 \quad &\times 2 = \quad 1,65625 \\ 0,65625 \quad &\times 2 = \quad 1,3125 \\ 0,3125 \quad &\times 2 = \quad 0,625 \\ 0,625 \quad &\times 2 = \quad 1,25 \\ 0,25 \quad &\times 2 = \quad 0,5 \\ 0,5 \quad &\times 2 = \quad 1 \end{aligned}$$

$$0,828125_{(10} = 0,110101_{(2}$$

Figura 2.2: Convirtiendo el número decimal 0,828125 a binario

Conversión hexadecimal-binario y viceversa

Para convertir un número hexadecimal a binario se sustituye cada dígito hexadecimal por su representación binaria con cuatro dígitos según el Cuadro 2.1.

Hexadecimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Cuadro 2.1: Tabla de conversión Hexadecimal-Binario

En tanto que la conversión binaria hexadecimal se realiza un proceso inverso. Se agrupan los dígitos binario de a 4 a partir del punto decimal hacia la izquierda y hacia la derecha, sustituyendo cada cuarteto por su correspondiente dígito hexadecimal.

Convertimos el número hexadecimal 7BA3,BC a binario

7	B	A	3	,	B	C
0111	1011	1010	0111	,	1011	1100

Entonces luego de la conversión, escribimos el número. Omitimos los 0 no significativos: $7BA3,BC_{(16} = 111101110100111,101111_{(2}$

Convertimos el número binario 110010100100,1011011 a hexadecimal

Completamos con 0 para obtener los cuartetos necesarios para la conversión. Realizamos la conversión inmediata utilizando el cuadro 2.1.

0001	1001	0100	1000	,	1011	0110
1	9	4	8	,	B	6

Finalmente resulta: $1100101001000,1011011_{(2} = 1948,B6_{(16}$

Conversión de cualquier base a decimal

Para ello se utiliza el teorema fundamental de la numeración y se convierte el número de la base que se disponga a la decimal.

2.3. Representación de números enteros

Las computadoras utilizan cuatro métodos para la representación interna de números enteros (positivos y negativos):

- Módulo y signo
- Complemento a 1
- Complemento a 2
- Exceso a 2^{n-1}

Estas representaciones de números utilizan el sistema binario y siempre se considera que tenemos un número limitado de bits para cada dato. Este número de bits disponibles lo representamos generalmente con la letra n. También se pueden representar mediante estos métodos números reales, como veremos más adelante.

Al tener una cantidad limitada de bits para representar, vamos a estar limitados en la cantidad de números que podemos representar. Se denomina **rango de representación** en un método determinado al conjunto de número representables en el mismo.

2.3.1. Módulo y signo

En este sistema de representación, el bit que está situado más a la izquierda (bit más significativo, *MSB, most significant bit* en inglés) representa el signo, y su valor será 0 para el signo positivo (+) y 1 para el signo negativo (−). Los bits restantes $(n-1)$ representan el módulo del número. Suponemos en principio que los números no poseen parte decimal, por lo que la coma se supone implícita a la derecha.

Por ejemplo, supongamos que disponemos de $n = 8$ bits, y queremos representar los números 10 y −10. Veamos cuales son sus representaciones.

0	0	0	0	1	0	1	0
+	módulo = 10						

Figura 2.3: Representación del número 10.

1	0	0	0	1	0	1	0
-	módulo = 10						

Figura 2.4: Representación del número -10.

Rango

Para módulo y signo el rango de representación de una palabra de n bits está determinado por:

$$-2^{n-1} + 1 \leq Rango \leq 2^{n-1} - 1 \tag{2.8}$$

En el caso de $n = 8$ bits, el rango de representación va desde -127 a 127. La ventaja que presenta este sistema frente a otros es la de poseer un *rango simétrico* (igual cantidad de números positivos que negativos). Mientras que su mayor inconveniente es el de poseer *dos representaciones para el número* 0. El cual se representa tanto con un signo positivo (0) como con uno negativo (1) y el resto de los bits en 0.

2.3.2. Complemento a 1

En este sistema de representación también el bit de más a la izquierda se interpreta como el signo, correspondiendo el 0 para el signo positivo (+) y el 1 para el signo negativo (−). Para los números positivos, los $n - 1$ bits de la derecha representan el módulo (igual que en el sistema anterior). El negativo de un número positivo se obtiene complementando todos sus dígitos (cambiando ceros por uno y viceversa) incluido el signo.

Veamos la representación en complemento a 1 de los números 10 y -10 para el caso de $n = 8$ bits.

0	0	0	0	1	0	1	0
+	módulo = 10						

Figura 2.5: Representación del número 10.

1	1	1	1	0	1	0	1
-	módulo = 117						

Figura 2.6: Representación del número -10.

Rango

Para el complemento a 1 el rango de representación de n bits:

$$-2^{n-1} + 1 \leq Rango \leq 2^{n-1} - 1 \tag{2.9}$$

En el caso de $n = 8$ bits, el rango de representación va desde -127 a 127. La ventaja que presenta este sistema frente a otros es la de poseer un *rango simétrico* (igual cantidad de números positivos que negativos). Mientras que su mayor inconveniente es el de poseer *dos representaciones para el número* 0. El cual se representa tanto con un signo positivo con todos sus bits en cero (0), y con signo negativo, con todos los bits en 1.

2.3.3. Complemento a 2

Al igual que los anteriores, este sistema de representación interpreta el bit de más a la izquierda como el signo, correspondiendo el 0 para el signo positivo (+) y el 1 para el signo negativo (−). Para los números positivos, los restantes $(n - 1)$ bits de la derecha representan el módulo (igual que en los dos sistemas anteriores). El negativo de un número positivo se obtiene en dos pasos:

1. Se complementa el número positivo en todos sus bits (cambiando ceros por uno y viceversa), incluido el bit de signo. Es decir se hace el complemento a 1.

2. Al resultado obtenido se le suma 1 (en binario), despreciando el último acarreo si existiera.

Veamos la representación en complemento a 2 de los números 10 y -10 para el caso de $n = 8$ bits.

0	0	0	0	1	0	1	0
+	módulo = 10						

Figura 2.7: Representación del número 10.

1	1	1	1	0	1	1	0
-	módulo = 118						

Figura 2.8: Representación del número -10.

Rango

Para el complemento a 2 el rango de representación de n bits:

$$-2^{n-1} \leq Rango \leq 2^{n-1} - 1 \tag{2.10}$$

Para el caso de $n = 8$ bits, el rango de representación va desde -128 a 127. La principal ventaja es la de tener una única representación para el número 0. Como principal desventaja de este sistema es que el rango de representación de números no es simétrico.

2.3.4. Exceso a 2^{n-1}

Este método de representación no utiliza la convención del bit más significativo para identificar el signo, con lo cual todos los bits representan un número o valor. Este valor se corresponde con el número representado más el exceso, que para n bits viene dado por 2^{n-1}. El signo del número resulta de una operación aritmética. Por ejemplo, para $n = 8$ bits el exceso será 128, con lo cual para representar un número deberá sumársele dicho exceso. De esta manera el número 10, que veníamos representando, recibirá la adición del número 128, con lo que representaremos el número decimal 138 en binario. Por otro lado, el número 10, se representará como el número decimal 118 en binario $(-10 + 128)$. De esta forma quedarán:

1	0	0	0	1	0	1	0
+	módulo = 138						

Figura 2.9: Representación del número 10.

0	1	1	1	0	1	1	0
-	módulo = 118						

Figura 2.10: Representación del número -10.

Rango

El rango de representación en exceso a 2^{n-1} viene dado por:

$$-2^{n-1} \leq Rango \leq 2^{n-1} - 1 \tag{2.11}$$

La principal ventaja de este sistema de representación resulta que tiene una única forma de representar el cero (0). Mientras que su principal desventaja es que el rango resulta asimétrico. La representación del 0 consiste en representar el exceso, por ejemplo 128 para 8 bits.

Resulta interesante observar que todo número representado en exceso a 2^{n-1} tiene la misma representación que un complemento a 2 con el bit de signo cambiado. Podría decirse entonces, que el bit mas significativo representaría el signo pero esta vez utilizando el 0 para el signo positivo (+) y el 1 para el signo negativo (−).

La representación en exceso a 2^{n-1} es simplemente desplazar el cero de su lugar, a donde está el exceso.

2.3.5. Suma en complemento a 1

En la aritmética de complemento a 1, dos números se suman de igual forma que en binario a 1, con la única diferencia que en caso de existir un último acarreo debe sumarse al resultado.

Sumamos los números 10 y −3 en complemento a 1 para $n = 8$ bits

La representación de los números es:

Decimal	Binario (valor absoluto)	Complemento a 1
10	00001010	00001010
−3	00000011	11111100

Por lo tanto la suma:

$$\begin{array}{rcccccccc|l} & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 10_{(\text{CA1}} \\ + & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & -3_{(\text{CA1}} \\ \hline \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 6_{(\text{CA1}} \\ + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1_{(\text{CA1}} \rightarrow \text{carry} \\ \hline & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 7_{(\text{CA1}} \end{array}$$

Sumamos los números 110 y 30 en complemento a 1 para $n = 8$ bits

$$\begin{array}{rcccccccc|l} & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 110_{(\text{CA1}} \\ + & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 30_{(\text{CA1}} \\ \hline & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -115_{(\text{CA1}} \end{array}$$

Cuando sumamos 110 y 30 en complemento a 1 el resultado que obtenemos es −113, esto es debido a que la suma entre estos números es 140 y está fuera del rango posible de representación del complemento a 1 de 8 bits. Obtenemos lo que se conoce como sobrecarga u "*overflow*".

2.3.6. Suma en complemento a 2

En la aritmética de complemento a 2, dos números se suman de igual forma que en complemento a 1, con la única diferencia que se desprecia el último acarreo en el caso que el mismo exista.

Sumamos los números 10 **y** -3 **en complemento a 2 para** $n = 8$ **bits**

La representación de los números es:

Decimal	**Binario (valor absoluto)**	**Complemento a 2**
10	00001010	00001010
-3	00000011	11111101

Por lo tanto para la suma:

$$\begin{array}{cccccccccl} & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 10_{(\text{CA2}} \\ + & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & -3_{(\text{CA2}} \\ \hline \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 7_{(\text{CA2}} \text{ el acarreo se descarta.} \end{array}$$

Sumamos los números 110 **y** 30 **en complemento a 2 para** $n = 8$ **bits**

$$\begin{array}{cccccccccl} & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 110_{(\text{CA2}} \\ + & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 30_{(\text{CA2}} \\ \hline & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -116_{(\text{CA2}} \end{array}$$

Cuando sumamos 110 y 30 en complemento a 2 el resultado que obtenemos es -116, esto es debido a que la suma entre estos números es 140 y también está fuera del rango posible de representación del complemento a 2 de 8 bits. Acá también obtenemos una sobrecarga u "*overflow*".

2.4. Representación en coma o punto fijo

El nombre de esta representación surge de suponer la coma decimal situada en una posición fija. El punto fijo es utilizado para la representación de números enteros, suponiéndose la coma decimal ubicada a la derecha de los bits. Cualquiera de los sistemas de representación de enteros vistos en el apartado anterior es una representación de punto fijo, donde por lo general la parte decimal es nula. También, el programador puede utilizar la representación en punto fijo para representar fracciones binarias escalando los números, de modo que la coma decimal quede ubicada implícitamente en otra posición entre los bits, y en el caso límite a la izquierda de todos ellos describiendo un número fraccional binario puro (menor a 1).

2.5. Representación en coma flotante

La coma o punto flotante surge de la necesidad de representar números reales y enteros con un rango de representación mayor que el que nos ofrece la representación en punto fijo y posibilitar a la computadora el tratamiento de números muy grandes y muy pequeños. Estas ventajas que nos ofrece la coma flotante

traen como contraprestación una disminución (relativamente pequeña) en la precisión de los números representados. En su representación se utiliza la notación científica o exponencial matemática en la que una cantidad se representa de la siguiente forma:

$$numero = mantisa \times base^{exponente} \tag{2.12}$$

Un número en esta notación científica tiene infinitas representaciones, de las que se toma como estándar la denominada *normalizada*. Consiste en que la mantisa no tiene parte entera y el primer dígito o cifra a la derecha del punto decimal es significativo, es decir, distinto de 0, salvo en la representación del número 0.

Representemos el número 835,4 con base de exponenciación 10

Con este ejemplo podemos observar cómo se construyen las infinitas representaciones. A nosotros nos va a interesar sólo la representación normalizada.

$$8354 \times 10^{-1} = 835,4 \times 10^{0} = 83,54 \times 10^{1} = 8,354 \times 10^{2} = 0,8354 \times 10^{3} \tag{2.13}$$

A comienzos de los ´80, cada fabricante de computadoras tenía su propio formato de punto flotante. A fin de rectificar esta situación a fines de los ´70 el IEEE (*Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos) formó un comité para estandarizar la aritmética de punto flotante. Con esto se podría no sólo intercambiar los datos de punto flotante entre diferentes computadoras sino que además proporcionaría a los diseñadores de hardware un diseño que se sabía correcto. Como resultado el IEEE presenta el estándar internacional de punto flotante, más conocido como IEEE 754. El estándar define tres formatos, precisión simple (32 bits), precisión doble (64 bits) y precisión extendida (80 bits). Este último por lo general es utilizado en las unidades aritmético-lógicas de punto flotante, con la intención de reducir los errores por redondeo, por lo que sólo nos limitaremos a hablar de los primeros dos formatos.

En todos los casos, como la computadora utiliza el sistema binario, la base en la que representaremos el número será la base 2. En ambos formatos la palabra binaria en la que se almacena el número en punto flotante se divide en 3 partes. La primera consiste en el signo, la segunda es la representación del exponente y por último la representación de la fracción.

La definición del punto flotante de una computadora sigue las siguientes reglas:

- El primer bit es utilizado como bit de signo, si este es 0 el signo es positivo (+) y si es 1 el signo es negativo (−), tanto en precisión simple como en precisión doble.
- El exponente se representa en exceso a 2^{n-1}, siendo siempre un número entero. Para el caso de la precisión simple el exceso será de 127 y de 1023 para la precisión doble. Los exponentes máximos (255 y 2047) y mínimo (0) se utilizan en casos especiales que describiremos más adelante.
- Los restantes bits representan la fracción del número.

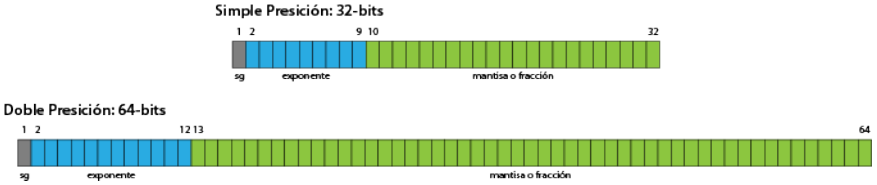

Figura 2.11: Formato de representación IEEE-754 simple y doble precisión

Una fracción normalizada comienza con la coma decimal seguida de un bit en 1 y luego el resto de la fracción. Por ejemplo: $0,10010001010101111101101$. Siguiendo una práctica iniciada en la PDP-11, los autores del estándar se dieron cuenta que el bit inicial de la fracción no tiene por qué almacenarse, ya que puede simplemente darse por hecho que está presente. Por lo tanto la norma define la fracción de una manera un tanto diferente a lo acostumbrado. La fracción consiste en un bit implícito, seguido de la coma decimal y los siguientes 23 o 52 bits correspondientes. De esta manera si los 23 o 52 bits son todos ceros la fracción representada es el número binario $1,0$. Si todos los bits de la fracción están en uno, la fracción representa un número poco menor que el decimal $2,0$.

Por lo general para simplicidad en la lectura, los números escritos en esta norma suelen representarse en su equivalente hexadecimal.

2.5.1. Convertimos en IEEE-754 de simple precisión al número $-6,2734375$

Representamos este número en punto fijo. Primero tomamos la parte entera y la escribiremos en binario y luego convertiremos la parte decimal como vimos anteriormente:

Parte entera: $6_{(10} = 110_{(2}$

$$\begin{aligned} 0,2734375 \quad &\times 2 = \quad 0,546875 \\ 0,546875 \quad &\times 2 = \quad 1,09375 \\ 0,09375 \quad &\times 2 = \quad 0,1875 \\ 0,1875 \quad &\times 2 = \quad 0,375 \\ 0,375 \quad &\times 2 = \quad 0,75 \\ 0,75 \quad &\times 2 = \quad 1,5 \\ 0,5 \quad &\times 2 = \quad 1,0 \\ 0,0 \quad &\times 2 = \quad 0 \end{aligned}$$

Parte decimal: $0,2734375_{(10} = 0,0100011_{(2}$

Entonces el número decimal $-6,2734375$ en binario se escribe $-110,0100011$

Figura 2.12: Convirtiendo el número decimal $-6,2734375$ a binario

Un número representado en coma fija, es igual a un número expresado en notación científica con la base elevada al exponente 0:

$$-6,2734375 \times 10^0 \equiv -110,0100011 \times 2^0 \tag{2.14}$$

Ahora que tenemos la notación científica, hallamos la forma normalizada para representar en IEEE-754:

$$-110,0100011 \times 2^0 = -11,00100011 \times 2^1 = -1,100100011 \times 2^2 \qquad (2.15)$$

Con el número normalizado tenemos el valor del exponente que debemos colocar. En este caso como estamos utilizando precisión simple, 32 bits, el exponente utiliza 8 bits por lo tanto el exceso es de 127:

$$127 + 2 = 129_{(10} = 10000001_{(2} \qquad (2.16)$$

Finalmente escribimos los 32 bits para formar la palabra de precisión simple en la norma IEEE-754. Como el número es negativo, el bit de signo será 1. Luego seguirán los 8 bits correspondientes a la representación del exponente (10000001) y por último los valores que quedaron detrás de la coma decimal en la forma normalizada del número en base 2 (100100011) seguidos de 0 para completar los últimos 23 bits.

signo	exponente	mantisa
1	1000 0001	1001 0001 1000 0000 0000 000

Como mencionamos anteriormente por lo general para simplicidad en la lectura se suele escribir el número en IEEE-754 con sus bytes hexadecimales:

1100	0000	1100	1000	1100	0000	0000	0000
C	0	C	8	C	0	0	0

2.5.2. Casos particulares

En el caso de que el exponente sea todo 0 el número representado en la fracción es un número desnormalizado. El cero se representa con el exponente en 0 y la fracción también en 0. Si tenemos el exponente representando el número máximo podremos representar en caso de tener la fracción en 0 a los infinitos. Si tenemos algún patrón distinto de cero en la fracción tendremos la representación de "ningún número" o NaN (por sus siglas en inglés, Not a Number).

SIGNO	EXPONENTE	MANTISA	VALOR
±	0 <exponente <MAX	X	Número normalizado
±	0	$\neq 0$	Número desnormalizado
±	0	0	Cero
±	111...1	0	Infinitos
±	111...1	$\neq 0$	Ningún número (NaN)

Figura 2.13: Tabla de representación IEEE-754

El rango de representación en punto flotante debe ser analizado teniendo en cuenta los máximos y mínimos valores representables tanto con signo positivo como negativo:

- mínimo número negativo = $-(mantisa_{MAX}) \times base^{exponente_{MAX}}$
- máximo número negativo = $-(mantisa_{MIN}) \times base^{-exponente_{MAX}}$

- mínimo número positivo = $mantisa_{MIN} \times base^{-exponente_{MAX}}$
- máximo número positivo = $mantisa_{MAX} \times base^{exponente_{MAX}}$

Concepto	**Simple**	**Doble**
Cantidad de bits de signo	1	1
Cantidad de bits del exponente	8	11
Cantidad de bits de la fracción	23	52
Total de bits	32	64
Sistema del Exponente	Exceso en 127	Exceso en 1023
Intervalo del Exponente	-126 a +127	-1022 a +1023
Número normalizado más pequeño	2^{-126}	2^{-1022}
Número normalizado más grande	2^{128}	2^{1024}
Intervalo decimal	$\approx 10^{-28}$ a 10^{38}	$\approx 10^{-308}$ a 10^{308}
Número desnormalizado más pequeño	$\approx 10^{-45}$	$\approx 10^{-324}$

Figura 2.14: Rangos y características IEEE-754 Simple y Doble presición

2.6. Representación interna de datos

Los datos e informaciones que se manejan internamente en un sistema informático se pueden representar, según sus características, de la siguiente forma:

Representaciones	alfanuméricas	ASCII
		EBDIC
		Unicode
		...
	numéricas	Coma fija
		IEEE-754
		Complemento a 2
		...

2.6.1. Códigos alfanuméricos

Una computadora puede trabajar internamente con un conjunto de caracteres que nos permitirán manejar datos, informaciones, instrucciones, órdenes de control, etc. Este conjunto de caracteres podemos subdividirlo en los siguientes grupos:

- caracteres alfabéticos
- letras mayúsculas (A..Z sin la Ñ)

- letras minúsculas (a..z sin la ñ)
- cifras decimales: los números 0, 1, ..., 9
- caracteres especiales
- caracteres como el . , ; : * @, etc.
- órdenes de control: Equivalen a las teclas ENTER, TABULACIÓN, ESC, etc.

En general cada carácter se maneja internamente en una computadora por medio de un conjunto de 8 bits mediante un sistema de codificación binario que denominaremos código de caracteres.

Cada computadora tiene su código de caracteres definidos por el fabricante, si bien la mayoría de ellos adaptan a sus equipos códigos estándar de los ya establecidos. En estos códigos se representa cada carácter por medio de un byte, con lo cual todo tipo de información puede ser utilizada internamente, formando cadenas de bytes sucesivos que representarán cadenas de caracteres para que la máquina las maneje e interprete. No todos los tipos de códigos utilizan para la representación de caracteres los ocho bits de un byte; en la actualidad se tiende a utilizar códigos de 8 bits aunque siguen existiendo algunos códigos de 6 y 7 bits.

El primer código alfanumérico fue el código Baudot de 5 bits (aunque no interpretado como nosotros lo hacemos ahora) se utilizaba en los teletipos de principios del siglo XX. Los primeros códigos utilizados en informática fueron de 6 bits, que permitían la representación de 64 caracteres, que generalmente se corresponden a:

- 26 letras mayúsculas
- 10 cifras numéricas
- 28 caracteres especiales y de control

Un ejemplo de código de 6 bits es el código FIELDATA utilizado durante la década del ´50 por el ejército de los Estados Unidos. Con el nacimiento de los lenguajes de programación de alto nivel comenzaron a utilizarse códigos de 7 bits que permiten la representación de los mismos caracteres que los códigos de 6 bits añadiendo las letras minúsculas y caracteres cuyo significado son órdenes de control entre periféricos. Un ejemplo muy utilizado de este tipo de códigos es el ASCII (American Standard Code for Information Interchange) de 7 bits. Hoy los códigos utilizados son los de 8 bits, de los cuales los más conocidos son el EBCDIC (Extended Binary Coded Decimal Interchage Code), el ASCII extendido que agrega un bit a la representación extendiendo la cantidad de símbolos disponibles a 256 y el Unicode.

Capítulo 3

Lógica digital

La electrónica digital está fundamentada en la base matemática formada por el álgebra de Boole (George Boole, matemático inglés, 1815-1864). Este método de análisis considera que todos los elementos poseen únicamente dos estados (biestables) o dos valores, verdadero o falso (1 ó 0) que son opuestos entre sí, no permitiéndose nunca la adopción de estados intermedios. Estudiando las distintas asociaciones entre ellos se obtienen las leyes generales sobre los procesos lógicos.

Fue Claude Shannon (matemático e ingeniero norteamericano, 1916-2001) quien aplicó estas técnicas de estudio, a los circuitos compuestos de elementos que sólo pueden adoptar dos estados estables posibles, apareciendo entonces los llamados circuitos lógicos. Puede decirse entonces que el álgebra de Boole es el sistema matemático empleado en el diseño de circuitos lógicos, que nos permite identificar mediante símbolos el objeto de un circuito lógico de modo que su estado sea equivalente a un circuito real.

Es interesante antes de abordar el estudio de las ecuaciones lógicas, comprender algunos conceptos básicos relativos a la teoría de conjuntos como pueden ser:

- **Conjunto:** reunión de elementos caracterizados por poseer una propiedad común.
- **Conjunto universal:** también denominado conjunto unidad es el que incluye la totalidad de los elementos con una propiedad en común.
- **Conjunto particular:** reunión de elementos pertenecientes al conjunto universal, pero que además poseen alguna características particular que los distingue del resto.
- **Conjunto vacío:** aquel que no posee ningún elemento. Se representa por 0.
- **Conjunto complementario** de otro conjunto A (también denominado conjunto negado o inverso): está constituido por todos los elementos del conjunto universal que no pertenecen al conjunto A.

3.1. Álgebra de Boole. Operaciones y teoremas.

Se definen básicamente tres tipos de operaciones sobre las variables del álgebra de Boole o variables booleanas que son: La complementación, la suma y el producto.

3.1.1. La complementación

Sea una variable booleana A, que por el hecho de serlo solamente podrá poseer dos estados. Si en un instante determinado posee el estado lógico 1, diremos que su estado inverso o complementado será el 0. Si por el contrario la variable A posee el estado lógico 0, su complemento será el 1.

El complemento de una variable A se representa simbólicamente por: $\bar{A}$ o $\sim A$ o $\neg A$.

La tabla de verdad de los estados lógicos correspondientes a una variable y a su complementaria o inversa es la siguiente:

A	$\bar{A}$
0	1
1	0

Figura 3.1: Tabla de verdad de la complementación o negación.

3.1.2. La suma

La operación lógica suma entre dos o más conjuntos (o variables booleanas) se representa mediante el signo "+". Por tanto si tenemos C = A + B , leeremos "el conjunto C es la suma de los conjuntos A y B". Sin embargo suele leerse "C es igual a A **o** B" Esta operación se denomina también unión de conjuntos.

La función suma se define mediante la siguiente tabla de la verdad:

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

Figura 3.2: Tabla de verdad de la suma lógica.

En la tabla 3.2 podemos apreciar cómo el resultado de esta operación es 1 lógico cuando la variable A, o la variable B valen 1. Este resultado puede generalizarse para n variables de entrada.

3.1.3. El producto

La operación producto entre dos conjuntos se representa mediante el símbolo "*", y da como resultado un conjunto formado por elementos comunes a dichos conjuntos. Esta operación se denomina también intersección de conjuntos. Por

tanto tendremos que $D = A * B$ representa un producto y se lee "D es igual a A por B", o también "D es igual a A y B." Para mayor comodidad se acostumbra a escribir $D = AB$.

La operación producto se define mediante la siguiente tabla de la verdad:

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

Figura 3.3: Tabla de verdad de la suma lógica.

Observemos en la tabla 3.3 como el resultado del producto será un 1 lógico si y sólo si ambas variables están en 1. Este resultado puede generalizarse para n variables.

3.1.4. Teoremas

Conocidas ya las tres operaciones más elementales del álgebra de Boole, enunciaremos a continuación de la forma más concisa posible, sus teoremas fundamentales:

1. El resultado de aplicar cualquiera de las tres operaciones antes definidas, a variables booleanas, es otra variable booleana y además el resultado es único.

2. **Ley de idempotencia:** tanto la suma como el producto de una variable booleana consigo misma da como resultado la misma variable.

$$A + A = A \tag{3.1}$$

$$A \cdot A = A \tag{3.2}$$

3. **Ley de involución:** una variable booleana negada dos veces, da como resultado la misma variable.

$$\bar{\bar{A}} = A \tag{3.3}$$

4. **Ley conmutativa:** se define respecto a la suma (y al producto) y nos dice que el orden de los sumandos (factores) no altera el resultado.

$$A + B = B + A \tag{3.4}$$

$$A \cdot B = B \cdot A \tag{3.5}$$

5. **Ley asociativa:** se define respecto a las operaciones suma y producto de la siguiente forma:

$$A + (B + C) = (A + B) + C \tag{3.6}$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C \tag{3.7}$$

6. **Ley distributiva:** se define respecto a las operaciones suma y producto de la siguiente forma:

$$A + (B \cdot C) = (A + B) \cdot (A + C) \tag{3.8}$$

$$A \cdot (B + C) = (A \cdot B) + A \cdot C \tag{3.9}$$

7. **Ley de absorción:**

$$A + (A \cdot B) = A \tag{3.10}$$

$$A \cdot (A + B) = A \tag{3.11}$$

8. **Leyes de De Morgan:** (Pueden ser generalizadas para n variables).

$$\overline{A + B} = \bar{A} \cdot \bar{B} \tag{3.12}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B} \tag{3.13}$$

A continuación se muestran algunas relaciones importantes que se deducen de las operaciones booleanas y de los teoremas anteriores:

$$0 + A = A \tag{3.14}$$

$$1 \cdot A = A \tag{3.15}$$

$$0 \cdot A = 0 \tag{3.16}$$

$$1 + A = 1 \tag{3.17}$$

Analizadas las variables booleanas y sus operaciones, pasamos a definir una función booleana como un conjunto de variables booleanas relacionadas entre sí por cualquiera de las tres operaciones ya definidas o una combinación de ellas. En general la representaremos por:

$$f(A, B, C, \ldots) \tag{3.18}$$

indicando que la función f depende de las variables A, B, C, etc. Además podemos asegurar que toda función booleana es también una variable booleana por el Teorema 1.

3.2. Compuertas lógicas

Existe un convenio gráfico para representar dispositivos (electrónicos, hidráulicos, mecánicos, etc.) que lleven a cabo funciones booleanas elementales y que, en función de la combinación o combinaciones diseñadas, se obtendrán funciones más complejas. Estos dispositivos que desarrollan las funciones booleanas, los denominaremos compuertas lógicas y son las compuertas: OR, AND, NOT, NOR, NAND, X-OR y X-NOR.

Compuertas OR

Desarrollan la suma booleana. Su símbolo gráfico está representado en la Figura 3.4 donde podemos apreciar que se trata de una puerta OR de dos entradas y que a su salida nos proporciona la suma de ambas. Su tabla de la verdad corresponde a la suma booleana.

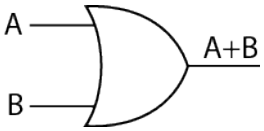

Figura 3.4: Compuerta OR

Si deseamos tener una puerta OR de tres entradas no tendremos más que añadir una tercera línea de entrada y su salida nos dará la suma de las tres variables de entrada.

Compuertas AND

Corresponden al producto booleano de las variables de entrada, proporcionándonos el resultado en su línea de salida. Su símbolo lógico puede verse en la Figura 3.5.

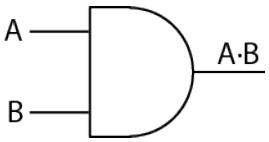

Figura 3.5: Compuerta AND

Como podemos apreciar, se trata de una puerta AND de dos entradas, si deseamos tener una puerta AND de tres entradas no tendremos más que añadir una tercera línea en la entrada obteniendo en la salida el producto de las tres variables de entrada.

Compuertas NOT

Realizan la función complementación o inversión. A estas puertas se las denomina generalmente inversores. Su representación simbólica es la mostrada en la Figura 3.6 , aunque en realidad, la inversión propiamente dicha se representa únicamente por el círculo final del símbolo mostrado, siendo la parte triangular la representación de un amplificador de señal (conocido como *buffer*) que no invierte ni complementa la entrada.

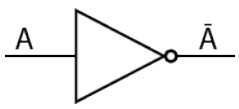

Figura 3.6: Compuerta NOT

Compuertas NOR

Realizan la función inversa de una operación suma, es decir, es la equivalente a una puerta OR complementada. Esta compuerta realiza la misma operación que una compuerta OR conectada con una compuerta NOT a su salida. La función lógica será por tanto:

$$f(A,B) = \overline{A+B} = \bar{A} \cdot \bar{B} \tag{3.19}$$

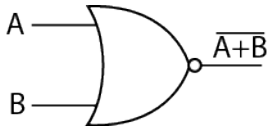

Figura 3.7: Compuerta NOR

En la Figura 3.7 podemos ver su representación simbólica y a continuación en la Figura 3.8 su tabla de la verdad, en la que podemos apreciar cómo la salida es un 1 lógico sólo si las variables de entrada son ambas 0 lógicos.

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

Figura 3.8: Tabla de verdad de la suma lógica negada.

Compuertas NAND

Estas puertas realizan la función lógica:

$$f(A,B) = \overline{A \cdot B} = \bar{A} + \bar{B} \tag{3.20}$$

Por tanto esta función es lo mismo que conectar en serie una función lógica producto y una complementación. Su símbolo lógico se puede observar en la Figura 3.9 y su tabla de la verdad a continuación.

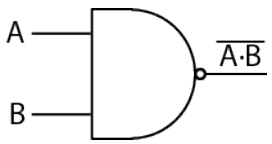

Figura 3.9: Compuerta NAND

Vemos aquí que la salida de esta puerta será 1 lógico siempre que al menos una de sus entradas sea 0 lógico.

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

Figura 3.10: Tabla de verdad del producto lógico negado.

Compuertas X-OR

Son puertas que a su salida nos proporcionan la función lógica:

$$f(A,B) = A \cdot \bar{B} + \bar{A} \cdot B = A \oplus B \tag{3.21}$$

Su símbolo en la Figura 3.11 y su tabla de la verdad a continuación. Como podemos ver la salida de esta puerta es 1 lógico siempre que una y sólo una de sus entradas tenga el nivel lógico 1, es decir sus entradas tienen que poseer valores distintos.

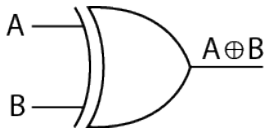

Figura 3.11: Compuerta XOR

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

Figura 3.12: Tabla de verdad de la suma lógica exclusiva.

Compuertas X-NOR

Como su propio nombre indica, realizan la operación inversa de una X-OR, por lo que proporcionan a su salida la función lógica:

$$f(A,B) = \overline{A \cdot \bar{B} + \bar{A} \cdot B} = \overline{A \oplus B} \tag{3.22}$$

En la Figura 3.13 podemos ver su símbolo lógico y a continuación su tabla de verdad.

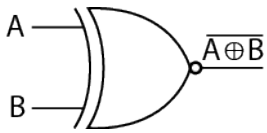

Figura 3.13: Compuerta XNOR

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

Figura 3.14: Tabla de verdad de la suma lógica exclusiva negada.

La salida adopta el valor lógico 1 solamente cuando ambas variables son 0 ó 1 al mismo tiempo.

3.3. Circuitos combinacionales

Un circuito combinacional es un conjunto de compuertas lógicas interconectadas, cuya salida, en un momento dado, es función solamente de los valores de las entradas en ese instante. Como sucede con una compuerta como las vistas anteriormente, la aparición de un valor en las entradas viene seguido casi inmediatamente por la aparición de un valor en la salida, con un retardo propio de la compuerta.

En general, un circuito combinacional consiste de n entradas binarias y m salidas binarias. Como una compuerta, un circuito combinacional puede definirse de tres formas:

- **Tabla de verdad:** Para cada una de las posibles combinaciones de las n señales de entrada, se enumera el valor binario de cada una de las m señales de salida.
- **Símbolo gráfico:** Describe la organización de las interconexiones entre compuertas.
- **Ecuaciones booleanas:** Cada señal de salida se expresa como una función booleana de las señales de entrada.

Los circuitos combinacionales implementan las funciones esenciales de una computadora digital. Sin embargo, ellos no proporcionan memoria, que es un elemento también esencial para el funcionamiento. Para estos fines, se utilizan circuitos lógicos digitales más complejos denominados circuitos secuenciales.

Capítulo 4

Unidades funcionales

4.1. Introducción

"No hay inventos, solo descubrimientos."

Thomas J. Watson, Sr.

Para comprender lo que realmente hay detrás de una computadora, es necesario dedicar mucho tiempo y esfuerzo al estudio de las ciencias de la computación y la ingeniería computacional. Daremos aquí una visión general de la estructura interna y el funcionamiento para introducir los principales conceptos.

Como se expresó en el primer capítulo, un sistema de cómputo consta de un procesador, una memoria, dispositivos de entrada/salida y las interconexiones entre estos componentes principales. En el nivel superior, podemos entender la función de cada una de estos componentes describiendo la estructura de su interconexión y el tipo de señales intercambiadas entre ellas.

Las computadoras en realidad sólo hacen cuatro cosas:

- **recibir entradas:** aceptan datos desde el mundo exterior,
- **producir salidas:** dan información al mundo exterior,
- **procesar información:** llevan a cabo operaciones aritméticas o lógicas con la información, y
- **almacenar información:** mueven y almacenan información en la memoria.

Con estas cuatro operaciones básicas las computadoras cumplen todas sus funciones. Todo sistema de cómputo tiene componentes de hardware dedicados a ellas. Los dispositivos de entrada aceptan entradas del mundo exterior, siendo los más comunes el teclado, el mouse y el joystick. Los dispositivos de salida envían información al mundo exterior. Los más usuales son el monitor y la impresora. Trataremos estos y otros dispositivos en el capítulo dedicado a los periféricos.

El procesador o unidad central de procesamiento (CPU, Central Processing Unit), procesa información, llevando a cabo todos los cálculos aritméticos y tomando decisiones básicas en base a los valores de la información. Se puede decir que es el "cerebro" de la computadora.

Los dispositivos de almacenamiento y la memoria sirven para almacenar información. Los medios de almacenamiento más conocidos son las unidades de disco, diskettes y cintas. La computadora transfiere información entre la memoria y los dispositivos de almacenamiento según se requiera.

La combinación de estos componentes constituye el hardware de un sistema de cómputo.

Recordemos que en el mundo de las computadoras la información es digital. Una computadora no entiende palabras, números, imágenes, notas musicales, ni letras del alfabeto. Sólo pueden digerir información que ha sido dividida en bits, que es la unidad de información más pequeña. Puede parecer extraño pensar que los cajeros automáticos, las consolas de juegos de video, y las supercomputadoras son procesadores de bits. Pero, independientemente de lo que pueda aparentar para el usuario, el núcleo de una computadora digital es una colección de conmutadores de encendido-apagado diseñada para convertir información de una forma a otra. El usuario proporciona a la computadora patrones de bits (entrada) y ésta sigue las instrucciones para transformar esa entrada en otro patrón de bits (salida) y devolverlo al usuario.

Virtualmente todos los diseños de computadoras contemporáneas están basados en los conceptos desarrollados por John von Neumann en el Institute for Advances Studies of Princeton. Tal diseño es conocido como la arquitectura de von Neumann, y se basa en tres conceptos claves:

- los datos e instrucciones están almacenados en una única memoria de lectura-escritura.
- los contenidos de esta memoria son direccionables por posición, sin importar el tipo de los datos guardados en ese lugar.
- la ejecución ocurre de manera secuencial (a menos que se modifique explícitamente) de una instrucción a la siguiente.

4.2. La unidad central de procesamiento

Las transformaciones son realizadas por la unidad central de procesamiento o procesador. Toda computadora tiene una CPU que interpreta y lleva a cabo las instrucciones de los programas, efectúa operaciones aritméticas y lógicas con los datos y se comunica con las demás partes del sistema. Una CPU moderna es un conjunto extraordinariamente complejo de circuitos electrónicos.

Cuando se incorporan todos estos circuitos en una pastilla de silicio, como sucede en la mayoría de las computadoras actuales, a este circuito integrado se lo denomina microprocesador. En una computadora de escritorio corriente, la CPU y otros IC y componentes electrónicos se ubican en una placa de circuitos o "*motherboard*".

En las computadoras personales se utilizan varios IC de CPU distintos. Aunque hay variantes en cuanto al diseño de estos, existen dos factores relevantes para el usuario: la compatibilidad y la velocidad.

4.2.1. Compatibilidad

No todo el software es compatible con todas las CPU. Es posible que el software escrito para un procesador no funcione en otro. Por ejemplo, el softwa-

re escrito para la familia de procesadores Motorola 68000 usados en las viejas computadoras Macintosh, no puede ejecutarse en los procesadores Intel de la mayoría de las computadoras compatibles con IBM; sencillamente, los procesadores Intel no pueden comprender los programas escritos para una CPU de Motorola. En algunos casos se pueden resolver estos problemas utilizando software especial de conversión, pero en general la compatibilidad es una función de la CPU. Para mencionar otro ejemplo, lo mismo sucede con el software escritos para PC que no pueden ejecutarse en los nuevos celulares inteligentes, "*smartphones*".

4.2.2. Velocidad

Hay una enorme diferencia en la rapidez con la cual los procesadores pueden manejar información. La velocidad de una computadora está determinada en gran parte por la velocidad de su reloj interno, el dispositivo cronométrico que produce pulsos eléctricos para sincronizar las operaciones. Por lo general, las computadoras se describen en términos de su velocidad de reloj, medido en unidades llamadas hertz (pulsos por segundo). Pero la velocidad de reloj no es suficiente para describir cuán rápido puede procesar palabras, números o imágenes una computadora.

La velocidad está determinada también por la arquitectura del procesador, esto es, el diseño que establece de qué manera están colocadas en el circuito las componentes individuales de la CPU. De hecho, la arquitectura de todo el sistema de cómputo es parte importante de la ecuación de velocidad. Desde la perspectiva del usuario, el punto crucial es que "más rápido" casi siempre significa "mejor". En la mayoría de las aplicaciones como el procesamiento de texto, es mejor emplear una máquina más rápida, mientras que las aplicaciones que usan muchos gráficos y cálculos necesitan máquinas todavía más rápidas, ya que deben hacer más operaciones por segundo.

Como la velocidad es decisiva, los ingenieros y científicos de la computación constantemente desarrollan técnicas para acelerar la capacidad de la computadora para manipular y mover bits. Una alternativa muy usada consiste en colocar más de un procesador en la computadora, por ejemplo para realizar determinadas operaciones como cálculos matemáticos o presentaciones gráficas.

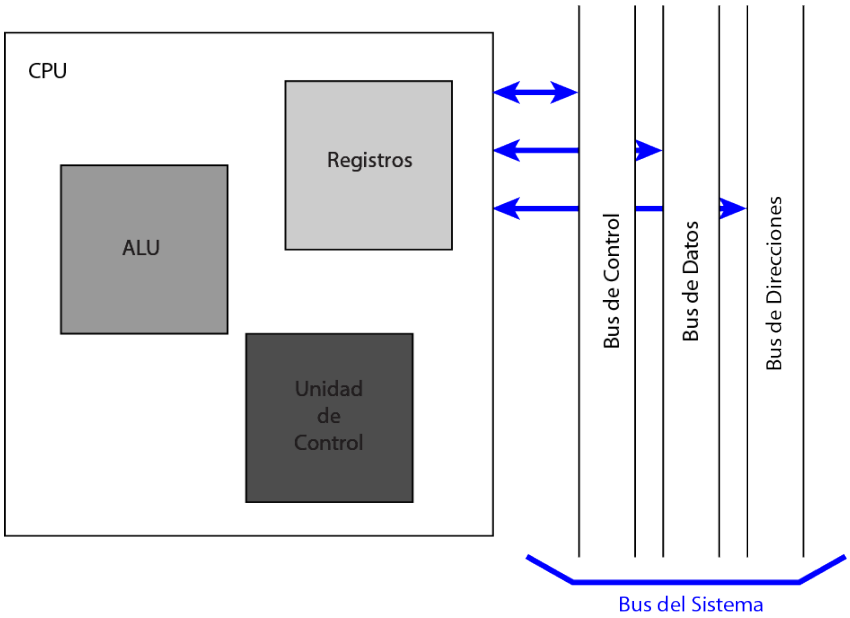

Figura 4.1: Visión simplificada del CPU

Por otro lado, la mayoría de las supercomputadoras tienen varios procesadores completos que pueden dividir los trabajos en porciones y trabajar con ellos en paralelo (procesamiento en paralelo). La Figura 4.1 muestra una visión simplificada de la CPU, indicando la conexión con el resto del sistema vía el bus de comunicación del sistema.

Internamente, las principales componentes de la CPU son una unidad aritmética y lógica (ALU, Arithmetic Logic Unit) y una unidad de control (UC). La ALU realiza la computación real o procesamiento de datos. La UC controla la operación de la ALU. Además, existe una mínima memoria interna de la CPU, consistente de un conjunto de posiciones de almacenamiento, llamadas registros.

Si nos planteamos una visión un poco más detallada de la CPU, existe un elemento llamado bus interno de CPU, necesario para transferir datos entre los distintos registros y la ALU, dado que esta de hecho opera sólo sobre datos en la memoria interna de la CPU.

Los registros de la CPU sirven para dos funciones:

- **Registros visibles al usuario:** permiten al programador en lenguaje de máquina minimizar las referencias a memoria principal optimizando el uso de los registros, que son de acceso más rápido que aquella.
- **Registros de control y estado:** son utilizados por la UC para controlar la operación de la CPU, y por programas del Sistema Operativo para controlar la ejecución de los programas. Entre estos registros se encuentran el contador de programa (que contiene la ubicación de la próxima instrucción a ser ejecutada) y el registro de instrucción (que contiene la última instrucción buscada).

La ALU es la parte de la computadora que ejecuta realmente las operaciones aritméticas y lógicas sobre los datos. Todos los otros elementos del sistema de cómputo (UC, registros, memoria, entrada/salida) están principalmente para traer datos a la ALU para que ésta los procese y luego tomar los resultados y comunicarlos.

Una ALU, y por lo tanto todas las componentes en la computadora están basadas en el uso de simples dispositivos lógicos digitales que pueden almacenar dígitos binarios y realizar operaciones lógicas simples (operaciones booleanas).

Los datos son presentados a la ALU en registros, y los resultados de una operación son almacenados en registros. La ALU también actualiza algunas señales (banderas o flags) como resultado de una operación. La UC provee señales que controlan la operación de la ALU, y el movimiento de los datos desde y hacia la ALU.

4.2.3. Flags

Luego de realizar una operación, las banderas o flags, que no son más que bits en un registro, cambiarán acorde al resultado de la operación. El estado de estos bits pueden ser consultados por el programador por medio de instrucciones especiales; la importancia de estos bits reside en que de acuerdo a su valor (1 ó 0), permitirán la toma de decisiones.

Se presentan aquí 4 de estas banderas, aquellas que son afectadas luego de realizar una suma ó una resta, por eso las llamaremos aritméticas. Otras instrucciones pueden no afectarlas, como así también debemos mencionar que existen otras banderas que se modificarán en respuesta a otras instrucciones. Primero se definirán estas 4 banderas ó bits y luego se verán algunos ejemplos.

- **Z** (cero, Zero en inglés): esta bandera toma el valor 1 indicando que el resultado de la operación fue cero. Para cualquier otro resultado el valor de esta bandera es cero.
- **N** (negativo, Negative en inglés): esta bandera toma el valor del bit más significativo del resultado. Dicho de otra manera si la bandera vale 1 es porque el resultado es negativo, y 0 si el resultado es positivo.
- **V** (desborde, Overflow en inglés): esta bandera vale 1, indicando una condición de desborde (fuera de rango) del resultado en números con signo (complemento a 2). Por condición de desborde se entiende que la cantidad de bits no alcanza para expresar el resultado.
- **C** (acarreo, Carry en inglés): esta bandera toma el valor 1 indicando que hay acarreo en la suma ó borrow en la resta. Cuando esta bandera toma el valor 1 indica una condición de fuera de rango en números sin signo.

Veamos algunos ejemplos, por simplicidad tomemos números de 4 bits:

$$
\begin{array}{cccccl}
 & & \textit{1} & \textit{1} & & \longrightarrow \text{carry} \\
 & 0 & 1 & 0 & 1 & \\
+ & 0 & 1 & 1 & 1 & \\
\hline
 & 1 & 1 & 0 & 0 & \mathbf{N} = 1 \quad \mathbf{Z} = 0 \quad \mathbf{V} = 1 \quad \mathbf{C} = 0
\end{array}
$$

Al lado de la operación realizada se muestra el estado de las banderas. La bandera **N** está en 1 indicando que el resultado fue negativo, esto lo vemos porque el mismo empieza con 1, como corresponde a números negativos en complemento a 2. La bandera **Z** es 0, porque el resultado no fue cero. La bandera de **V** (overflow) está en 1 indicando una condición de desborde en números con signo. ¿Cómo se llega a esta conclusión?. Afortunadamente no es tan difícil, si se observa la suma, se verá que los operandos empiezan con 0, es decir que son positivos, por tanto la suma no puede dar un resultado negativo (que comienza con 1 en complemento a 2). Sólo puede haber desborde en una suma si los dos números son del mismo signo. La bandera **C** es 0 pues no hay carry en el último dígito del resultado. Mirando la bandera de overflow podemos concluir que el resultado en números con signo está mal, pero haciendo la interpretación de números sin signo está bien pues la bandera de carry está en cero.

Analicemos la cuenta, si consideramos números con signo quisimos sumar 5 + 7 = -4 (mal), la cuenta debería dar 12, pero este número no puede ser expresado con 4 bits, pues el positivo más grande que se puede expresar con esta cantidad de dígitos es +7. Si consideramos números sin signo sumamos 5 + 7 = 12, con 4 bits sin signo se puede expresar hasta el 15, por lo tanto la interpretación sin signo es correcta.

Otro ejemplo puede ser el siguiente:

		1	*1*	*1*		$\longrightarrow$ carry
		1	1	0	1	
+		0	0	1	1	
	1	0	0	0	0	**N** = 0 **Z** = 1 **V** = 0 **C** = 1

La bandera N es 0, Z = 1 indica que el resultado es cero, aquí el resultado se considera con 4 bits como los operandos, V = 0, y C = 1 pues hay arrastre en el bit más significativo del resultado, es decir "aparece" un quinto bit.

Interpretando el resultado de acuerdo a las banderas estará bien si sumamos números con signo (V = 0) pero estará mal si la suma representa números sin signo (C = 1). Con signo sumamos –3 + 3 = 0 (correcto) pero sin signo es 13 + 3 = 0 (debería resultar 16 que no puede ser expresado con 4 bits).

Veamos un ejemplo de resta:

	1	0	1	0	1	
-		0	1	1	1	
		1	1	1	0	**N** = 1 **Z** = 0 **V** = 0 **C** = 1

Analicemos el estado de las banderas luego de realizar la operación, N = 1, indica resultado negativo, Z = 0, V = 0 no hay overflow, la condición a analizar es la misma que para la suma, la diferencia es que para la resta el desborde se puede producir cuando los operandos tienen signo contrario y no iguales. C = 1 indica el borrow y por tanto un resultado incorrecto en la resta de números sin signo.

Con signo (complemento a 2) la cuenta es 5 – 7 = -2 resultado correcto (V = 0) y si consideramos sin signo la cuenta resulta 5 – 7 = 14, pues debería dar –2 y no se puede expresar en números sin signo.

4.3. La memoria

"¿Cuánto es uno más uno más uno más uno más uno más uno
más uno más uno más uno?
No lo se, dijo Alicia. Perdí la cuenta.
No sabe sumar, dijo la reina Roja."

Louis Carroll, en A través del espejo.

La función principal de la CPU es obedecer las instrucciones codificadas en los programas. Sin embargo, como Alicia en A través del espejo, la CPU sólo puede manejar una instrucción y unos cuantos datos a la vez. La computadora tiene que "recordar" el resto del programa y los datos hasta que el procesador esté listo para usarlos. Cuando está listo, ¿cómo sabe la CPU dónde están los mismos?

La CPU está conectada con resto de los componentes del sistema a través de 3 buses distintos: el bus de direcciones, el bus de datos y el bus de control. Independientemente de la implementación de cada procesador, la información que viaja por el bus de direcciones tiene como objetivo "identificar" otro componente con el cual la CPU quiere comunicarse, todos los componentes están conectados al bus de direcciones, pero aquél que "reconoce" su dirección queda conectado a la CPU y el resto es como si no estuvieran. Una vez comunicada la CPU con otro componente, puede enviar ó recibir información a través del bus de datos.

Se tiene entonces, un medio para identificar: el bus de direcciones, un medio para transportar el dato propiamente dicho: el bus de datos, y para controlar el intercambio de información: el bus de control.

En este punto se puede preguntar ¿de dónde saca información la CPU para tomar las acciones adecuadas sobre el bus de control? La CPU saca información de la misma instrucción que debe ejecutar, así sabe por ejemplo en que sentido deberían viajar los datos (lectura ó escritura) enviando en consecuencia las señales adecuadas al bus de control.

Trate ahora de responder las siguientes preguntas: dado un tamaño del bus de direcciones ¿cuánta memoria puede direccionar?, o para una determinada cantidad de memoria ¿cuál debe ser el ancho del bus de direcciones?

4.3.1. Modelo de memoria

Suponga un modelo de memoria que está formado por "cajitas" que pueden guardar información. Dentro de esa cajita se escribirá el dato a almacenar (`DATO`) y al costado escribiremos la `DIRECCIÓN` de esa posición de memoria. Recuerde que una dirección es un número binario (combinación de unos y ceros) que identifican a un dispositivo, en este caso una posición de memoria. Cada dirección deberá ser única, a fin de que cuando la CPU quiera comunicarse con una posición de memoria, lo haga con una a la vez.

DIRECCIÓN	DATO

Si se tienen solamente 2 posiciones de memoria, con 1 único bit alcanzará para direccionar cada una sin posibilidad de error.

0	DATO1
1	DATO2

Con 4 posiciones de memoria harían falta 2 bits de direcciones para poder identificar cada una de ellas. Es importante destacar que cuando se habla de cantidad de bits del bus de direcciones, se está hablando de cuántos dígitos binarios necesito para identificar cada celda de memoria.

00	DATO1
01	DATO2
10	DATO3
11	DATO4

En general para identificar N diferentes posiciones de memoria debe resultar que $N = 2^n$ donde n es el número de bits del bus de direcciones.

4.3.2. RAM y ROM

La RAM (*random access memory*, memoria de acceso aleatorio) es el tipo más común de almacenamiento primario o memoria de la computadora. Los dispositivos RAM contienen circuitos que sirven para almacenar temporalmente instrucciones de programas y datos. Un IC de RAM está dividido en posiciones o celdas de igual tamaño, identificadas por una dirección única, de manera que el procesador puede distinguirlas y ordenar que se guarde o recupere información de ella.

La información almacenada en la RAM no es más que un patrón de corriente eléctrica que fluye por circuitos microscópicos en pastillas de silicio. Esto significa que si se interrumpe la energía eléctrica, por cualquier razón, la computadora olvida inmediatamente todo lo que estaba recordando en la RAM. Técnicamente, la RAM es una memoria volátil, ya que la información que contiene no se conserva de manera permanente. Esto representaría un problema muy grave si la computadora no tuviera otro tipo de memoria donde guardar de manera permanente la información importante.

Esta memoria no volátil se denomina ROM (*read-only memory*, memoria de sólo de lectura) porque la computadora puede leer información de ella, pero no escribir nueva información. Todas las computadoras modernas cuentan con dispositivos de ROM que contienen las instrucciones de arranque y otra información crítica. La información en la ROM se graba permanentemente cuando "nace" la computadora, de modo que siempre está disponible cuando ésta opera, pero no puede cambiarse a menos que se reemplace el IC de ROM.

Adicionalmente, existen otros medios donde almacenar información y que constituyen una forma de memoria externa, como por ejemplo los discos rígidos. Nos referiremos a ellos en el capítulo de Periféricos. Aunque aparentemente simple en concepto, la memoria exhibe quizás el mayor rango de tipo, tecnología, organización, performance y costo de todas los componentes. Ninguna tecnología es óptima para satisfacer los requerimientos de memoria de un sistema de cómputo. Como consecuencia, las máquinas están equipadas con un sistema de memoria, compuesto por elementos internos (accesibles directamente por el procesador) y externos (accesibles vía un módulo de entrada/salida). Dentro de la memoria interna encontramos la memoria principal y la memoria local de la CPU (registros). La memoria externa consiste en dispositivos de almacenamiento periférico, como discos y cintas.

Algunas características de la memoria son su capacidad, su velocidad y su costo. Las restricciones de diseño del sistema de memoria de una computadora

pueden ser resumidas por tres preguntas: ¿cuánta?, ¿cuán rápida?, ¿a qué costo?

La pregunta sobre cuánta es de respuesta abierta: si la capacidad está, las aplicaciones serán desarrolladas para usarla.

La pregunta de cuán rápida es un poco más fácil de responder. Para obtener mayores prestaciones, la memoria debe estar acorde con el procesador. Es decir, a medida que la CPU ejecuta instrucciones no queremos que se demore esperando instrucciones o datos.

Para un sistema práctico, el costo de memoria debe ser razonable en relación a los otros componentes.

Como es de esperar, hay una relación entre las tres características clave de la memoria (costo, capacidad y tiempo de acceso):

- menor tiempo de acceso, mayor costo por bit.
- mayor capacidad, menor costo por bit.
- mayor capacidad, mayor tiempo de acceso.

Para obtener las mejores prestaciones, el diseñador deberá elegir y combinar diferentes subsistemas de memoria para poder balancear las partes costosas y rápidas con las económicas y lentas.

4.4. Buses de Entrada/Salida

En una computadora de escritorio corriente, la CPU y los circuitos de memoria se fijan a placas de circuitos junto con otras componentes clave. La información viaja entre las componentes a través de grupos de cables llamados buses o canales. Por lo general, los buses tienen 8, 16 o 32 cables; un bus con 16 cables se denomina bus de 16 bits, ya que puede transmitir 16 bits de información al mismo tiempo, dos veces más que uno de 8 bits. De la misma manera en que una autopista con varios carriles permite que grandes cantidades de vehículos se muevan con mayor rapidez que un camino de un solo carril, los buses más anchos pueden transmitir información con más rapidez que los angostos. Las computadoras más nuevas y potentes cuentan con buses más anchos, para que puedan procesar la información con mayor rapidez.

Además de la CPU y un conjunto de módulos de memoria, el tercer elemento clave de un sistema de cómputo es un conjunto de módulos de entrada/salida (E/S). Cada módulo realiza la interfaz con el bus del sistema y controla uno o más dispositivos periféricos. Un módulo de E/S no es simplemente un grupo de conectores mecánicos que enlazan un dispositivo con el bus del sistema, sino que contiene alguna "inteligencia", es decir, contiene lógica para realizar las funciones de comunicación. Un módulo de E/S es la entidad responsable de controlar uno o más dispositivos externos y de intercambiar datos entre estos dispositivos y la memoria principal y/o los registros de la CPU. Luego, el módulo de E/S debe tener una interfaz interna a la computadora (la CPU y la memoria principal) y una interfaz externa a la computadora (el dispositivo externo).

Algunos canales están conectados a ranuras de expansión en la caja de la computadora, que permiten personalizar las máquinas insertando placas de circuitos de propósito especial en ellas. Otros canales están conectados a puertos

externos, esto es, puntos de conexión en la parte exterior del chasis de la computadora. Ambas formas de expansión simplifican la adición de dispositivos externos o periféricos para que la CPU pueda comunicarse con el mundo exterior y almacenar información que se usará después.

4.5. Funcionamiento: el ciclo de instrucción

La función básica realizada por una computadora es la ejecución de programas. El programa a ser ejecutado consiste de un conjunto de instrucciones almacenadas en la memoria. La CPU realiza el trabajo real ejecutando las instrucciones especificadas en el programa.

Para entender mejor el funcionamiento y la manera en la cual interactúan las principales componentes para ejecutar un programa, necesitamos mirar en más detalle el proceso de ejecución. El punto de vista más simple es considerar el procesamiento de instrucción como consistente de dos pasos: la CPU lee (busca) las instrucciones desde la memoria una a la vez, y las ejecuta. La ejecución del programa consiste en repetir el proceso de búsqueda y ejecución.

Por supuesto, la ejecución de una instrucción puede involucrar en sí misma un número de pasos. En este punto, podemos justificar la simplificación por lo siguiente. La búsqueda de instrucción es una operación común para cada instrucción, y consiste en leer una instrucción desde una posición de memoria. La ejecución puede involucrar varias operaciones y depende de la naturaleza de la instrucción.

El procesamiento requerido para una sola instrucción es llamado ciclo de instrucción. Usando la descripción simplificada, los dos pasos son el ciclo de búsqueda y el ciclo de ejecución. La ejecución se detiene sólo si la máquina es apagada, si ocurre algún error irrecuperable, o se encuentra una instrucción de programa que detenga la computadora.

4.5.1. Los ciclos de búsqueda y ejecución

En el comienzo de cada ciclo de instrucción, la CPU busca una instrucción desde la memoria. En una CPU típica, se usa un registro llamado contador de programa para conocer cuál es la próxima instrucción a ser buscada. A menos que se diga otra cosa, la CPU siempre incrementa el contador de programa después de cada búsqueda de instrucción, de modo de quedar listo para buscar la próxima en secuencia (es decir, la instrucción ubicada en la siguiente posición de memoria).

Esta secuencia puede ser alterada de alguna manera que indicaremos. La instrucción buscada es cargada en un registro de la CPU conocido como registro de instrucción. La instrucción está en la forma de un código binario que especifica qué acción debe tomar la CPU; ésta interpreta la instrucción y realiza la acción requerida. En general, estas acciones caen en 4 categorías:

- **CPU - Memoria:** los datos pueden ser transferidos desde la CPU a la memoria o viceversa.
- **CPU - E/S:** los datos pueden ser transferidos hacia o desde el mundo exterior por una transferencia entre la CPU y el módulo de E/S.

- **Procesamiento de datos:** la CPU puede realizar alguna operación aritmética o lógica sobre los datos.
- **Control:** una instrucción puede especificar que la secuencia de ejecución sea alterada (como una operación de salto o jump). Por ejemplo, la CPU puede buscar una instrucción de la posición 149, que especifica que la próxima instrucción sea buscada en la posición 182. La CPU recordará esto poniendo el 182 en el contador de programa. Así, en el próximo ciclo de búsqueda, la instrucción será buscada en la posición 182 en vez de la 150.

Por supuesto, la ejecución de una instrucción puede involucrar una combinación de estas acciones. El ciclo de ejecución para una instrucción particular puede contener más de una referencia a memoria. Además, en lugar de referencias a memoria, una instrucción puede especificar una operación de E/S.

4.5.2. Interrupciones

Virtualmente todas las computadoras proveen un mecanismo por el cual otros módulos (E/S, memoria) pueden interrumpir el procesamiento normal de la CPU. Estas interrupciones tienen implicancias sobre el ciclo de instrucción y la estructura de interconexión.

Las interrupciones se proveen principalmente como una manera de mejorar la eficiencia de procesamiento. Por ejemplo, la mayoría de los dispositivos externos son mucho más lentos que el procesador. Supongamos que el procesador está transfiriendo datos a una impresora usando el esquema de ciclo de instrucción visto. Después de cada operación de escritura, el procesador tendrá que parar y estar ocioso hasta que la impresora tome el dato. La longitud de esta pausa puede estar en el orden de varios cientos o miles de ciclos de instrucción que no involucren memoria.

Claramente, esto es un gran desperdicio del uso del procesador. Con las interrupciones, el procesador puede ejecutar otras instrucciones mientras la operación de E/S progresa, con la consiguiente ganancia en el uso del procesador.