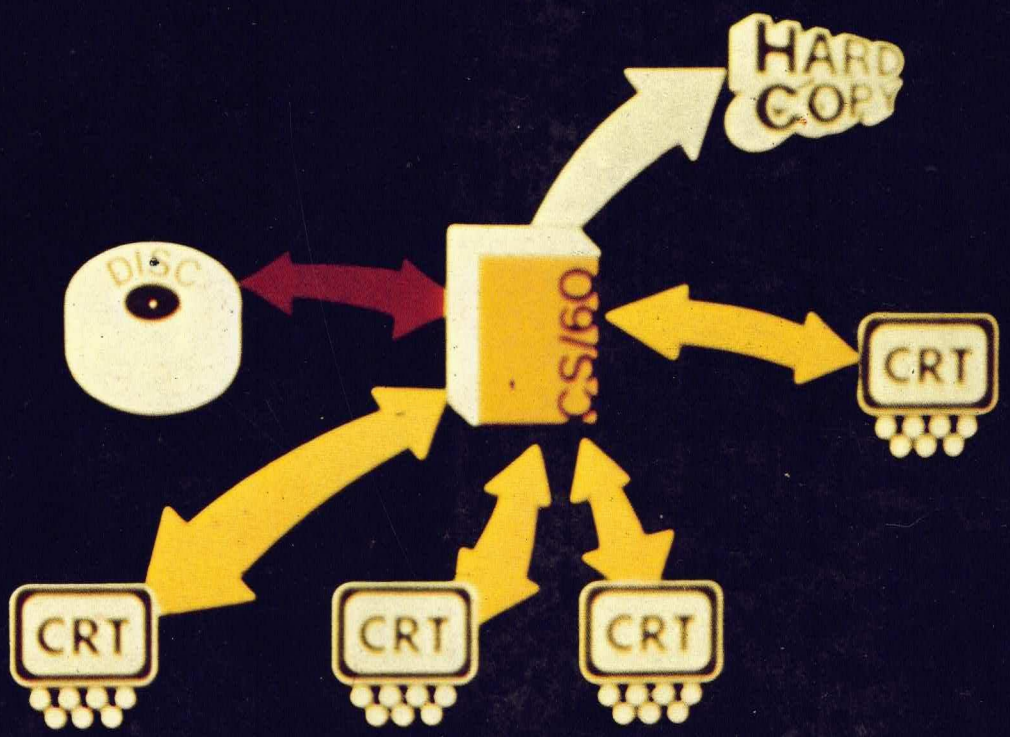
PROGRAMACION DE ORDENADORES

ESTRUCTURA BASICA DEL PROCESO DE DATOS



EDITORIAL tesys

TOMO 2

El presente tratado de **Programación de Ordenadores** está compuesto por un conjunto de temas que constituyen la estructura imprescindible para todo aquel que desee un amplio conocimiento del campo de la Programación.

Dichos temas vienen englobados en volúmenes diversos que a continuación se detallan; constituyéndose por separado como unidades monográficas susceptibles de un estudio particular, al margen del conocimiento conjunto y global de todos ellos, como compendio completo para el logro final de programar.

La agrupación por tomos se desarrolla de la siguiente forma:

Tomo 1. **Estructura Básica del Proceso de datos.**

Tomo 2. **Estructura Básica del Proceso de datos.**

Tomo 3. **Estructura Básica del Proceso de datos.**

Tomo 4. **Estructura Básica del Proceso de datos.**

Tomo 5. **Estructura Básica del Proceso de datos.**

Tomo 6. **Lenguaje Basic**

Tomo 7. **Lenguaje Cobol**

Tomo 8. **Lenguaje Cobol**

Tomo 9. **Leyes de Construcción de programas.**

Tomo 10. **Diccionario de Proceso de datos**

El tratamiento de los temas comprende tanto los aspectos teóricos como las aplicaciones al diseño de estructuras básicas de posible uso inmediato, reagrupándose en los siguientes conjuntos específicos que por sí mismos pasan a ser unidades independientes.

Son los siguientes:

Grupo I: Tomos 1, 2, 3, 4 y 5. **(Estructura Básica del Proceso de Datos).**

Grupo II: Tomo 6. **(Lenguaje Basic).**

Grupo III: Tomos 7 y 8. **(Lenguaje Cobol).**

Grupo IV: Tomo 9. **(Leyes de Construcción de Programas).**

PROGRAMACION DE ORDENADORES

ESTRUCTURA BASICA DEL PROCESO DE DATOS

PROGRAMACION DE ORDENADORES

ESTRUCTURA BASICA DEL PROCESO DE DATOS

AUTOR

JOSE LUIS MORALES PASCUAL
Licenciado en Informática

COLABORADORES

CARLOS ANTONIO MARTINEZ TREVIN
Ingeniero de Telecomunicación

ANDRES J. ERNESTO RUANA GIRONES
Ingeniero Industrial

BARCELONA (ESPAÑA)

© E.C.C. - Barcelona, 1980

Editado por:
EDITORIAL TESYS, Barcelona (España), 1980

Primera edición: Octubre, 1980

Depósito legal: B.: 25869-1980 (II)

I.S.B.N. 84-85838-00-9 (Obra completa)
I.S.B.N. 84-85838-02-5 (Tomo II)

Impreso por:
GRAFOS, Arte sobre papel.
Paseo Carlos I, 157 - Barcelona (13).

Impreso en España - Printed in Spain

CONTENIDO

Introducción 9

1. Conceptos matemáticos básicos 11

2. Sistemas de numeración 37

3. Representación de la información 81

4. La lógica del ordenador 111

Soluciones de los ejercicios 149

Indice general 153

INTRODUCCION

En este segundo tomo de la obra «Programación de Ordenadores» se continúa el estudio de la Estructura básica del Proceso de Datos.

La finalidad de este tomo es la de proporcionar la base necesaria para una mayor facilidad en el estudio de la programación.

Siguiendo con atención el desarrollo de los temas tratados a lo largo del libro se obtendrá mayor predisposición para una fácil asimilación de los temas siguientes.

Todos los temas tratados en el libro se han presentado teniendo en cuenta, fundamentalmente, el facilitar el estudio y el hacerlo agradable.

Han sido tratados con el máximo interés la perfecta claridad explicativa de los textos y el acompañamiento de numerosos ejemplos adicionales que facilitan el estudio; asimismo, y al igual que en el tomo I, la exposición del Cuadro Resumen previo a cada tema y los ejercicios de autocomprobación son elementos didácticos cuya finalidad es la de ayudar.

El libro está dividido en cuatro partes, en las que se desarrollan los siguientes temas:

Primero. Bajo el título de «Conceptos matemáticos básicos», se efectúa un repaso de los conceptos de: Potenciación, Logaritmos, Factorial, Coeficiente binómico y de la Teoría de Conjuntos.

Segundo. «Sistemas de Numeración». El estudio de los sistemas numéricos con los que trabajan los ordenadores y su interrelación, es el tema de este capítulo. El sistema binario se desarrolla más ampliamente, por ser el más usual en los ordenadores.

Tercero. Los ordenadores reciben, a través de códigos, toda una serie de instrucciones. Los códigos empleados y la representación de las instrucciones son los temas estudiados aquí.

Cuarto. El ordenador es una máquina lógica, su fundamento, el Algebra de Boole, es ampliamente estudiado en este cuarto y último capítulo.

Nuestra recomendación es que no pase al estudio de un nuevo apartado sin tener la seguridad de que ha comprendido perfectamente el apartado anterior. El cuadro resumen le indicará cuál es la ubicación exacta de un tema dentro del contexto de la lección. Los ejercicios de autocomprobación le proporcionarán la medida exacta de cuál ha sido su asimilación del tema.

Compare las soluciones que obtenga Vd. en los ejercicios, con las que se incluyen al final del libro y vuelva a estudiar el tema que corresponda a aquel posible ejercicio que no haya resuelto correctamente.

1.

CONCEPTOS MATEMATICOS BASICOS

CUADRO RESUMEN DE LA LECCION

Conceptos matemáticos básicos

- Potencia
  - Reglas de la potenciación
  - Formas de potencias
  - Aplicaciones
- Logaritmos
  - Definición
  - Reglas
  - Cambio de base
- Factorial
- Coeficiente binómico
- Conjuntos
  - Conjunto unidad
  - Conjunto vacío
  - Conjunto Universal
  - Subconjuntos
  - Igualdad
  - Conjunto de las partes
  - Complementos
    - Complemento del conjunto universal
    - Complemento de un complemento
  - Unión
  - Intersección

1. CONCEPTOS MATEMATICOS BASICOS

Los conceptos relacionados de algún modo con la materia que se va a estudiar son los siguientes:

— potencias
— logaritmos
— factorial
— coeficiente binómico
— conjuntos

1.1. POTENCIA

La potencia n-sima de un número cualquiera A puede expresarse de la forma siguiente:

$$A^n$$

que significa: *A* elevado a *n*; siendo *n* el *exponente* y *A* la *base*. De donde *n* es un número entero y positivo que indica las veces que se ha de multiplicar la base por sí misma:

$$A^n = \underbrace{A * A * A * \cdots * A}_{n \text{ factores}}$$

Ejemplos:

Sea $A = 10$, $n = 6$; tendremos:

$$A^n = 10^6 = 10 * 10 * 10 * 10 * 10 * 10 = 1000000$$

Sea $A=2$, $n=5$

$$A^n=2^5=2*2*2*2*2=32$$

Sea $A=3$, $n=4$:

$$A^n=3^4=3*3*3*3=81$$

En estos ejemplos, 10, 2 y 3 son *bases*, y 6, 5 y 4 son *exponentes* (enteros y positivos).

Ahora bien, si el exponente es una fracción positiva, *1/n*, tendremos:

$$A^{1/n}=\sqrt[n]{A}$$

de donde $\sqrt[n]{A}$ significa la *raíz n-sima* de A.

Ejemplo:

Sea $A=81$, $n=4$, entonces

$$A^{1/n}=81^{1/4}=\sqrt[4]{81}$$

como 81 es igual a $3*3*3*3=3^4$, tendremos:

$$\sqrt[4]{81}=\sqrt[4]{3^4}=3$$

Para los distintos valores de *n* enteros y positivos se tiene:

— *raíz cuadrada:* para $n=2$, y se representa $\sqrt{A}$, (sin escribir el número 2).

— *raíz cúbica:* para $n=3$, $\sqrt[3]{A}$

— *raíz cuarta:* para $n=4$, $\sqrt[4]{A}$

— *raíz quinta*: para $n=5$, $\sqrt[5]{A}$

— ...

— ...

— *raíz n-sima:* para un valor cualquiera de n, $\sqrt[n]{A}$

5.1.1. Reglas de la potenciación

En las expresiones potenciales se utilizan una serie de reglas que ayudan a la interpretación y resolución de tales expresiones:

1.ª $A^0=1$

Todo número elevado a la potencia cero es igual a 1.

2.ª $A^1=A$

3.ª $A^{-n}=\frac{1}{A^n}$

4.ª $A^{1/n}=\sqrt[n]{A}=$raíz n-sima de A

5.ª $A^m \cdot A^n=A^{m+n}$ $\qquad A^m/A^n=A^{m-n}$

El producto (cociente) de dos potencias de la misma base es igual a la base elevada a la suma (diferencia) de los exponentes.

6.ª $(A^m)^n=A^{m \cdot n}$

7.ª $(A \cdot B)^n=A^n \cdot B^n$

A partir de estas siete reglas fundamentales pueden establecerse otras relaciones. Por ejemplo:

Si $A^{-n}=\frac{1}{A^n}$ entonces $A^{-1/n}=\frac{1}{A^{1/n}}=\frac{1}{\sqrt[n]{A}}$

Si $\frac{A^n}{A^n}=1$ tenemos que $\frac{A^n}{A^n}=A^{n-n}=A^0=1$

Ejemplos:

1.	$A^0=1$	$3^0=1$ $1^0=1$ $10^0=1$ $1024^0=1$	2.	$A^1=A$	$3^1=3$ $1^1=1$ $10^1=10$ $1024^1=1024$
3.	$A^{-n}=\frac{1}{A^n}$	$3^{-1}=\frac{1}{3^1}=\frac{1}{3}$ $3^{-2}=\frac{1}{3^2}=\frac{1}{9}$ $1^{-1}=\frac{1}{1^1}=\frac{1}{1}=1$			$10^{-1}=\frac{1}{10^1}=\frac{1}{10}$ $8^{-4}=\frac{1}{8^4}=\frac{1}{4096}$
4.	$A^{1/n}=\sqrt[n]{A}$	$3^{1/2}=\sqrt{3}$ $243^{1/5}=\sqrt[5]{243}=\sqrt[5]{3*3*3*3*3}=\sqrt[5]{3^5}=3$ $8^{1/3}=\sqrt[3]{8}=\sqrt[3]{2*2*2}=\sqrt[3]{2^3}=2$			
5.	$A^m \cdot A^n=A^{m+n}$ $A^m/A^n=A^{m-n}$	$2^3*2^5=2^{3+5}=2^8=2*2*2*2*2*2*2*2=256$ $4^5/4^2=4^{5-2}=4^3=64$			
6.	$(A^m)^n=A^{m \cdot n}$	$(4^2)^3=4^{2*3}=4^6=4096$			
7.	$(A \cdot B)^n=A^n \cdot B^n$	$(2*6)^3=2^3*6^3=8*216=1728$			

1.1.2. Formas de potencia

En una potencia, A^n, el exponente *n* puede adoptar las siguientes condiciones:

- A^n
  - Si $n<0$
    - Entero $\left(4^{-2}=\frac{1}{4^2}=\frac{1}{4*4}=\frac{1}{16}\right)$
    - Fraccionario
      - Fracción propia $\left(64^{-1/2}=\frac{1}{64^{1/2}}=\frac{1}{\sqrt{64}}=\frac{1}{8}\right)$
      - Fracción decimal $\left(64^{-0.5}=64^{-1/2}=\frac{1}{8}=0.125\right)$
  - Si $n=0$ $(A^0=1$, por definición$)$
  - Si $n>0$
    - Entero $(4^2=4*4=16)$
    - Fraccionario
      - Fracción propia $(64^{1/2}=\sqrt{64}=8)$
      - Fracción decimal $(64^{0.5}=64^{1/2}=8)$

1.1.3. Aplicaciones

El empleo de las potencias es de gran utilidad e importancia sobre todo en *potencias decimales*, es decir, en potencias cuya base es 10.

Veamos unos ejemplos:

10^3 metros $=1$ kilómetro (km)

10^6 bits $=1$ Megabit (Mbit)

10^{-3} segundos $=1$ milisegundo (mseg)

10^{-6} segundos $=1$ microsegundo (μseg)

10^{-9} segundos $=1$ nanosegundo (nseg)

10^{-12} segundos $=1$ picosegundo (pseg)

1.2. LOGARITMOS

Los logaritmos proporcionan un método para reducir las operaciones de cálculo, cómo la multiplicación, división y potenciación, a simples sumas y restas.

Las tablas de logaritmos, inventadas por Neper en 1614, utilizaban como base el número *e* (siendo e=2.71). Más tarde, el inglés Henry Briggs publicó unas tablas de logaritmos en donde se usaba la base 10, consiguiéndose mayor sencillez en los cálculos.

Los logaritmos decimales, es decir, los que emplean la base 10, son los más usuales y se utilizan generalmente para calcular potencias.

Otro sistema logarítmico que se utiliza, sobre todo en Proceso de Datos y en la Teoría de la Información, es el sistema logarítmico binario, es decir, el que utiliza la base 2.

Puesto que en un sistema de proceso de datos las operaciones están basadas en dos estados binarios (0 y 1), el sistema logarítmico que se utiliza es el de base 2.

1.2.1. Definición

El logaritmo L de base *a* del número *N* es el exponente de la potencia a la que hay que elevar *a* para obtener el número *N*, es decir

$$N = a^L$$

o lo que es lo mismo,

$$L = \log_a N$$

por lo tanto, el logaritmo de base *a* de un número *N* ($\log_a N$) es el número al que hay que elevar la base *a* para obtener el número *N*. (Es el exponente de la potencia)

Por ejemplo, sea la base a=10 y el número N=100. Tendremos,

$$N = a^L$$

$$100 = 10^L = 10^2$$

es decir, L=2, o lo que es lo mismo $\log_{10} 100 = 2$.

Supongamos ahora que la base a = 8 y que el número N = 512, el logaritmo de ese número, 512, de base 8, es:

$$N = a^L$$

$$512 = 8^L = 8^3$$

de donde L = 3, es decir,

$$\log_8 512 = 3$$

1.2.2. Reglas de los logaritmos

Para operar con logaritmos se tienen en cuenta las siguientes reglas:

1.ª- $N = a^L$

2.ª- $L = \log_a N$

3.ª- $\log_a 1 = 0$. Ejemplos: $\log_8 1 = 0$; $\log_{10} 1 = 0$

4.ª- $\log_a a = 1$. Ejemplos: $\log_2 2 = 1$; $\log_{10} 10 = 1$

5.ª- $\log_a (A.B) = \log_a A + \log_a B$.
Ejemplo: $\log_{10}(100 * 1000) = \log_{10} 100 + \log_{10} 1000 = 2 + 3 = 5$

6.ª- $\log_a (A/B) = \log_a A - \log_a B$.
Ejemplo: $\log_{10}(100/1000) = \log_{10} 100 - \log_{10} 1000 = 2 - 3 = -1$

7.ª- $\log_a (N^m) = m \cdot \log_a N$. Ejemplo: $\log_{10}(10^3) = 3 \cdot \log_{10} 10 = 3 * 1 = 3$

1.2.3. Cambio de base

Conociendo el logaritmo de base *a* del número N puede deducirse el logaritmo del mismo número en otra base *b*.

$$\log_b N = \frac{1}{\log_a b} \cdot \log_a N$$

Ejemplo:

Calcular el logaritmo de base 2 del número 100.

Como el logaritmo en base 2 no se encuentra en unas tablas como ocurre con el logaritmo decimal, para calcularlo se utiliza la fórmula anterior para pasarlo al logaritmo decimal:

$$\log_2 100 = \frac{1}{\log_{10} 2} \cdot \log_{10} 100$$

Ahora se busca en las tablas de logaritmos decimales y se tiene que:

$$\log_{10} 2 = 0.301$$

$$\log_{10} 100 = 2$$

por lo tanto,

$$\log_2 100 = \frac{1}{0.301} * 2 = 3{,}32 * 2 = 6{,}64$$

1.3. FACTORIAL

El *factorial* de un número cualquiera, N, entero y positivo, viene dado por la siguiente expresión:

$$N! = N * (N-1) * (N-2) \cdots 3 * 2 * 1$$

Ejemplo:

El factorial de 9 es:

$$9! = 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 362880$$

El factorial de 4 es:

$$4! = 4 * 3 * 2 * 1 = 24$$

1.4. COEFICIENTE BINOMICO

El *coeficiente binómico* se representa por $\binom{a}{b}$ que significa *a sobre b* y viene dado por la siguiente expresión:

$$\binom{a}{b}=\frac{a!}{b!\,(a-b)!}$$

siendo *a* y *b* números enteros y positivos y *a* mayor o igual que *b*.

Ejemplos:

¿Qué valor tiene el coeficiente binómico $\binom{5}{4}$?

$$\binom{5}{4}=\frac{5!}{4!\,(5-4)!}=\frac{5!}{4!\,1!}=\frac{120}{24}=5$$

También se puede calcular simplificando del siguiente modo:

$$\binom{5}{4}=\frac{5!}{4!\,(5-4)!}=\frac{5*4*3*2*1}{4*3*2*1}=\frac{5}{1}=5$$

¿Qué valor tiene el coeficiente binómico $\binom{8}{5}$?

$$\binom{8}{5}=\frac{8!}{5!\,(8-5)!}=\frac{8!}{5!\,3!}=\frac{8*7*6}{3*2*1}=\frac{8*7}{1}=56$$

Para calcular los coeficientes binómicos se pueden aplicar también las siguientes reglas:

1.ª $\binom{a}{0}=1$

En este caso se tiene en cuenta que el factorial de cero es 1, por definición, es decir

$$0!=1$$

Ejemplo: $\binom{5}{0}=\frac{5!}{0!\,5!}=\frac{5*4*3*2*1}{1*5*4*3*2*1}=1$

2.ª $\binom{a}{a}=1$ *Ejemplo:* $\binom{5}{5}=\frac{5!}{5!(5-5)!}=\frac{5!}{5!0!}=1$

3.ª $\binom{a}{b}=\binom{a}{a-b}$ *Ejemplo:* $\binom{6}{2}=\binom{6}{4}$

$$\binom{6}{2}=\frac{6!}{2!(6-2)!}=\frac{6!}{2!4!}=15$$

$$\binom{6}{4}=\frac{6!}{4!(6-4)!}=\frac{6!}{4!2!}=15$$

4.ª $\binom{a+1}{b+1}=\binom{a}{b}+\binom{a-1}{b}+\binom{a-2}{b}+\cdots+\binom{b}{b}$

Ejemplo:

$$\binom{5+1}{3+1}=\binom{5}{3}+\binom{4}{3}+\binom{3}{3}=\frac{5!}{3!2!}+\frac{4!}{3!1!}+1=$$

$$=10+4+1=15$$

1.5. CONJUNTOS

Un *conjunto*, A, es una reunión o colección de elementos, a, b, c,...

Cada *elemento* del conjunto A puede ser un objeto o concepto determinado y se dice que *pertenece* al conjunto A.

$$A=\{a,\ b,\ c,\ \ldots\}$$

donde los elementos a, b, c,... pertenecen al conjunto A, y se expresa del siguiente modo:

$$a \in A, \quad b \in A, \quad c \in A$$

Ejemplos:

— Un hombre es un elemento del conjunto de los seres humanos.
— Un animal es un elemento del conjunto de los animales.
— La Física es un elemento del conjunto de las ciencias.
— El mes de Enero es un elemento del conjunto de meses.

1.5.1. Conjunto unidad

Un conjunto que posee un sólo elemento se llama *conjunto unidad.*

Ejemplo:

El conjunto de los dígitos que representan el valor cero está formado por un sólo elemento: 0.

1.5.2. Conjunto vacío

Un conjunto que no tiene elementos se llama *conjunto vacío* y se representa por el símbolo $\varnothing$.

1.5.3. Conjunto universal

El *conjunto universal*, llamado también *conjunto de referencia*, contiene el total de los elementos que existen en la definición de un concepto determinado.

Ejemplo:

Todos los elementos del conjunto de los seres humanos y todos los elementos del conjunto de los animales constituyen el conjunto de los seres vivos.

El conjunto de los seres vivos en este caso puede considerarse como un conjunto universal o de referencia.

Todos los conjuntos que se puedan formar con los elementos del conjunto universal son a su vez elementos del conjunto universal.

Supongamos que A es uno de los conjuntos que constituyen el conjunto universal, por ejemplo, sea A el conjunto de los seres humanos. En la figura 1 está representado gráficamente el conjunto A.

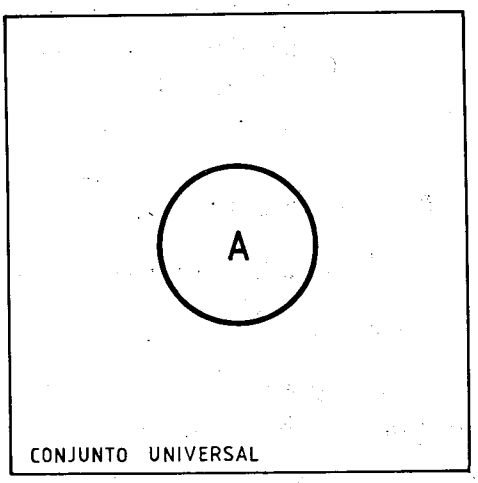


Fig. 1.1. Representación gráfica del conjunto A. *Todos los elementos de* A *son, a su vez, elementos del conjunto universal o conjunto de referencia.*

Este tipo de esquemas se denominan *diagramas de Venn.*

Los diagramas de Venn se utilizan para representar gráficamente los conjuntos y sirven para una mejor explicación y entendimiento del álgebra de conjuntos.

1.5.4. Subconjuntos

Un *subconjunto*, S, de un conjunto, A, es un conjunto cuyos elementos pertenecen al conjunto A.

El subconjunto S puede contener:

— ningún elemento de A.
— algunos elementos de A.
— todos los elementos de A.

Si el subconjunto S no tiene ningún elemento del conjunto A, entonces S se denomina conjunto vacío ($\varnothing$).

El conjunto vacío, por tanto, está contenido en cualquier conjunto dado, es decir, el conjunto vacío es siempre un subconjunto de un conjunto dado.

Para indicar que el conjunto S es un subconjunto del conjunto A se utilizan las siguientes notaciones:

— S es subconjunto de A.
— S está contenido en A: $S \subset A$
— A contiene a S: $A \supset S$.

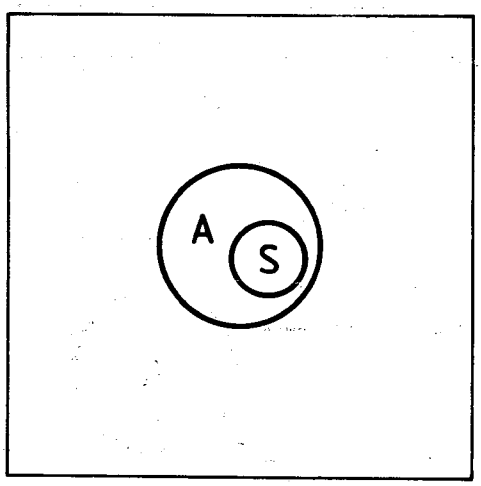


Fig. 1.2. El diagrama de Venn nos muestra claramente cómo el conjunto S está incluido dentro de A. Todos los elementos de S forman un subconjunto de A.

Ejemplo 1:

Sea A el conjunto formado por todos los alumnos de una clase. Sea S el conjunto de los alumnos de esa clase que son menores de 15 años.

Entonces se dice que S es un subconjunto de A, ya que el conjunto A, de todos los alumnos incluye a los alumnos que son menores de 15 años, es decir, incluye o contiene al conjunto S:

$$S \subset A$$

Ejemplo 2:

Supongamos un conjunto universal formado por todos los libros de una biblioteca. Sea A el conjunto de libros que tratan sobre Metemáticas y B el conjunto de libros que tratan sobre Física. Los conjuntos A y B son *subconjuntos* del conjunto universal.

En la figura 1.3 están representados, mediante un diagrama de Venn, los conjuntos A y B contenidos en el conjunto universal. El rectángulo de la figura contiene todos los elementos del conjunto universal, es decir, todos los libros de la biblioteca. El círculo B contiene todos los elementos del subconjunto B, es decir, los libros que tratan sobre Física; y el círculo A contiene todos los elementos que pertenecen al conjunto A, es decir, los libros de Matemáticas.

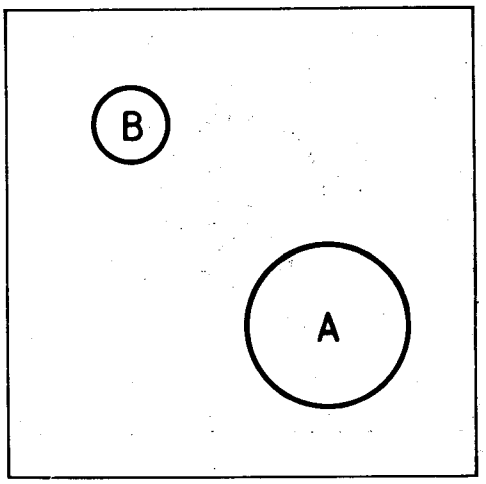


Fig. 1.3. Los libros de matemáticas (conjunto A) y los libros de física (conjunto B) son dos subconjuntos del conjunto de referencia (libros de la biblioteca).

1.5.5. Igualdad

Sean A y B dos conjuntos cualesquiera, estos conjuntos son iguales, es decir, cumplen la relación de *igualdad* (A=B) si se cumple que B está contenido en A y A está contenido en B:

$$\left.\begin{array}{l} B \subset A \\ A \subset B \end{array}\right\} \quad A = B$$

Si A=B, todos los elementos de A pertenecen al conjunto B y todos los elementos de B pertenecen al conjunto A.

Ejemplo:

Sea el conjunto $A=\{7, 3, 9\}$ y sea $B=\{7, 3, 9\}$

se cumple que A=B porque $B \subset A$ y $A \subset B$, es decir, los elementos de A (7, 3, 9) pertenecen a B y los elementos de B pertenecen al conjunto A.

1.5.6. Conjunto de las partes

El *conjunto de las partes* de un conjunto A es el conjunto P(A), cuyos elementos son todos los subconjuntos posibles de A.

Ejemplo:

Sea el conjunto A formado por los siguientes elementos:

$$A=\{4,\ 5,\ 6\}$$

Los subconjuntos posibles de A son:

$$A_1=\{4,\ 5,\ 6\}$$

$$A_2=\{4,\ 5\}$$

$$A_3=\{4,\ 6\}$$

$$A_4=\{5,\ 6\}$$

$$A_5=\{4\}$$

$$A_6=\{5\}$$

$$A_7=\{6\}$$

$$A_8=\{\varnothing\}$$

El conjunto de las partes de A es:

$$P(A)=\{A_1,\ A_2,\ A_3,\ A_4,\ A_5,\ A_6,\ A_7,\ \varnothing\}$$

de donde A_1, A_2, ..., A_7 y $\varnothing$ pertenecen a P(A), es decir,

$$A_1 \in P(A)$$

o, lo que es lo mismo,

$$4,\ 5,\ 6 \in P(A)$$

1.5.7. Complementos

Sea R el conjunto universal o de referencia y sea A un subconjunto de R. El *complemento* de A en R es el conjunto de elementos que pertenecen al conjunto universal R pero que no pertenecen al conjunto A.

El complemento de A se representa por $\bar{A}$, que significa *no A* (o bien *NOT A*).

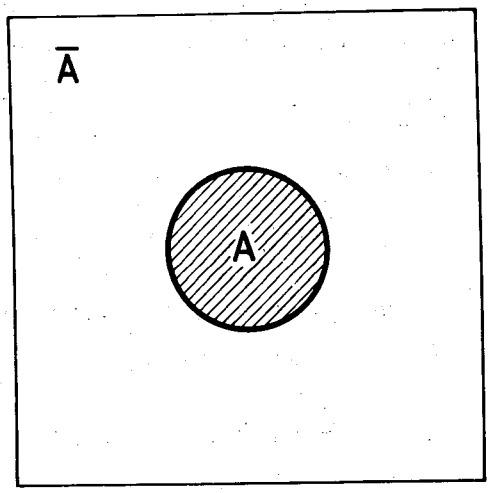


Fig. 1.4. Todos los libros de matemáticas forman el conjunto A. *Todos los libros que no sean de matemáticas, es decir, los libros* no A, *son el complemento de* A.

En la figura 1.4 la zona rayada representa al conjunto A, mientras que el resto del rectángulo (no rayado) representa al complemento de A, es decir $\bar{A}$.

$$\text{Complemento de A} = \text{no A} = \bar{A}$$

Ahora supongamos que a y b son dos elementos que pertenecen al conjunto universal R. Supongamos que a pertenece al conjunto A y que b no pertenece al conjunto A, en cuyo caso pertenecerá al conjunto $\bar{A}$.

$$\text{Es decir: si } a \in A \Rightarrow a \notin \bar{A}$$

esta expresión se lee: «si *a pertenece* ($\in$) a A *implica* ($\Rightarrow$) que *a no pertenece* ($\notin$) a $\bar{A}$ (no A)».

También se puede escribir:

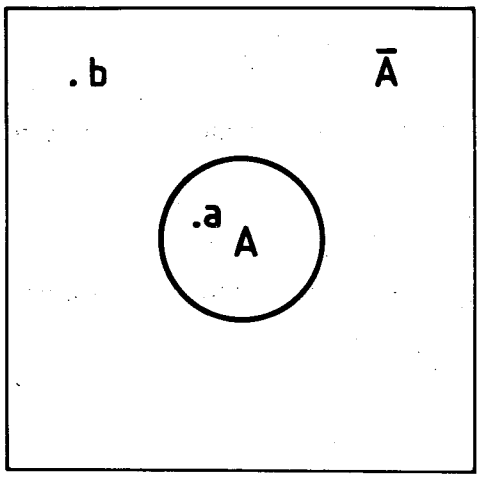
$$\text{si } a \notin A \Rightarrow a \in \bar{A}$$

Estas dos expresiones se pueden escribir en una sola, de la forma:

$$a \in A \Leftrightarrow a \notin \bar{A}$$

que se lee: «*a* pertenece a A si y sólo si ($\Leftrightarrow$) *a* no pertenece a $\bar{A}$».

Por lo tanto, cada uno de los elementos del conjunto universal R per-



$a \in A$
$b \notin A$

Fig. 1.5. El libro de «Algebra» es el elemento a *del conjunto* A *(libros de matemáticas). El libro de «Historia de España» (*b*), es un elemento que no pertenece al conjunto* A. *El libro* b *pertenece al complemento de* A, *es decir, al* no A.

tenece bien al conjunto A o bien al conjunto $\bar{A}$, por lo que se puede escribir la relación:

$$A + \bar{A} = R$$

esta relación expresa:

elementos de A + elementos de $\bar{A}$ = elementos de R

Ejemplo:

Sea el conjunto universal R de los seres vivos. Sea A el conjunto de los seres humanos.

$$A + \bar{A} = R$$

siendo $\bar{A}$ el conjunto de los seres vivos que no son humanos, es decir, el conjunto de los animales, o lo que es lo mismo, el conjunto de los elementos que no pertenecen a A.

Pues bien, todos los elementos del conjunto R de los seres vivos pertenecen bien al conjunto A de los seres humanos o bien al conjunto $\bar{A}$ de los seres vivos *no* humanos.

Complemento del conjunto universal

Sea A un subconjunto del conjunto universal R tal que se cumple la relación:

$$A = R$$

es decir, los elementos del conjunto universal R están incluídos en el subconjunto A de R. Por tanto,

$$A + \bar{A} = R$$

como $A = R$ significa que el conjunto $\bar{A}$ no tiene ningún elemento, es decir, $\bar{A}$ es el conjunto vacío:

$$\bar{\bar{A}} = \varnothing$$

por lo que se puede escribir

$$\left.\begin{aligned} &A + \bar{A} = R \\ &A = R \\ &\bar{A} = \varnothing \end{aligned}\right\} \; R + \varnothing = R$$

lo cual significa que el conjunto vacío es el complemento del conjunto universal. O también, el conjunto universal es el complemento del conjunto vacío.

Complemento de un complemento

El complemento del complemento de un conjunto A es el mismo conjunto A.

$$\bar{\bar{A}} = A$$

Ejemplo:

Sea el conjunto de los seres humanos, A; el complemento $\bar{A}$ corresponde a los animales (ambos conjuntos constituyen el conjunto universal de los seres vivos). El complemento del conjunto de los animales, $\bar{A}$, es el conjunto de los seres humanos.

Es decir, $\bar{\bar{A}} = A$

1.5.8. Unión

La *unión* de dos conjuntos A y B es el conjunto cuyos elementos pertenecen al conjunto A o al conjunto B o a los dos conjuntos.

La unión de A y B se escribe de la forma: $A \cup B$, que significa «A unión B». (Ver figura 1.6).

Ejemplo 1:

Sean los conjuntos A y B (figura 1.7):

$$A = \{1, 2, 3, 4\}$$

$$B = \{3, 5, 7\}$$

$$A \cup B = \{1, 2, 3, 4, 5, 7\}$$

en este caso el elemento 3 pertenece a los dos conjuntos, pero como es el mismo elemento, en la *unión* debe aparecer como un sólo elemento.

Los conjuntos A y B son, en este caso, subconjuntos del conjunto universal de los números naturales.

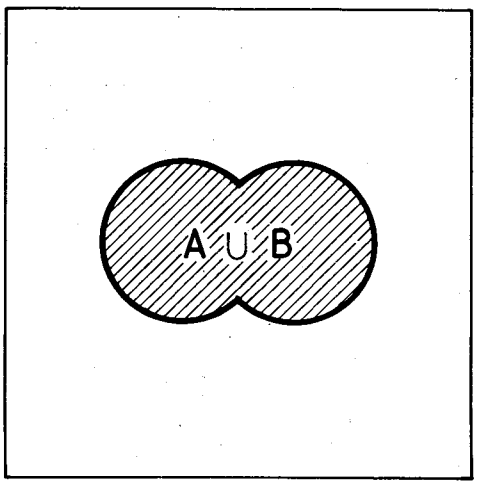


Fig. 1.6. El diagrama de Venn nos representa el conjunto A U B. *Es decir, el conjunto formado por la unión de los conjuntos* A *y* B.

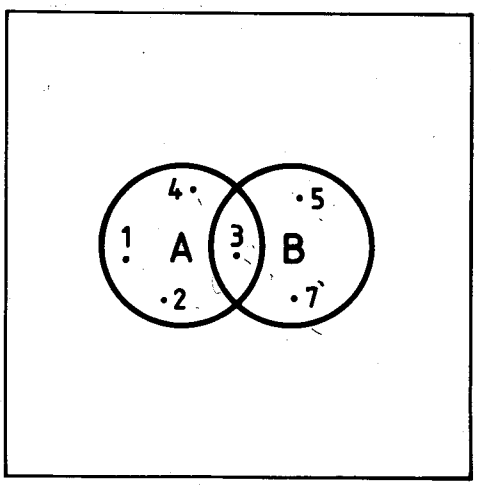


Fig. 1.7. Al unir el conjunto A = *{1, 2, 3, 4}, con el conjunto* B = *{3, 5, 7}, vemos como el elemento 3 pertenece tanto al conjunto* A *como al conjunto* B.

Ejemplo 2:

Sea R el conjunto universal de todos los libros existentes en una biblioteca.

Sea A el conjunto de los libros de Matemáticas y $\bar{A}$ el conjunto de libros que *no* tratan de Matemáticas. Evidentemente se cumple la relación:

$$A \cup \bar{A} = R \quad \text{(conjunto universal)}$$

Supongamos ahora el conjunto B de los libros de Física y $\bar{B}$ los que *no* tratan de Física. Tendremos que

$$A \cup B = \text{libros de Matemáticas} + \text{libros de Física}$$

Si suponemos un tercer conjunto C: el conjunto de los libros de Química, tendremos,

$$A \cup B \cup C = \text{libros de Matemáticas} + \text{libros de Física} + \text{libros de Química}$$

1.5.9. Intersección

La *intersección* de dos conjuntos A y B es el subconjunto de A y de B cuyos elementos pertenecen a la vez a A y a B.

La intersección de A y B se escribe de la forma: $A \cap B$, que significa «A intersección B».

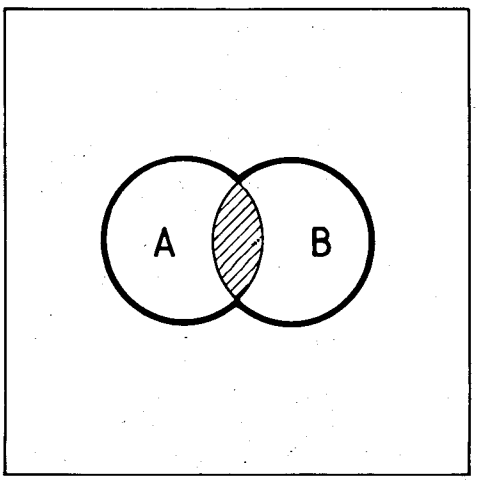


Fig. 1.8. La zona rayada representa la intersección de A *y* B. *Es un subconjunto de* A *y también es un subconjunto de* B.

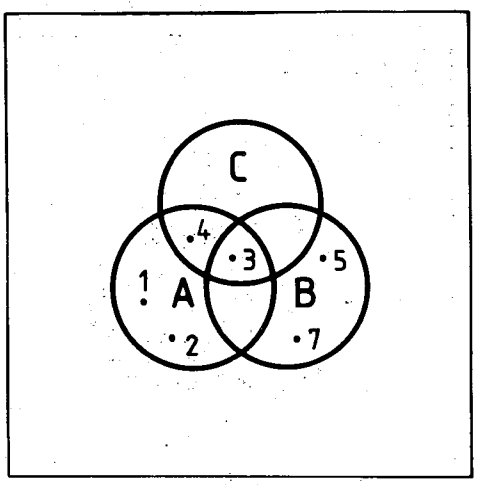


Fig. 1.9. El número 3 es el único que pertenece por igual a los conjuntos A, B y C. La intersección de estos tres conjuntos es, por tanto, igual a 3.

Ejemplo 1:

Sean los subconjuntos A, B y C del conjunto universal de los números naturales (Vea la figura 1.9):

$$A = \{1, 2, 3, 4\}$$

$$B = \{3, 5, 7\}$$

$$C = \{3, 4, 9\}$$

$$A \cap B = \{3\}$$

$$A \cap C = \{3, 4\}$$

$$B \cap C = \{3\}$$

$$A \cap B \cap C = \{3\}$$

Ejemplo 2:

Siguiendo con el ejemplo 2 del apartado anterior, supongamos que existe un libro que trata de las tres materias (Matemáticas, Física y Química). Este libro constituye la intersección de los tres conjuntos

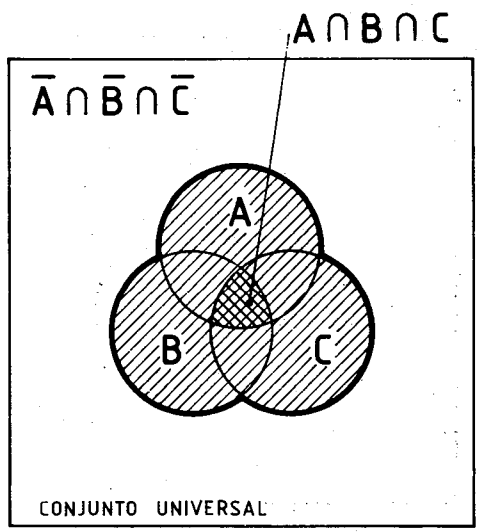


Fig. 1.10. La zona cuadriculada representa el libro que es intersección de los conjuntos A, B *y* C. *La zona blanca sería la intersección de los libros:* no A, no B *y* no C.

Los libros que no tratan de alguna de estas tres materias están representados por:

$$\bar{A} \cap \bar{B} \cap \bar{C}$$

Es fácil demostrar que la intersección de A y su complemento $\bar{A}$ no tiene elementos, es decir, es el conjunto vacío $\varnothing$.

$$A \cap \bar{A} = \varnothing$$

EJERCICIOS

1. Calcular el valor numérico de las siguientes expresiones:

   a) $16^{-0,25}$

   b) $81^{0,25}$

   c) $25^{1/2}$

   d) $2^4 * 2^3$

   e) $(12 * 2)^3$

   f) $\log_{10} 10000$

   g) $\log_3 1$

   h) $\log_{10}(10^7)$

   i) $\binom{7}{4}$

2. Dados los conjuntos:

   $A = \{3, 6, 8, 9\}$

   $B = \{1, 2, 3, 4, 5, 6, 7\}$

   Hallar:

   a) la *unión* de ambos conjuntos

   b) la *intersección*

3. Dado un conjunto R formado por los subconjuntos:

   $A = \{0, 1, 2, 3, 4\}$

   $B = \{5, 6, 7\}$

   $C = \{2, 7, 8, 9\}$

   Calcular:

   a) $\bar{A} \cap B$

   b) $A \cap \bar{B} \cap C$

   c) $A \cap B \cap C$

   d) $A \cup B \cup C$

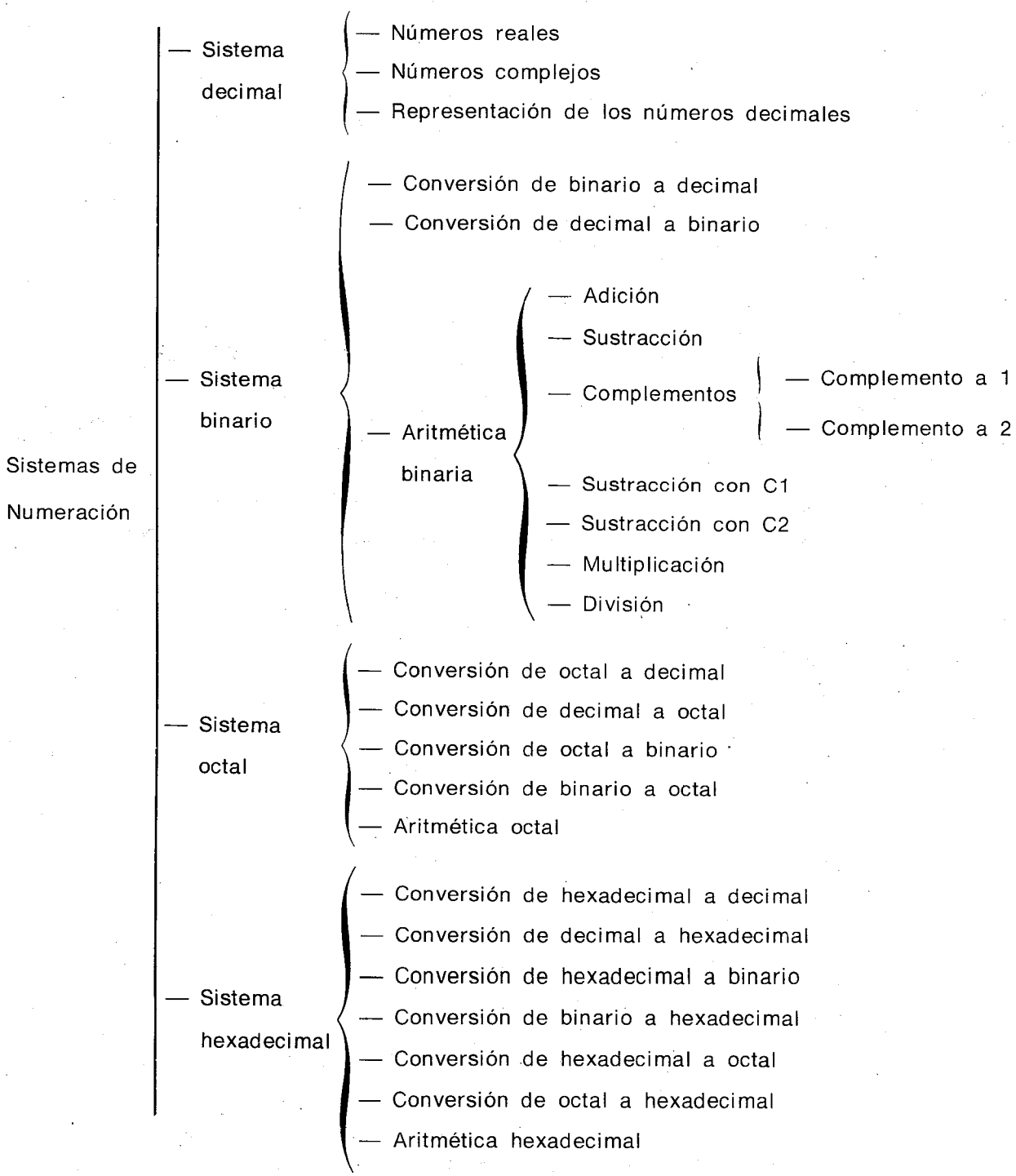
   e) $\bar{A} \cap A$

   f) $\bar{A} \cup A$

2.

SISTEMAS DE NUMERACION

CUADRO RESUMEN DE LA LECCION



2. SISTEMAS DE NUMERACION

Los datos y números que emplean los ordenadores pueden ser dígitos decimales, letras del alfabeto o caracteres especiales. Esta lección trata de los diversos sistemas de numeración que se utilizan en un sistema de proceso de datos, de las relaciones entre ellos y de las operaciones aritméticas que con uno de estos sistemas numéricos pueden realizarse.

En la tabla de la página siguiente puede observarse la representación de varias cantidades en distintos sistemas de numeración.

2.1. SISTEMA DECIMAL

El sistema de numeración decimal es aquel en el que se combinan los diez símbolos, 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9 para representar las cantidades. Estos diez símbolos forman la *base* del sistema decimal.

Los números decimales se clasifican en dos grupos:

— números reales
— números complejos

2.1.1. Números reales

Los números reales comprenden todos los números posibles enteros y fraccionarios, positivos y negativos. Por ejemplo, son números reales los siguientes: — 0.25, — 1.75436, 5, 490, 1478.75 (el punto decimal se utiliza para separar la parte entera de la fraccionaria en lugar de la coma decimal).

Cantidad	*Decimal (10)*	*Binario (2)*	*Octal (8)*	*Hexadecimal (16)*
Cero	0	0	0	0
Uno	1	1	1	1
Dos	2	10	2	2
Tres	3	11	3	3
Cuatro	4	100	4	4
Cinco	5	101	5	5
Seis	6	110	6	6
Siete	7	111	7	7
Ocho	8	1000	10	8
Nueve	9	1001	11	9
Diez	10	1010	12	A
Once	11	1011	13	B
Doce	12	1100	14	C
Trece	13	1101	15	D
Catorce	14	1110	16	E
Quince	15	1111	17	F
Dieciséis	16	10000	20	10
Diecisiete	17	10001	21	11
Dieciocho	18	10010	22	12
Diecinueve	19	10011	23	13
Veinte	20	10100	24	14
Veintiuno	21	10101	25	15
Veintidós	22	10110	26	16
Veintitrés	23	10111	27	17
Veinticuatro	24	11000	30	18
Veinticinco	25	11001	31	19
Veintiséis	26	11010	32	1A
Veintisiete	27	11011	33	1B
Veintiocho	28	11100	34	1C
Veintinueve	29	11101	35	1D
Treinta	30	11110	36	1E
Treinta y uno	31	11111	37	1F
Treinta y dos	32	100000	40	20

La clasificación de los números reales puede verse en el siguiente esquema:

- Números reales
  - — Números racionales
    - — Enteros (0, 1, −1, −80, 105, etc.)
    - — Fracciones decimales periódicas (3.33333 ..., −3.131313 ...)
  - — Números irracionales
    - — Fracciones decimales no periódicas. (e, π, $\sqrt{2}$, etc.)

Los *números racionales* están compuestos por números enteros y por fracciones decimales periódicas, que son las que en su parte decimal poseen dígitos que se repiten indefinidamente. Así, por ejemplo, tenemos que: 1/3=0.3333 ..., a este número se puede añadir tantos dígitos 3 como queramos.

Los *números enteros* están formados por combinaciones de los diez dígitos del sistema decimal sin que aparezca en dichas combinaciones el punto decimal o si aparece, los dígitos a la derecha del punto no han de ser distintos de cero. Así, el número entero 28 se puede expresar también mediante un punto decimal, como, por ejemplo: 28., 28.00, etc.

Los números enteros están constituidos por el número cero, los enteros positivos y los enteros negativos.

Los números enteros positivos se denominan *números naturales* (1, 2, 3, 4, ...).

Las *fracciones decimales periódicas* son números en cuya parte decimal (a la derecha del punto decimal) poseen una serie de dígitos que se repiten indefinidamente. Estos números pueden representarse mediante fracciones propias. Así, por ejemplo, la fracción 1/3 significa 0.33333... A la derecha del último dígito 3 pueden añadirse tantos dígitos 3 como se quiera.

Los *números irracionales* están constituidos por las *fracciones decimales no periódicas*, es decir, por números que cuando se transforman en fracciones decimales posean dígitos a la derecha del punto decimal, que no

se repiten indefinidamente. Por lo tanto, estas fracciones no pueden representarse de manera exacta mediante fracciones propias.

Por ejemplo, la fracción propia 1/2 representa un número exacto, sin embargo, el número π (3.1415926 ...) no puede representarse exactamente mediante fracciones propias. Entonces, la representación exacta del número 3.1415926 ... es mediante el símbolo π.

Otros ejemplos de números irracionales son: el número *e* (2.71828 ...), la mayoría de los logaritmos (log 29, log 13, etc.) y la mayor parte de las raíces ($\sqrt{2}$, $\sqrt{7}$, $\sqrt[3]{5}$, etc.).

2.1.2. Números complejos

Los *números complejos* están formados por dos partes: una parte *real* (A) y otra *imaginaria* (B i)

$$\text{número complejo: } A+B\ i$$

Estando constituida la parte real por números reales (A); la parte imaginaria está formada por números reales (B) y por la raíz cuadrada de -1 (i).

$$i=\sqrt{-1}$$

Ejemplo:

$$4+7\ i=4+7\sqrt{-1}$$

$$5-2\ i=5-2\sqrt{-1}$$

$$-12.5-109.02\ i=-12.5-109.02\sqrt{-1}$$

2.1.3. Representación de los números decimales

Un número decimal se puede representar de distintas formas. Por ejemplo, el número 3809.25, el cual tiene seis dígitos (cuatro enteros y dos decimales) se puede representar también de la siguiente manera:

$$3809.25=3*10^{3}+8*10^{2}+0*10^{1}+9*10^{0}+2*10^{-1}+5*10^{-2}$$

En este número, los exponentes (3, 2, 1, 0, -1, -2) de la base (10) indican la posición o el orden de cada dígito. Los números que aparecen a

la izquierda del signo de multiplicar se llaman coeficientes (3, 8, 0, 9, 2 y 5).

El desarrollo de la expresión anterior sería:

$$3 * 10^3 = 3 * 1000 = 3000$$

$$8 * 10^2 = 8 * 100 = 800$$

$$0 * 10^1 = 0 * 10 = 0$$

$$9 * 10^0 = 9 * 1 = 9$$

$$2 * 10^{-1} = 2 * \frac{1}{10} = 0.2$$

$$5 * 10^{-2} = 5 * \frac{1}{10^2} = 0.05$$

$$3809.25$$

En general, podemos escribir

$$N_{10} = \cdots + C_2 * 10^2 + C_1 * 10^1 + C_0 * 10^0 + C_{-1} * 10^{-1} + C_{-2} * 10^{-2} + \cdots$$

siendo N un número decimal y C_2, C_1, etc., los coeficientes.

Los tres puntos de la izquierda y los tres de la derecha indican que la serie puede extenderse tanto como sea necesario.

2.2. SISTEMA BINARIO

En el sistema de numeración *binario* la base es 2. Sólo se pueden representar cantidades mediante combinaciones de dos dígitos: 0 y 1.

Por ejemplo, el número 20 en base decimal (20_{10}) es equivalente al número 10100 en base binaria (10100_2):

$$20_{10} = 10100_2$$

o también se puede escribir: 20 (base 10) = 10100 (base 2).

2.2.1. Conversión de binario a decimal

Cualquier número binario B representado por los dígitos binarios b_n b_{n-1} ... b_3 b_2 b_1 puede representarse en decimal de la siguiente forma:

$$B = b_n\ b_{n-1} \ldots b_3\ b_2\ b_1$$

$$B = b_n \cdot 2^{n-1} + b_{n-1} \cdot 2^{n-2} + \cdots + b_3 \cdot 2^2 + b_2 \cdot 2^1 + b_1 \cdot 2^0$$

donde b_1, b_2, b_3, ..., b_{n-1}, b_n son los coeficientes; 2 es la base, y 0, 1, 2, ..., $n-2$, $n-1$ son los exponentes. (Empleamos el punto, $\cdot$, en lugar del aspa, $\times$, para representar el producto.)

Por ejemplo, el número binario

$$B = 1101001$$

se puede representar de la siguiente forma:

$$1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

donde los dígitos 0, 1, 2, 3, 4, 5 y 6 son los exponentes (que aumentan en una unidad de derecha a izquierda), el 2 es la base y los dígitos 1, 1, 0, 1, 0, 0 y 1 son los coeficientes.

Si se suman los términos anteriores se obtiene el número decimal equivalente al número binario

$$1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$

$$= 1 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 =$$

$$= 64 + 32 + 0 + 8 + 0 + 0 + 1 =$$

$$= 105$$

Por lo tanto, el número binario 1101001 representa al número decimal 105:

$$1101001_2 = 105_{10}$$

Veamos otro ejemplo: sea el número binario 110100

2^5	2^4	2^3	2^2	2^1	2^0	$\longrightarrow$	32	16	8	4	2	1
1	1	0	1	0	0		1	1	0	1	0	0

$$1 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 32 + 16 + 0 + 4 + 0 + 0 = 52$$

es decir, $110100_2 = 52_{10}$.

2.2.2. Conversión de decimal a binario

Un método para pasar de un número decimal a su equivalente en binario es efectuar divisiones sucesivas del número decimal mediante la base binaria, 2. Los dígitos que componen el número binario están representados por el último cociente y los restos en orden inverso a las divisiones efectuadas.

Ejemplo 1

Sea el número decimal 285.

285	2							
08	142	2						
05	02	71	2					
1	*0*	11	35	2				
		1	15	17	2			
			1	*1*	8	2		
					0	4	2	
						0	2	2
							0	*1*

$285_{10} = 100011101_2$

Ejemplo 2

94	2					
14	47	2				
0	07	23	2			
	1	03	11	2		
		1	*1*	5	2	
				1	2	2
					0	*1*

$94_{10} = 1011110_2$

Otro método para convertir un número decimal en binario es el siguiente:

Ejemplo 3

Se trata de ajustar el número a la siguiente tabla binaria. Convertir 94_{10} en binario:

128	64	32	16	8	4	2	1

Empezaremos por el 64, ya que el 94 es menor que el 128 y por tanto el número 94 no puede contener al 128. El 64 es menor

64	32	16	8	4	2	1	
1	0	1	1	1	1	0	$\longleftarrow 94_{10}$

que el 94, por tanto ponemos un 1 en la columna del 64. Entonces queda: $94-64=30$.

El 32 es mayor que el 30, entonces pondremos un 0 en la columna del 32. Y así sucesivamente:

30 mayor que 16 $\longrightarrow$ 1; $30-16=14$
14 mayor que 8 $\longrightarrow$ 1; $14-8=6$
6 mayor que 4 $\longrightarrow$ 1; $6-4=2$
2 igual a 2 $\longrightarrow$ 1; $2-2=0$
0 menor que 1 $\longrightarrow$ 0

Entonces, $94_{10}=1011110_2$

Ejemplo 4

Convertir por el mismo método el número decimal 45 en binario:

32	16	8	4	2	1	
1	0	1	1	0	1	$\longleftarrow 45_{10}$

$45>32 \longrightarrow 1$
$45-32=13<16 \longrightarrow 0$
$13>8 \longrightarrow 1$
$13-8=5>4 \longrightarrow 1$
$5-4=1<2 \longrightarrow 0$
$1=1 \longrightarrow 1$

entonces, $45_{10}=101101_2$

2.2.3. Aritmética binaria

La aritmética que efectúan la mayoría de los ordenadores se basa en el sistema de numeración binario, aunque también se utilizan los sistemas de numeración octal y hexadecimal.

Los datos de entrada quedan inmediatamente traducidos en forma binaria en el momento de ser almacenados en la memoria del ordenador. Después de efectuarse las operaciones necesarias con estos datos, los resultados vuelven a ser traducidos al sistema más adecuado.

Las operaciones aritméticas binarias son las siguientes:

operación	*operador*
adición	+
sustracción	−
multiplicación	*
división	/

Adición

El símbolo del operador de la suma es +, los elementos se llaman *sumandos* y el resultado *suma*.

Las reglas de la suma binaria son las siguientes:

Sumandos		*Sumas*	
0+0	=	0	
0+1	=	1	
1+0	=	1	
1+1	=	0	y «nos llevamos» 1 a la posición binaria inmediata superior.

La suma de dos dígitos 1 binarios da como resultado dos bits: uno de *suma* (0) y uno de *arrastre* (1). Es decir,

$$1+1=1\ 0$$

bit de arrastre

El máximo valor que se puede representar con un dígito binario es *uno*, por este motivo cuando ha de representarse el valor *dos* se utilizan dos dígitos (1 y 0).

En el sistema decimal el máximo valor que se puede representar con un sólo dígito es *nueve*, por lo que el siguiente valor, *diez*, se representa con dos dígitos (10).

En general, para todos los sistemas de numeración la representación de la base es 10. Por ejemplo, en el sistema binario el dígito 2 no existe, por lo que la suma de $1+1$ es 2_{10}, pero representado en binario es 10.

	base	*representación*
Binario	dos	10
Octal	ocho	10
Decimal	diez	10
Hexadecimal	dieciséis	10

Por tanto, tendremos las siguientes igualdades:

$$2_{10}=10_2$$

$$8_{10}=10_8$$

$$16_{10}=10_{16}$$

Ejemplo

Sumar los números decimales 54 y 26

$$\begin{array}{r} 54 \\ +26 \\ \hline 0 \end{array}$$

la suma de 4 y 6 produce dos dígitos decimales: 0 de suma y 1 de arrastre. Es decir, $4+6=0$ y «nos llevamos» 1 a la siguiente posición decimal,

$$\begin{array}{r} 54 \\ +26 \\ \hline 1 \\ \hline 80 \end{array}$$

dígito de arrastre

Para sumar estos números en binario, en primer lugar se convierten del sistema decimal al binario.

$54_{10} = 110110_2$

$26_{10} = 11010_2$

la suma sería la siguiente:

	1	1	0	1	1	0
+		1	1	0	1	0
						0

la primera suma parcial es $0+0=0$; a continuación se efectúa la suma de $1+1=0$ y se lleva 1 (el 1 entre paréntesis) al orden inmediatamente superior:

				(1)		
	1	1	0	1	1	0
+		1	1	0	1	0
					0	0

En la tercera columna se efectúa la operación $1+1=0$ y se lleva 1 a la siguiente.

			(1)	(1)		
	1	1	0	1	1	0
+		1	1	0	1	0
				0	0	0

La siguiente operación vuelve a ser $1+1=0$ y se lleva 1.

		(1)	(1)	(1)		
	1	1	0	1	1	0
+		1	1	0	1	0
			0	0	0	0

Ahora tenemos $1+1+1$. En este caso se efectúa primero la operación $1+1=0$ y se lleva 1 a la columna siguiente.

$$\begin{array}{cccccccc} & (1) & (1) & (1) & (1) & & \\ & 1 & 1 & 0 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 1 & 0 \\ \hline & & & 0 & 0 & 0 & 0 \end{array}$$

De la anterior operación queda $0+1=1$.

$$\begin{array}{cccccccc} & (1) & (1) & (1) & (1) & & \\ & 1 & 1 & 0 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 1 & 0 \\ \hline & & 1 & 0 & 0 & 0 & 0 \end{array}$$

A continuación se suma $1+1=0$ y se lleva 1.

$$\begin{array}{cccccccc} (1) & (1) & (1) & (1) & (1) & & \\ & 1 & 1 & 0 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 1 & 0 \\ \hline & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Por último, la suma queda:

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 1 & 0 & \longleftarrow 54_{10} \\ + & & 1 & 1 & 0 & 1 & 0 & \longleftarrow 26_{10} \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 0 & \longleftarrow 80_{10} \end{array}$$

La adición en binario, como puede comprobarse, es muy semejante a la suma en decimal. Cuando se suman fracciones de números binarios ocurre lo mismo que cuando se suman en decimal, es decir, el punto fraccionario (punto binario) no influye en el procedimiento.

Ejemplos:

$$\begin{array}{r} 0.1 \\ +\ 0.1 \\ \hline 1.0 \end{array} \qquad \begin{array}{r} 11.01 \\ +\ 10.11 \\ \hline 110.00 \end{array} \qquad \begin{array}{r} 100.11 \\ +\ 101.00 \\ \hline 1001.11 \end{array} \qquad \begin{array}{r} 11001.11011 \\ +\ 110.00110 \\ \hline 100000.00001 \end{array}$$

En cuanto a los *signos* se siguen las mismas reglas que en el sistema decimal.

(−8)	−8	−1000	−24	−11000
+ (+8) ⇒	+8	+1000	+30	+11110
0	0	0000	6	00110

Sustracción

El símbolo del operador de la resta es −, los elementos se llaman *minuendo* y *sustraendo* y el resultado se llama *diferencia*.

minuendo		*sustraendo*		*diferencia*
0	−	0	=	0
0	−	1	=	−1
1	−	0	=	1
1	−	1	=	0

El significado de −1 es el mismo que en el sistema decimal; se resta 1 del dígito inmediatamente a la izquierda. Es decir, si existe un dígito a la izquierda, 0−1 se convierte en 10−1=1, con 1 tomado de la columna inmediata a la izquierda.

Por ejemplo: restar 59_{10} de 92_{10}

	9	2
−	5	9

en primer lugar tenemos que efectuar la operación 2−9, para ello esta operación se convierte en 12−9=3 y se resta 1 a la columna inmediata de la izquierda:

	(−1)	
	9	2
−	5	9
		3

de manera que la siguiente operación es $9-5-1=9-6=3$,

$$\begin{array}{rcc} & 9 & 2 \\ - & 5 & 9 \\ \hline & 3 & 3 \end{array}$$

Para restar estos números en binario se sigue el mismo procedimiento:

$92_{10}=1\ 0\ 1\ 1\ 1\ 0\ 0_2$
$59_{10}=1\ 1\ 1\ 0\ 1\ 1_2$

$$\begin{array}{rccccccc} & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & & & 1 \end{array}$$

en primer lugar se efectúa la operación $0-1=1$ y se resta 1 a la columna siguiente,

$$\begin{array}{rccccccc} & & & & & & (-1) & \\ & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & & & 1 \end{array}$$

La siguiente operación es $0-1=1$ y se resta 1 al siguiente minuendo.

Después de $1-1=0$ queda:

$$\begin{array}{rccccccc} & & & & & (-1) & (-1) & \\ & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & & 0 & 1 \end{array}$$

Ahora tenemos $1-1=0$ y $0-0=0$.

$$\begin{array}{rccccccc} & & & & & (-1) & (-1) & \\ & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & & & 0 & 0 & 1 \end{array}$$

A continuación las dos operaciones de $1-1=0$

$$\begin{array}{rccccccc} & & & & & (-1) & (-1) & \\ & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \\ \hline & & & 0 & 0 & 0 & 0 & 1 \end{array}$$

la operación $0-1=1$ y se resta 1 del siguiente, y, por último, $1-1=0$.

$$\begin{array}{rccccccc} & (-1) & & & & (-1) & (-1) & \\ & 1 & 0 & 1 & 1 & 1 & 0 & 0 \quad \longleftarrow 92_{10} \\ - & & 1 & 1 & 1 & 0 & 1 & 1 \quad \longleftarrow 59_{10} \\ \hline & 0 & 1 & 0 & 0 & 0 & 0 & 1 \quad \longleftarrow 33_{10} \end{array}$$

Ejemplo 1.

$$\begin{array}{r} 0 \\ -1 \\ \hline -1 \end{array}$$

Ejemplo 2.

$$\begin{array}{r} -1 \\ -1 \\ \hline -10 \end{array}$$

Cuando el minuendo es negativo $(-)$ la resta se convierte en suma aunque permanece el signo $(-)$.

Ejemplo 3.

$$\begin{array}{rcccc} - & 1 & 1 & 1 & 1 \\ - & & 1 & 0 & 0 \\ \hline - & 1 & 0 & 0 & 1 \end{array}$$

Ejemplo 4.

$$\begin{array}{rcccc} & & (-1) & & \\ & 1 & 0 & 0 & 1 \\ - & & 1 & 1 & 0 \\ \hline & & & 1 & 1 \end{array}$$

como el minuendo es ahora -1 (negativo) la siguiente operación es sumar el minuendo y el sustraendo 1 de la siguiente columna,

$$\begin{array}{rcccc} & (-1) & (-1) & & \\ & 1 & 0 & 0 & 1 \\ - & & 1 & 1 & 0 \\ \hline & 0 & 0 & 1 & 1 \end{array}$$

Ejemplo 5.

$$\begin{array}{rcccccc} & 1 & 0 & 1 & 1 & 0 & 0 \\ - & 1 & 1 & 1 & 0 & 0 & 1 \\ \hline \end{array}$$

En este caso el minuendo es menor que el sustraendo, entonces lo que se hace es cambiar el orden de forma que el número mayor quede como minuendo; a continuación se efectúa la resta y se coloca el signo menos en la diferencia:

$$\begin{array}{rcccccc} & 1 & 1 & 1 & 0 & 0 & 1 \\ - & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline & 0 & 0 & 1 & 1 & 0 & 1 \end{array}$$

Ejemplo 6.

$$\begin{array}{r} -0.1 \\ -1.0 \\ \hline -1.1 \end{array}$$

Ejemplo 7.

$$\begin{array}{r} 0.1 \\ -1.0 \\ \hline -0.1 \end{array}$$

Ejemplo 8.

$$\begin{array}{rccccccc} & 1 & 0 & 0 & 0 & . & 0 & 0 \\ - & & 1 & 0 & 0 & . & 0 & 1 \\ \hline & 0 & 0 & 1 & 1 & . & 1 & 1 \end{array}$$

Complementos de los números binarios

En las calculadoras mecánicas, las sumas se efectúan girando hacia adelante ruedas dentadas en tantas unidades como indica el número que va a ser sumado. Para la sustracción, las ruedas dentadas giran hacia atrás en tantas unidades como indica el número que va a ser restado. En la unidad de cálculo de un sistema de proceso de datos no existe un mecanismo de este tipo que gire en ambos sentidos, debiendo convertir la sustracción en una adición de forma que el mecanismo de adición debe utilizarse para ambas operaciones.

La conversión de la sustracción en adición se efectúa por medio de los *complementos*. La adición del complemento de un número binario da el mismo resultado que la sustracción de ese número.

En la unidad de cálculo de un ordenador hay un circuito de complementación que realiza las sustracciones mediante el mecanismo de adición.

Un número binario, N, tiene asociados dos números llamados *complemento a 1* (C1) y *complemento a 2* (C2), del número N.

Complemento a 1

Complemento a 1, C1, de un número binario, N, se obtiene restando el número dado, N, de tantos dígitos 1 como dígitos tenga el número N.

Ejemplo:

$$\text{Sea } N = 00111$$
$$C1 = 11111 - 00111 = 11000$$

es decir, el complemento a 1 de un número binario es otro número binario cuyos dígitos son los opuestos a los del número dado. Entonces la operación anterior se hace más sencilla cambiando directamente ceros por unos y los unos por ceros,

$$N = 00111$$
$$C1 = 11000$$

Ejemplo:

$$\text{Sea } N = 1001$$
$$C1 = 0110$$

Por tanto, el complemento a 1 de 0 es 1 y el de 1 es 0.

Complemento a 2

El *complemento a* 2, C2, de un número binario, N, se obtiene a partir del complemento a 1 por adición de un dígito 1 a la posición inferior (de más a la derecha).

Por ejemplo, sea N = 111

el complemento a 1 es C1 = 000

el complemento a 2 será, por tanto,

```
  000 ⟶ C1
+   1
-------
  001 ⟶ C2
```

El complemento a 2 del número N sumado al número N da siempre como resultado el número binario de orden superior inmediato, es decir, de un dígito más.

Ejemplo:

N = 111
C1 = 000
C2 = 000 + 1 = 001

```
  111 ⟶ N
+ 001 ⟶ C2
-------
 1000 ⟶ Suma
```

Ejemplo:

Sea N = 1001
C1 = 0110

```
     0110
+       1
---------
C2=  0111

  1001 ⟶ N
+0111 ⟶ C2
---------
 10000 ⟶ Suma
```

Ejemplo:

Sea N = 00111
C1 = 11000

```
  11000
+     1
-------
C2 = 11001

  00111 ⟶ N
+11001 ⟶ C2
---------
 100000 ⟶ Suma
```

Sustracción utilizando el complemento a 1

Para efectuar una sustracción utilizando complementos, en primer lugar se calcula el complemento del número que lleve a su izquierda el signo menos (números negativos). En una sustracción donde los dos números sean positivos a quien corresponde el signo negativo es al sustraendo, por tanto, hay que calcular el complemento del sustraendo y sumarlo al minuendo.

En el caso de que el minuendo sea negativo, se calcula complemento a 1 del sustraendo y el complemento del minuendo.

Ejemplo 1

Efectuar la operación

```
 11010
-01100
-------
```

en decimal es:

minuendo ⟶ 11010_2 ⟵ 26_{10}
sustraendo ⟶ 01100_2 ⟵ 12_{10}

```
  26
-12
----
```

14_{10}

En primer lugar se calcula el complemento a 1 del sustraendo y a continuación se suma al minuendo.

C1 del sustraendo = 10011

```
  11010  <-- Minuendo
+ 10011  <-- C1 del sustraendo
-------
(1)01101
 ↑------- este 1 se suma al resultado
```

Si a la izquierda aparece un dígito 1 de más, como en este caso, entonces se suma al resultado obteniéndose la *diferencia positiva*:

```
 01101
     1
-------
 01110 = +14₁₀ (diferencia positiva)
```

Ejemplo 2

Minuendo	01100	12
Sustraendo	−11010	−26
		−14₁₀

C1 del sustraendo = 00101

Minuendo	01100
C1	+00101
	10001

C1 de 10001 = 01110

Diferencia negativa = −01110 = -14_{10}

Ejemplo 3

Minuendo	1	C1 del sustraendo = 0
Sustraendo	−1	Minuendo 1
		C1 +0
		1

C1 de 1 = 0

Diferencia negativa = −0

En este caso, el resultado es cero negativo. En los ordenadores que trabajan con este procedimiento de sustracción se distinguen los ceros po-

sitivos de los ceros negativos, teniendo ambos distintos «valores». Por ejemplo, la representación binaria con signo es la siguiente: a los números positivos les preceden un cero y una coma y a los negativos un uno y una coma:

Núms. positivos	*Núms. negativos*
$+3=0,11$	$-3=1,11$
$+2=0,10$	$-2=1,10$
$+1=0,01$	$-1=1,01$
$+0=0,00$	$-0=1,00$
↑	↑
bit de signo (positivo)	bit de signo (negativo)

En caso contrario, es decir, si no aparece el dígito 1 a la izquierda del resultado de sumar el minuendo al C1 del sustraendo, entonces se calcula el C1 de esa suma obteniéndose la *diferencia negativa*.

Ejemplo 4

Supongamos un carácter de 6 bits (sexteto), de los cuales 5 bits son para representar el número y 1 bit para el signo aritmético.

Sea el número 7 (positivo), en binario

$+7=0,00111$

↑ signo positivo

el primer dígito de la izquierda corresponde al signo: positivo, si es 0; negativo, si es 1.

Supongamos la sustracción siguiente:

$$\begin{array}{r} +7 \\ -4 \\ \hline +3 \end{array} \qquad \text{y en binario} \qquad \begin{array}{r} 0,00111 \\ -0,00100 \\ \hline \end{array}$$

la transformación en adición se efectúa por medio del complemento a 1 del número negativo (-4),

C1 de 0,00100 = 1,11011

entonces, la operación a realizar es:

```
  0,00111
  1,11011
-----------
(1)0,00010
 ↑——— este dígito se suma al resultado
```

```
  0,00010
+       1
-----------
  0,00011 = +3₁₀ (diferencia positiva)
  ↑——— signo +
```

Ejemplo 5

Efectuar la siguiente sustracción:

```
  −8          −0,01000
  −4          −0,00100
-----         ---------
−12₁₀
```

como el minuendo y el sustraendo tienen el signo negativo se calcula el complemento a 1 de ambos:

C1 del minuendo = 1,10111
C1 del sustraendo = 1,11011

y se efectúa la suma:

```
   1,10111
   1,11011
-----------
−(1)1,10010
  +      1
-----------
   1,10011
  −0 1100 ——→ C1
```

con el complemento C1 se ha obtenido la diferencia:

$-01100 = -12_{10}$ (diferencia negativa)

Sustracción utilizando el complemento a 2

En la sustracción binaria pueden darse dos casos: 1.°) que el minuendo sea positivo, o 2.°) que el minuendo sea negativo.

Minuendo positivo

En este caso se calcula el complemento a 2 del sustraendo y se suma al minuendo. Esta suma ha de efectuarse teniendo en cuenta que ambos sumandos tengan el mismo número de bits. Si el resultado de esta suma tiene el mismo número de bits que los sumandos entonces se calcula el complemento a 2 de este resultado obteniéndose la *diferencia negativa*.

Si, por el contrario, el resultado de la suma tiene un bit más que los sumandos entonces se suprime este bit de exceso (el bit de más a la izquierda del número) y se obtiene la *diferencia positiva*.

Ejemplo de diferencia positiva:

minuendo	⟶	1 1 0 1 0	26
sustraendo	⟶	− 0 1 1 0 0	−12
			14_{10}

en primer lugar se calcula el complemento a 2 del sustraendo,

```
  0 1 1 0 0 ←— sustraendo
  1 0 0 1 1 ←— C1
+         1
  ---------
  1 0 1 0 0 ←— C2
```

ahora se suma el C2 del sustraendo al minuendo,

```
  1 1 0 1 0 ←— minuendo
+ 1 0 1 0 0 ←— C2 del sustraendo
  ---------
(1) 0 1 1 1 0 ←— suma

 ↑
 └————————— bit de exceso
```

en la suma se produce un bit de exceso.

Este bit se suprime y queda el resultado

$0\ 1\ 1\ 1\ 0 = 14_{10}$ (diferencia positiva)

Ejemplo de diferencia negativa:

minuendo ⟶	0 1 1 0 0	12
sustraendo ⟶	− 1 1 0 1 0	−26
		-14_{10}

el primer paso es hallar el C2 del sustraendo:

1 1 0 1 0 ⟵	sustraendo
0 0 1 0 1 ⟵	C1
+ 1	
0 0 1 1 0 ⟵	C2

el siguiente paso es sumar el C2 hallado al minuendo:

0 1 1 0 0 ⟵	minuendo
+ 0 0 1 1 0 ⟵	C2 del sustraendo
1 0 0 1 0 ⟵	suma

Ahora se halla el complemento a 2 del resultado

1 0 0 1 0 ⟵	suma
0 1 1 0 1 ⟵	C1
+ 1	
0 1 1 1 0 ⟵	C2

obteniéndose así la *diferencia negativa:*

$$-\ 0\ 1\ 1\ 1\ 0 = -14_{10}$$

Minuendo negativo

En este caso se calcula el complemento a 2 del sustraendo y el complemento a 2 del minuendo y se suman entre sí.

Cuando el minuendo es negativo la diferencia es siempre negativa.

Ejemplo:

minuendo ⟶	− 1 0 0 1	− 9
sustraendo ⟶	− 0 1 0 0	− 4
		-13_{10}

```
   1 0 0 1  ←—— minuendo
   0 1 1 0  ←—— C1
+        1
-----------
   0 1 1 1  ←—— C2

   0 1 0 0  ←—— sustraendo
   1 0 1 1  ←—— C1
+        1
-----------
   1 1 0 0  ←—— C2

   0 1 1 1  ←—— C2 del minuendo
+  1 1 0 0  ←—— C2 del sustraendo
-----------
(1) 0 0 1 1  ←—— suma
 ↑
 └———————————bit de exceso (se suprime)
```

el resultado se halla calculando el C2 de la suma anterior:

```
   0 0 1 1  ←—— suma
   1 1 0 0  ←—— C1
+        1
-----------
   1 1 0 1  ←—— C2
```

obteniéndose de este modo la *diferencia negativa:*

$$-1101 = -13_{10}$$

Ejemplos con bit de signo

Supongamos la siguiente sustracción,

```
+7  ——→ 0,0 0 1 1 1
-4  ——→ 1,0 0 1 0 0
-----   -----------
+3(10)
```

```
   0 0 1 0 0  ←—— sustraendo
   1 1 0 1 1  ←—— C1
+        1
-----------
   1 1 1 0 0  ←—— C2
```

```
          0,0 0 1 1 1 ←—— minuendo
        + 1,1 1 1 0 0 ←—— C2 del sustraendo
      (1) 0,0 0 0 1 1
bit de exceso ——↑  ↑———————— signo +
```

La diferencia es positiva: $0{,}00011 = +3_{10}$

Veamos otro ejemplo, ahora con el minuendo negativo:

$$-8 \longrightarrow 1{,}01000$$
$$-4 \longrightarrow 1{,}00100$$
$$-12_{10}$$

```
    0 1 0 0 0 ←—— minuendo
    1 0 1 1 1 ←—— C1
+           1
  1 1 0 0 0 ←—— C2

    0 0 1 0 0 ←—— sustraendo
    1 1 0 1 1 ←—— C1
+           1
  1 1 1 0 0 ←—— C2

    1,1 1 0 0 0 ←—— C2 del minuendo
+ 1,1 1 1 0 0 ←—— C2 del sustraendo
(1) 1,1 0 1 0 0 ←—— suma
bit de exceso ↑  ↑———————— signo –

    1 0 1 0 0 ←—— suma
    0 1 0 1 1 ←—— C1
+           1
  0 1 1 0,0
```

La diferencia es negativa: $1{,}01100 = -12_{10}$.

Multiplicación binaria

El símbolo del operador de la multiplicación es el asterisco (*), los elementos se llaman multiplicando y multiplicador y el resultado es el producto.

Multiplicando	*	*Multiplicador*	=	*Producto*
0	*	0	=	0
0	*	1	=	0
1	*	0	=	0
1	*	1	=	1

Ejemplo:

Multiplicar en binario 45_{10} por 19_{10}

Multiplicando * *Multiplicador*		*Multiplicando* * *Multiplicador*
45_{10} * 19_{10}	⟶	101101_2 * $1001\ 1_2$

```
   45
 * 19
  405
  45
  855
```

```
    101101
  * 10011
    101101
   101101
  000000
 000000
101101
1101010111
```

Las reglas para colocar los puntos binarios son las mismas que en el sistema decimal. Por ejemplo:

```
   10.11
 * 1.1
   1011
  1011
  100.001
```

es decir, el resultado del producto tendrá tantos dígitos a la derecha del punto binario como dígitos tengan entre el multiplicando y el multiplicador a la derecha de el punto binario respectivo.

En cuanto a los *signos*, se aplica la misma regla que en el sistema decimal, es decir, si los signos del multiplicando y multiplicador son iguales el signo del producto es positivo:

$$(+)*(+)=+$$
$$(-)*(-)=+$$

y si los signos son distintos, el signo del producto es negativo:

$$(-)*(+)=-$$
$$(+)*(-)=-$$

Ejemplo:

$$\begin{array}{r} -1011 \\ *\ \ 11 \\ \hline 1011 \\ 1011 \\ \hline -100001 \end{array} \qquad \begin{array}{r} -110 \\ *(-101) \\ \hline 110 \\ 000 \\ 110 \\ \hline 11110 \end{array}$$

División binaria

El símbolo del operador de la división es /, los elementos se llaman dividendo y divisor y el resultado es el cociente.

Dividendo	/	*Divisor*	=	*Cociente*
0	/	1	=	0
1	/	1	=	1

Si el divisor es cero se produce una *indeterminación* (no es posible efectuar la división de 0/0 ó 1/0).

La división se efectúa igual que en el sistema decimal.

Ejemplo:

$$\begin{array}{r|l} 147 & 6 \\ -12 & 24 \\ \hline 027 & \\ -24 & \\ \hline 03 & \end{array}$$

Dividendo ⟶ $147_{10}=10010011_2$

Divisor ⟶ $6_{10}=110_2$

1.ª *operación*

```
 10010011 | 110
-  110    |  1
 0011
```

2.ª *operación*

```
 10010011 | 110
-  110    |  11
 00110
-110
 000
```

3.ª operación

```
 10010011 | 110
-  110    |  110
 00110
-110
 0000
```

4.ª *operación*

```
 10010011 | 110
-  110    |  1100
 00110
-110
 00001
```

5.ª *operación*

```
 10010011 | 110
-  110    |  11000 = 24₁₀ (cociente)
 00110
-  110
 000011 = 3₁₀ (resto)
```

Dividendo = (Cociente * Divisor) + Resto = (24 * 6) + 3 = 144 + 3 = 147

Ejemplo:

```
 27 | 5
-25 | 5
 02
```

```
 11011 | 101
-101   | 101
 00111
-  101
 010
```

2.3. SISTEMA OCTAL

En el sistema de numeración *octal* la base es 8, es decir, tiene ocho dígitos: 0, 1, 2, 3, 4, 5, 6 y 7. Por ejemplo:

$$20_{10} = 24_8.$$

2.3.1. Conversión de octal a decimal

Aquí se hará lo mismo que con el binario pero utilizando la base 8. Se pueden utilizar las siguientes tablas:

...	8^4	8^3	8^2	8^1	8^0

$\longrightarrow$

...	4096	512	64	8	1

Ejemplo 1

Convertir el número octal 3714 en decimal.

512	64	8	1
3	7	1	4

$$3*512+7*64+1*8+4*1=1536+448+8+4=1996$$

es decir, $3714_8=1996_{10}$

Ejemplo 2:

Convertir el número 536_8 a decimal

$$5*8^2+3*8^1+6*8^0=5*64+3*8+6*1=$$
$$=320+24+6=350$$

$$536_8=350_{10}$$

2.3.2. Conversión de decimal a octal

Por el mismo procedimiento anterior, pero a la inversa, obtendremos la conversión de un número decimal a octal.

Ejemplo 1

Sea el número 658_{10}

512	64	8	1
1	2	2	2

$658 > 512 \longrightarrow 1$

es decir, en 658 sólo hay un 512, por lo que se pone 1 en la columna de 512.

$658 - 512 = 146 > 64 \longrightarrow 2$

en el 146 caben dos 64, por lo que se pone un 2 en la columna del 64, es decir, $64 * 2 = 128$.

$$146 - 128 = 18 > 8 \longrightarrow 2$$
$$18 - 16 = 2 > 1 \longrightarrow 2$$

es decir, $658_{10} = 1222_8$

Ejemplo 2

Convertir 493_{10} en octal.

64	8	1	$\longrightarrow$	64	8	1
				7	5	5

$$493 > 64 \longrightarrow 7 \quad (7 * 64 = 448)$$
$$493 - 448 = 45 > 8 \longrightarrow 5 \quad (5 * 8 = 40)$$
$$45 - 40 = 5 > 1 \longrightarrow 5 \quad (5 * 1 = 5)$$

por lo tanto, $493_{10} = 755_8$

Ejemplo 3

Convertir 493_{10} en octal (por otro procedimiento).

493	8	
13	61	8
5	5	7

$\longrightarrow 755_8 = 493_{10}$

Ejemplo 4

Convertir 782_{10} en octal.

$$\begin{array}{rrrr} 782 & \lfloor 8 & & \\ 62 & 97 & \lfloor 8 & \\ 6 & 17 & 12 & \lfloor 8 \\ & 1 & 4 & 1 \end{array} \longrightarrow 1416_8 = 782_{10}$$

2.3.3. Conversión de octal a binario

Para convertir un número octal a binario se representa cada dígito del número octal por grupos de tres dígitos binarios según la tabla siguiente:

Octal	*Binario*
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ejemplo

Dado el número 65470_8, convertirlo en binario.

6 = 110
5 = 101
4 = 100
7 = 111
0 = 000

Conservando el orden del número octal, el binario equivalente es:

$65470_8 = 110\ \ 101\ \ 100\ \ 111\ \ 000_2$

2.3.4. Conversión de binario a octal

Ejemplo:

Convertir el número 11100011101001_2 a octal.

$$\underset{3}{\underline{11}}\ \underset{4}{\underline{100}}\ \underset{3}{\underline{011}}\ \underset{5}{\underline{101}}\ \underset{1}{\underline{001}}$$

$$11\ 100\ 011\ 101\ 001_2 = 34351_8$$

El proceso es el inverso, es decir, dado el número binario se forman grupos de tres dígitos binarios empezando por la derecha, y se obtienen los dígitos del número octal equivalente conservando el mismo orden

2.3.5. Aritmética octal

Las operaciones aritméticas que pueden realizarse con el sistema de numeración octal son las mismas que en el sistema decimal o en cualquier otro sistema. El modo de operar es idéntico en todos los sistemas, pero con la diferencia del número de dígitos con que se opera en cada sistema.

Veamos unos ejemplos de sumas y restas en octal:

1) Sumar 22_8 y 30_8

$$\begin{array}{rcl} 22 & \longrightarrow & 18 \\ \underline{+30} & \longrightarrow & \underline{24} \\ 52_8 & & 42_{10} \end{array}$$

2) Sumar 27_8 y 12_8

$$\begin{array}{r} 27 \\ \underline{+12} \end{array}$$

como en el resto de los sistemas de numeración, cuando el valor de una suma es superior al mayor dígito del sistema, se resta la base y «se lleva» 1 a la siguiente columna. En este ejemplo, el procedimiento a seguir puede ser:

$$7_8 + 2_8 = (9_{10}) - (8_{10}) = 1 \text{ y nos llevamos 1}$$
$$\uparrow \text{ base octal}$$

es decir,

$$\begin{array}{cc} (+1) & (-8) \\ 2 & 7 \\ +1 & 2 \\ \hline 4 & 1_8 \end{array}$$

en decimal sería:

$$27_8 = 23_{10} \qquad \begin{array}{r} 23 \\ +10 \\ \hline 33_{10} = 41_8 \end{array}$$
$$12_8 = 10_{10}$$

3) En la sustracción, el procedimiento es el siguiente:

$$\begin{array}{rcl} 30 & \longrightarrow & 24_{10} \\ \underline{-12} & \longrightarrow & 10_{10} \end{array}$$

cuando el minuendo es menor se le suma la base y se resta 1 al minuendo de la columna siguiente:

$$\begin{array}{cccc} (-1) & (+8) & & \\ 3 & 0 & \longrightarrow & 24_{10} \\ -1 & 2 & \longrightarrow & 10_{10} \\ \hline 1 & 6 & \longrightarrow & 14_{10} \end{array}$$

2.4. SISTEMA HEXADECIMAL

En el sistema *hexadecimal* la base es 16. Los dígitos que utiliza son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

Por ejemplo:

$$20_{10} = 14_{16}$$

2.4.1. Conversión hexadecimal a decimal

Siguiendo con el mismo procedimiento, tendremos:

$$\ldots \quad 16^3 \quad 16^2 \quad 16^1 \quad 16^0 \qquad \longrightarrow \qquad \ldots \quad 4096 \quad 256 \quad 16 \quad 1$$

Ejemplo 1

El número hexadecimal 1B7F será en decimal el siguiente:

4096	256	16	1
1	B	7	F

⟶

4096	256	16	1
1	11	7	15

$$1*4096+11*256+7*16+15*1=$$
$$=4096+2816+112+15=7039$$

es decir, $1B7F_{16}=7039_{10}$

Ejemplo 2

Convertir el número $3C8_{16}$ a decimal.

$$3*16^2+12*16^1+8*16^0=3*256+12*16+8*1=$$
$$=768+192+8=968$$

$$3C8_{16}=968_{10}$$

2.4.2. Conversión de decimal a hexadecimal

Ejemplo 1

Convertir 5492_{10} en hexadecimal:

4096	256	16	1

⟶

4096	256	16	1
1	5	7	4

$5492 > 4096 \longrightarrow 1$
$5492-4096=1396 > 256 \longrightarrow (5*256=1280)$
$1396-1280=116 > 16 \longrightarrow (7*16=112)$
$116-112=4 > 1 \longrightarrow 4$

es decir, $5492_{10}=1574_{16}$

Ejemplo 2

El mismo número, 5492_{10} a hexadecimal por el método de las divisiones sucesivas:

5492	16			
069	343	16		
052	023	21	16	
4	*7*	*5*	*1*	⟵ $1574_{16}=5492_{10}$

Ejemplo 3

94075_{10} a hexadecimal:

94075	16				
140	5879	16			
127	107	367	16		
155	119	47	22	16	
11	*7*	*15*	*6*	*1*	← $16F7B_{16}=94075_{10}$

2.4.3. Conversión de hexadecimal a binario

Al igual que en el sistema octal, un número hexadecimal se convierte a binario formando grupos, ahora, de cuatro dígitos binarios según la tabla siguiente:

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

Ejemplo 1

Convertir el número $7AE90_{16}$ en binario

7	A	E	9	0
0111	1010	1110	1001	0000

$7AE90_{16}=111\ \ 1010\ \ 1110\ \ 1001\ \ 0000_2$

Ejemplo 2

Convertir el número $B38A_{16}$ en binario.

$B = 1011$
$3 = 0011$
$8 = 1000$
$A = 1010$

$B38A_{16} = 1011 \quad 0011 \quad 1000 \quad 1010_2$

2.4.4. Conversión de binario a hexadecimal

Ejemplo

Convertir 111000101001101_2 a hexadecimal.

$$\begin{array}{cccc} \underline{111} & \underline{0001} & \underline{0100} & \underline{1101} \\ 7 & 1 & 4 & D \end{array}$$

$111 \quad 0001 \quad 0100 \quad 1101_2 = 714D_{16}$

2.4.5. Conversión de hexadecimal a octal

Para convertir un número hexadecimal a octal el procedimiento más sencillo es pasar de hexadecimal a binario y de binario a octal.

Ejemplo

Convertir $B38A_{16}$ a octal.

$B38A_{16} = 1011001110001010_2$

$$\begin{array}{cccccc} \underline{1} & \underline{011} & \underline{001} & \underline{110} & \underline{001} & \underline{101} \\ 1 & 3 & 1 & 6 & 1 & 2 \end{array}$$

$B38A_{16} = 131612_8$

2.4.6. Conversión de octal a hexadecimal

El procedimiento a seguir es el mismo que el anterior, pero a la inversa.

Ejemplo

$$70515_8 = 111000101001101_2$$

111	0001	0100	1101
7	1	4	D

$$70515_8 = 714D_{16}$$

2.4.7. Aritmética hexadecimal

Veamos unos ejemplos de sumas y restas en hexadecimal:

$$\begin{array}{r} F \\ +1 \\ \hline 10 \end{array}$$

el dígito hexadecimal mayor es el dígito F. Por tanto, para representar un valor mayor que F se ha de utilizar más de un dígito haxadecimal.

El procedimiento para la suma y resta es el mismo que el que se vio en el sistema octal:

$$F + 1 = (16_{10}) - (16_{10}) = 0 \text{ y nos llevamos 1}$$

↑ base hexadecimal

$$\begin{array}{cc} (+1) & (-16) \\ & F \\ + & 1 \\ \hline 1 & 0 \end{array}$$

$$\begin{array}{cccc} & & (-16) & \\ (+1) & (+1) & (-16) & \\ 7 & F & D & 4 \\ + & 3 & 5 & A \\ \hline 8 & 3 & 2 & E \end{array}$$

$$\begin{array}{cccc} & & (-1) & (+16) \\ 7 & F & D & 4 \\ - & 3 & 5 & A \\ \hline 7 & C & 7 & A \end{array}$$

EJERCICIOS

1. Convertir en números decimales los siguientes números binarios:

   a) 1001110; *b*) 111001; *c*) −111111; *d*) 10001111;
   e) 111101000; *f*) 1000001110

2. Convertir en números binarios los siguientes números decimales:

   a) 38; *b*) 259; *c*) 1202; *d*) 127; *e*) 4096; *f*) 8300

3. Convertir en números decimales los siguientes números octales:

   a) 527; *b*) 4076; *c*) 40010

4. Convertir los siguientes números de decimal a octal:

   a) 19094; *b*) 4096; *c*) 903

5. Pasar de base hexadecimal a base 10 los números:

   a) A10B; *b*) 370CD; *c*) FA9E

6. Pasar de base 10 a base 16:

   a) 3278; *b*) 19041; *c*) 80045

7. Pasar de base 8 a base 2:

   a) 527; *b*) 360; *c*) 1504

8. Pasar de base 2 a base 8:

   a) 100111; *b*) 1100110; *c*) 1000110101

9. Pasar de base 16 a base 2:

   a) AC1; *b*) 498; *c*) 310F

10. Pasar de base 2 a base 10:

    a) 110110101; *b*) 1111010110; *c*) 10110100010

11. Sumar en binario:

    a) 111011001 + 10101011
    b) 110011001 + 1000111000
    c) 1011110 + 1111011

12. Restar en binario:

    a) 111011001 − 10101011
    b) 1101011 − 1110101
    c) 1001000 − 110011

13. Multiplicar en binario:

    a) 1010110 × 1011
    b) 1101011 × 110
    c) 111.1001 × 11.01

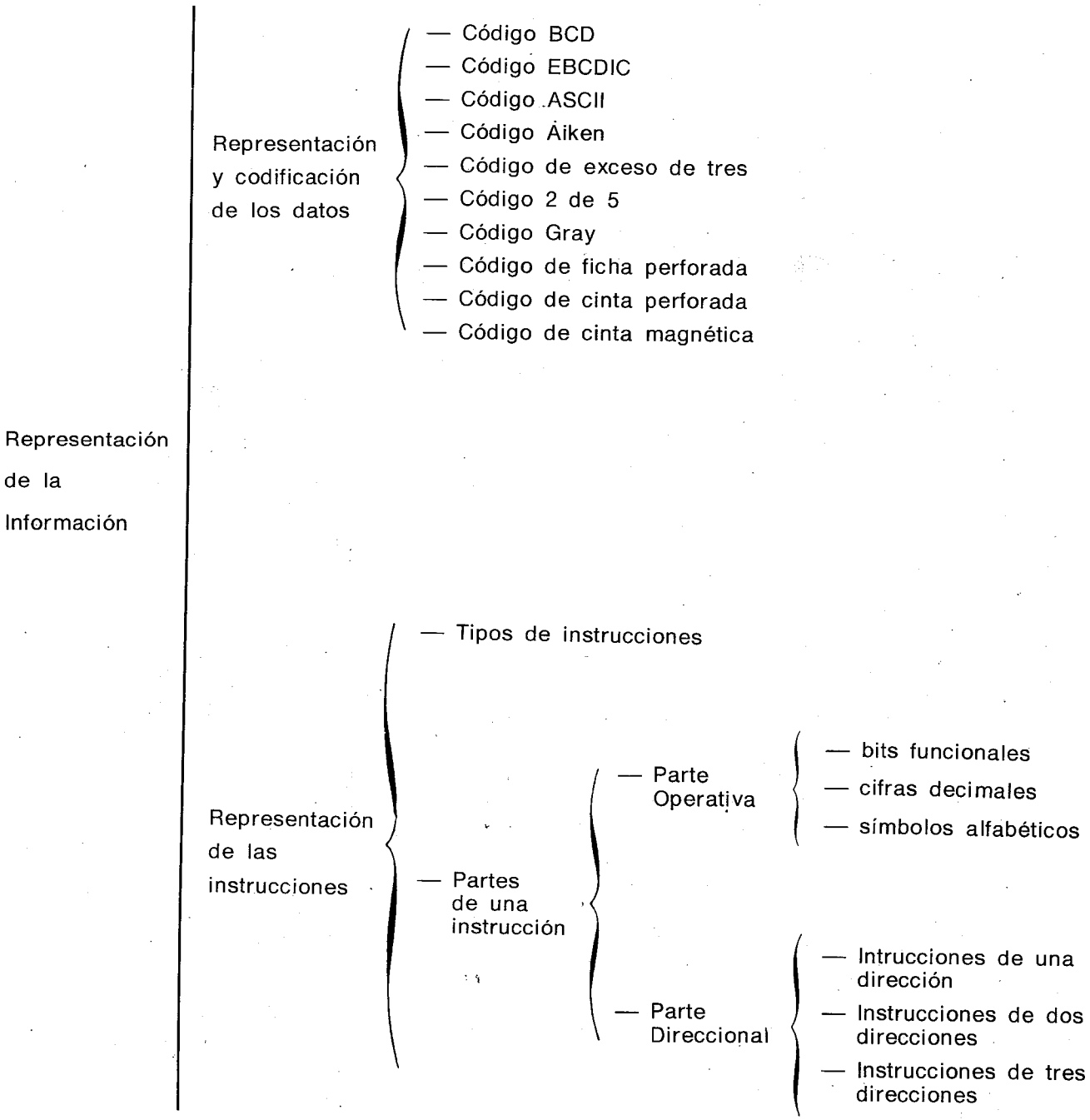
14. Dividir en binario:

    a) 1101010/1110
    b) 10001101/1001
    c) 11011100/101

3.

REPRESENTACION DE LA INFORMACION

CUADRO RESUMEN DE LA LECCION



3. REPRESENTACION DE LA INFORMACION

El tratamiento de la información en un sistema de proceso de datos comprende las funciones de almacenamiento, combinación y transferencia de la información. Para realizar estas funciones es necesario un sistema de representación de la información.

En un sistema de proceso de datos se realizan cuatro operaciones fundamentales:

— leer
— ordenar (clasificar, seleccionar, etc.)
— calcular
— escribir

Estas operaciones están relacionadas con tres tipos de información

— informaciones ordenadas
— informaciones cuantitativas
— informaciones de mando

Las *informaciones ordenadas* son las que sirven para realizar la operación básica de *ordenar* (clasificar, mezclar, elegir, etc.). Por ejemplo, en una empresa, la información sobre un cliente: nombre, número de calle en donde vive, número de orden, número de cuenta corriente, etc.

Las *informaciones cuantitativas* son las que se utilizan para las operaciones de *cálculo*. Por ejemplo: precios de artículos, número de piezas existentes en un almacén, etc.

Las *informaciones de mando* son las que sirven para poder iniciar las cuatro operaciones fundamentales de leer, ordenar, calcular y escribir. Pa-

ra poder iniciar una de estas operaciones es necesario que se dé una orden o una indicación, es decir, se requiere una *instrucción.*

Las instrucciones son informaciones que causan el proceso de los datos.

Las informaciones ordenadas y las informaciones cuantitativas constituyen los *datos.*

Las informaciones de mando constituyen las *instrucciones.*

Datos { Informaciones ordenadas; Informaciones cuantitativas }

Instrucciones ⟶ Informaciones de mando

3.1. REPRESENTACION Y CODIFICACION DE LOS DATOS

La representación de los datos en las diferentes unidades de un sistema de proceso de datos (unidades de entrada, de salida o de almacenamiento) se realiza por medio de señales eléctricas.

Estas señales poseen dos posibles estados, designados por los dígitos binarios 0 y 1.

Por ejemplo, si se produce una señal eléctrica es 1 y en caso contrario 0.

Si queremos representar los diez dígitos del sistema decimal (0, 1, 2 ..., 9), las 26 letras del alfabeto (A, B, C, ..., Z), o los distintos caracteres especiales (., +, −, /, ...) mediante dígitos binarios. Una combinación de este tipo se llama *código* (clave).

El término *codificar* significa «poner en clave» *decodificar* es la operación inversa a la de codificar.

Un ejemplo de código lo constituye la correspondencia entre los dígitos binarios y decimales: $5_{10} = 101_2$ (101 sería el código de 5, es decir, el dígito 5 decimal codificado en binario es 101).

3.1.1. Código BCD

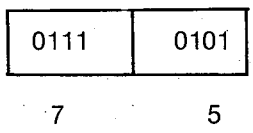
Dado que la mayor parte de los componentes de un sistema de proceso de datos pueden estar en dos estados: conectados o desconectados (*off* u *on*), magnetización en un sentido o en otro, etc. Estos estados pueden representarse por 0 y 1.

Para representar las cifras decimales no es necesario transformarlas en binario sino que sólo se codifican individualmente en binario. Esta forma de representación de los dígitos decimales en binario se denomina *código BCD* (Binary Coded Decimal = Decimal Codificado en Binario, DCB).

En este sistema de representación cada dígito decimal se representa por su equivalente binario, utilizando siempre para cada caso 4 bits.

decimal	*código BCD*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Por ejemplo, para representar el número 75 decimal se utilizan 8 dígitos binarios



este tipo de representación se denomina *representación por tétradas* (4 bits).

Este sistema de codificación puede ampliarse para incluir dígitos, letras y caracteres especiales. Dos de estos códigos BCD ampliados son:

— código EBCDIC
— código ASCII

3.1.2. Código EBCDIC

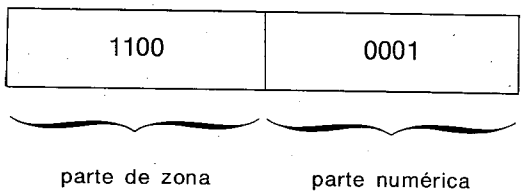
Generalmente los ordenadores emplean palabras con una longitud de 32 bits, es decir, de 4 octetos (bytes) de 8 bits cada uno.

Para representar un *carácter* (dígito, letra, carácter especial, etc.) en estas máquinas se utiliza un byte.

El código EBCDIC (Extended Binary Coded Decimal Interchange Code, Código decimal codificado en binario extendido para intercambio de información) asigna 8 bits a cada carácter.

Si se emplean 8 bits por carácter esto significa que pueden representarse un total de 256 caracteres.

Cada carácter está formado por dos tétradas. Por ejemplo, el carácter A se representa por:



La tétrada de la izquierda se llama *parte de zona* y la tétrada de la derecha se llama *parte numérica*.

La parte de zona puede tener los siguientes valores:

1100: para el grupo de letras de la A a la I.
1101: letras J – R
1110: letras S – Z
1111: cifras 0 – 9

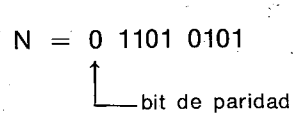
Con el código EBCDIC también se representan los caracteres especiales, los caracteres alfabéticos en minúscula, etc.

Este código es el más utilizado en la mayoría de los sistemas de proceso de datos.

Los datos se almacenan en la memoria central del ordenador en forma de bytes de 8 bits. Sin embargo, a estos 8 bits se le añade un bit adicional

llamado *bit de paridad*. Este bit de paridad se utiliza para detectar errores cuando por un error de la máquina se ha cambiado un 0 por un 1 o un 1 por un 0.

En el código EBCDIC se emplea el bit de paridad impar. Este bit será un 0 o un 1 según sea necesario para que el número total de bits 1 sea impar. Por ejemplo, la letra N es 1101 0101, como el número de bits 1 es 5 (impar) el bit de paridad adicional será 0. Por tanto, el código quedará:



En algunos casos se utiliza el bit de paridad par.

Veamos el código EBCDIC para los dígitos decimales y alfabéticos en binario y en hexadecimal.

Código EBCDIC.

Caracteres numéricos, en binario y en hexadecimal.

Carácter	*EBCDIC (binario)*	*EBCDIC (hexa.)*
0	1111 0000	F0
1	1111 0001	F1
2	1111 0010	F2
3	1111 0011	F3
4	1111 0100	F4
5	1111 0101	F5
6	1111 0110	F6
7	1111 0111	F7
8	1111 1000	F8
9	1111 1001	F9

Código EBCDIC.

Caracteres alfabéticos, en binario y en hexadecimal.

Carácter	*EBCDIC (binario)*	*EBCDIC (hexa.)*
A	1100 0001	C1
B	1100 0010	C2
C	1100 0011	C3
D	1100 0100	C4
E	1100 0101	C5
F	1100 0110	C6
G	1100 0111	C7
H	1100 1000	C8
I	1100 1001	C9
J	1101 0001	D1
K	1101 0010	D2
L	1101 0011	D3
M	1101 0100	D4
N	1101 0101	D5
O	1101 0110	D6
P	1101 0111	D7
Q	1101 1000	D8
R	1101 1001	D9
S	1110 0010	E2
T	1110 0011	E3
U	1110 0100	E4
V	1110 0101	E5
W	1110 0110	E6
X	1110 0111	E7
Y	1110 1000	E8
Z	1110 1001	E9

3.1.3. Código ASCII

El *código ASCII* (American Standard Code for Information Interchange =Código americano stándard para intercambio de la información) es un sistema ampliado de codificar los dígitos decimales en binario:

Carácter	*Código ASCII*
0	0101 0000
1	0101 0001
2	0101 0010
3	0101 0011
4	0101 0100
5	0101 0101
6	0101 0110
7	0101 0111
8	0101 1000
9	0101 1001
A	1010 0001
B	1010 0010
C	1010 0011
.	. .
.	. .
.	. .
I	1010 1001
J	1010 1010
K	1010 1011
.	. .
.	. .
.	. .
R	1011 0010
S	1011 0011
.	. .
.	. .
.	. .
Z	1011 1010

3.1.4. Código Aiken

El código Aiken es un código binario para cifras decimales. Las cifras 0, 1, 2, 3 y 4 se representan por su equivalente binario y las cifras 5, 6, 7, 8 y 9 se representan con los dígitos binarios que corresponderían a los números decimales 11, 12, 13, 14 y 15 (en hexadecimal, B, C, D, E y F).

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

En código Aiken un arrastre decimal produce un arrastre de una tétrada.

Ejemplo

Sea la suma de los números decimales 8 y 4:

```
  8        1110=(8 en código Aiken)
 +4        0100=(4 en código Aiken)
 --
 12
 ↑
 └──  arrastre decimal
```

```
        1110
      + 0100
 ------------
  0001  0010
  ~~~~  ~~~~
    1     2
```

El número 12 (decimal) viene dado por las tétradas: 0001 0010

3.1.5. Código de exceso de tres

El *código de exceso de tres* es un código binario para cifras decimales. Las tétradas de este código no corresponden a la representación binaria de la cifra decimal, sino que la aumenta en tres.

Es decir, cada dígito decimal, N, se representa por un número binario que equivale a N+3.

Una de las características fundamentales del código de exceso de tres es que cuando se produce un arrastre en una suma decimal, se genera un arrastre correspondiente a la suma binaria de representación en código de exceso de tres.

Ejemplo 1

```
  8        1011 = (8 en código de exceso de tres)
 +4        0111 = (4 en código de exceso de tres)
 --
 12
```

```
      1011
     +0111
 ----------
  0001 0010
```

Ahora se suma 3 a las dos tétradas:

```
  0001   0010
  0011   0011
  ----   ----
  0100   0101
  ~~~~   ~~~~
   4 1    5 2
```

Ejemplo 2

Supongamos una suma decimal que no produce arrastre:

5	1000
+2	0101
7	1101

Cuando no existe arrastre se resta 3 para obtener el resultado:

```
 1101
-0011
------
 1010 = 7 en código de exceso de tres
```

Decimal	*Binario*	*Código de exceso de tres*
—	0	—
—	1	—
—	10	—
0	11	0011
1	100	0100
2	101	0101
3	110	0110
4	111	0111
5	1000	1000
6	1001	1001
7	1010	1010
8	1011	1011
9	1100	1100
—	1101	—
—	1110	—
—	1111	—

3.1.6. Código dos de cinco

El *código dos de cinco* es un código binario de cifras decimales que tiene la propiedad de que cada código posee dos dígitos 1 y tres dígitos 0.

Este código tiene la ventaja de que se pueden detectar fácilmente los errores debido a que cada código tiene siempre dos dígitos 1 y tres dígitos 0.

El inconveniente de este código es el tener que utilizar cinco dígitos para representar una cifra, mientras que con los demás códigos se puede representar con cuatro dígitos binarios.

Decimal	*Código dos de cinco*
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

3.1.7. Código Gray

El *código Gray* se utiliza generalmente para transformaciones analógicas-digitales. Esto es debido a que cada cifra sólo se diferencia de la inmediata anterior en una posición binaria.

Este tipo de código no se utiliza en la representación de datos interna en un ordenador, o para calcular, sino que se emplea, por ejemplo, para transmitir datos a distancia a un ordenador central.

Decimal	*Código Gray*
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

3.1.8. Código de ficha perforada

El sistema de codificación utilizado para las fichas perforadas es el *código Hollerith*.

La ficha perforada tiene 80 columnas de 12 posiciones de perforación (12 filas). Un carácter se representa en una columna. Los caracteres numéricos tienen 1 perforación, los alfabéticos 2 perforaciones y los signos especiales 3 perforaciones.

Con la utilización de un código binario, cada perforación de la ficha representa un bit 1 en esa posición y la no perforación representa un bit 0 En la memoria central en lugar de utilizar el código Hollerith se usa el código EBCDIC, ASCII o cualquier otro.

En la ficha perforada de *96 columnas*, en cada columna hay seis posiciones de perforación. Si se utilizan estas seis posiciones para representar un carácter, es posible obtener 64 combinaciones de perforación distintas utilizando el código BCD.

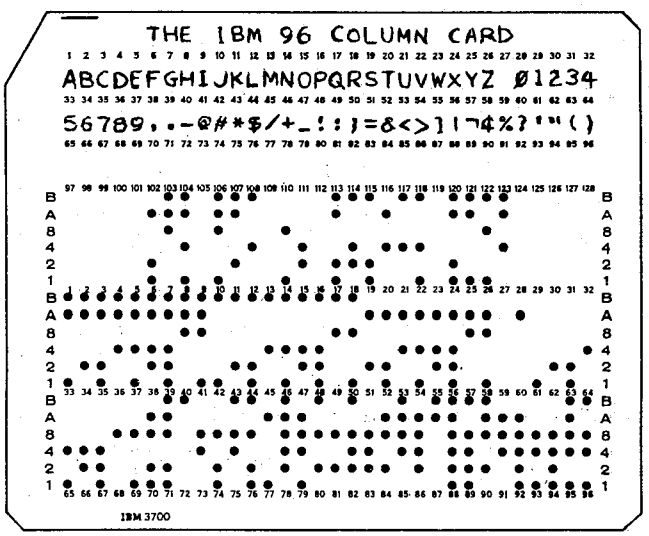


Fig. 3.1. Caracteres perforados en una ficha de 96 columnas.

Carácter	Perforaciones en las filas	Carácter	Perforaciones en las filas
0	0	J	11–1
1	1	K	11–2
2	2	L	11–3
3	3	M	11–4
4	4	N	11–5
5	5	O	11–6
6	6	P	11–7
7	7	Q	11–8
8	8	R	11–9
9	9		
		S	0–2
A	12–1	T	0–3
B	12–2	U	0–4
C	12–3	V	0–5
D	12–4	W	0–6
E	12–5	X	0–7
F	12–6	Y	0–8
G	12–7	Z	0–9
H	12–8		
I	12–9		

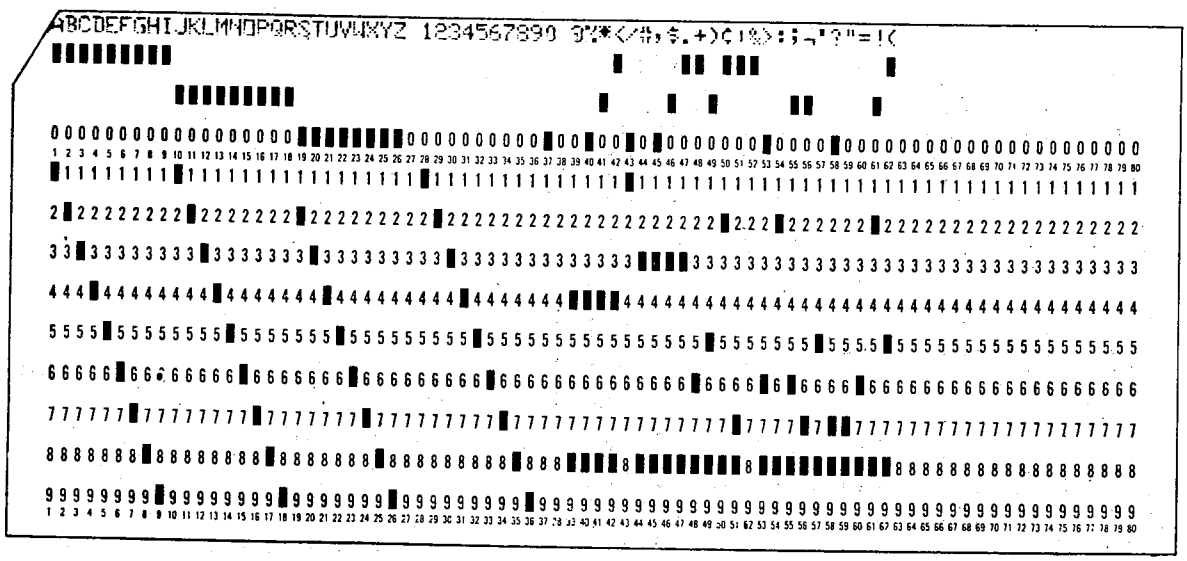


Fig. 3.2. Código Hollerith.

3.1.9. Código de cinta perforada

En las cintas perforadas se utilizan los códigos de 8 y de 5 canales.

Con el *código de 8 canales* cada carácter está representado mediante una combinación de ocho perforaciones o no perforaciones.

Los caracteres numéricos del 0 al 9 se perforan en los cuatro canales inferiores de la cinta, de forma que el canal inferior representa 1 si hay perforación; el segundo canal, 2; el tercero, 4 y el cuarto 8.

Además de estos cuatro canales existe un canal de *verificación* de errores (bit de paridad impar) para cada carácter; un canal de *línea* para indicar el final de un registro; y los dos canales de *zona*

Las cintas de papel además de las perforaciones de código disponen de una fila de perforaciones de arrastre.

La cinta perforada de *cinco canales* utiliza un código de cinco bits sin perforación de verificación.

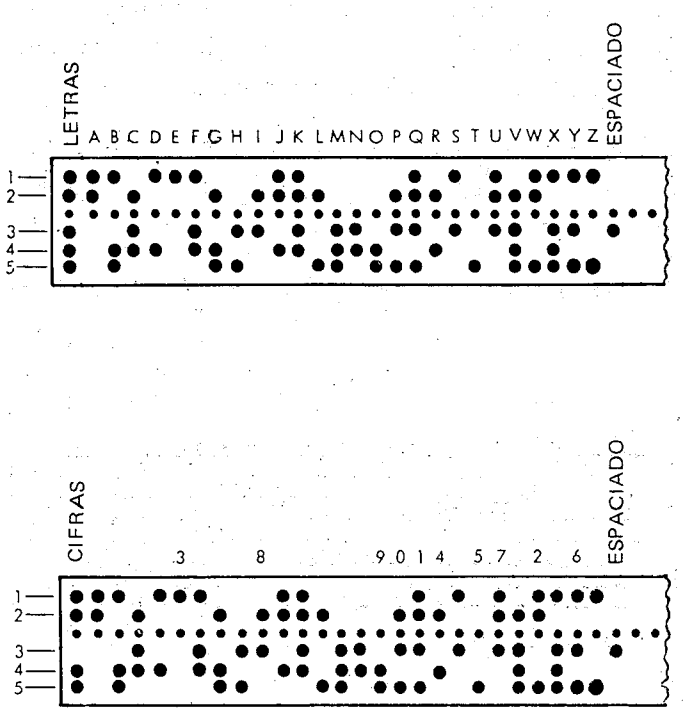


Fig. 3.3. Cinta perforada de 5 canales. Arriba, caracteres alfabéticos. Abajo, caracteres numéricos. Nótese cómo ambas cintas tienen idénticas perforaciones, salvo que las letras van precedidas siempre por una perforación en los canales 1, 2, 3, 4 y 5; mientras que a las cifras les precede una perforación en los canales 1, 2, 4 y 5.

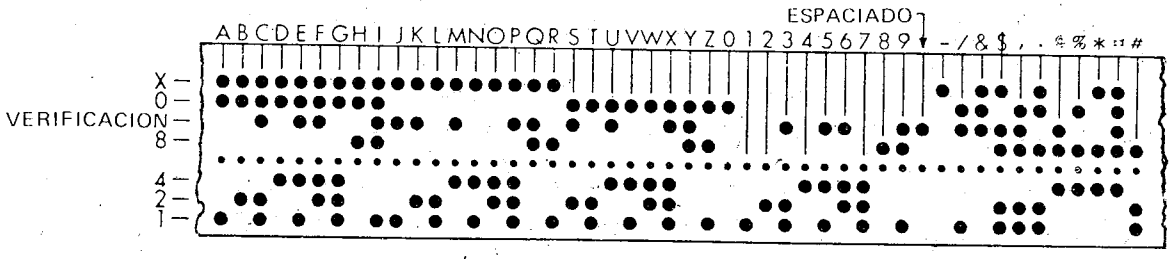


Fig. 3.4. Código de perforación de una cinta de 8 canales.

3.1.10. Código de cinta magnética

En las cintas magnéticas los datos están representados por puntos magnetizados en los canales horizontales de la cinta.

Las cintas están divididas en nueve canales de forma que cada carácter corresponde a una combinación de nueve puntos magnéticos (bits) que pueden ser 1 (punto magnetizado) ó 0 (punto no magnetizado).

De los nueve canales, ocho son para representar el carácter (utilizando el código EBCDIC) y el noveno es el bit de paridad (en las cintas magnéticas se utiliza la paridad par).

En las cintas de 7 canales se utiliza el código BCD (6 bits más un bit de control de paridad).

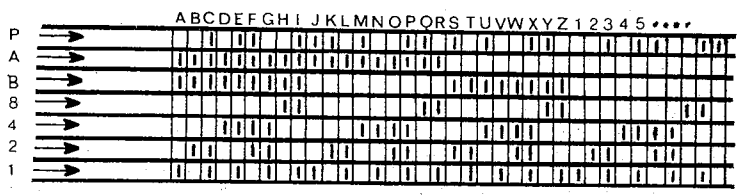


Fig. 3.5. Representación de los caracteres en una cinta magnética de 6 canales más el canal de paridad.

3.2. REPRESENTACION DE LAS INSTRUCCIONES

Para resolver un problema en un sistema de proceso de datos el ordenador se sirve de los *programas*.

Un programa consta en general de dos partes: las instrucciones y los datos.

Los datos pueden introducirse en la memoria del ordenador al inicio del proceso o durante la ejecución del programa.

Las instrucciones pueden almacenarse conjuntamente como una sucesión completa de órdenes.

El almacenamiento de los datos y las instrucciones (programa) nos ha llevado al concepto de sistema de proceso de datos de *programa almacenado*. El programa se almacena en la memoria de una forma contínua, instrucción tras instrucción antes de que se inicie el proceso de ejecución del problema.

La representación física de las instrucciones depende del tipo del ordenador. En algunos ordenadores, una instrucción está representada en una *palabra*, mediante 4 octetos (32 bits), en otros, en *media palabra* (2 octetos) y en otros, una instrucción tienen la longitud de una palabra más la longitud de media palabra, es decir, en 6 octetos.

3.2.1. Tipos de instrucciones

Recordemos que las instrucciones pueden ser las siguientes:

— *Aritméticas:* +, −, ×, /, etc.

— *Lógicas:* — comparativas: > (mayor que)
= (igual que)
< (menor que)
— de salto (bifurcación)

— *De control*: — de entrada
— de salida
— de desplazamiento
— de transferencia
— de parada

Las instrucciones pueden consistir en introducir, o leer datos de la memoria, sumar dos números, comparar dos datos de la memoria, transferir o mover datos de una posición a otra de la memoria, etc.

Estas instrucciones se ejecutan secuencialmente, es decir, una tras otra, en el orden que las ha escrito el programador y en el mismo orden en que han sido almacenadas en la memoria, excepto en el caso de que se trate de una instrucción de bifurcación.

Las *instrucciones de bifurcación* hacen que se deje la secuencia de ejecución cuando se cumplen ciertas condiciones, es decir, producen una ruptura de secuencia.

Cuando se procesa un programa en el ordenador, no se ejecutan todas las instrucciones del programa de una vez, sino que se ejecutan una tras otra: al ejecutarse la primera instrucción, la unidad de control localiza y extrae de la memoria la siguiente instrucción para ejecutarla, y una vez ejecutada esta segunda instrucción se repite de nuevo la misma operación hasta que se procesan todas las instrucciones del programa.

Estas operaciones pueden hacerse de dos modos diferentes: mediante el desarrollo *lineal* del programa o por medio de un programa con *instrucciones de bifurcación*.

En el *proceso lineal* del programa, la unidad de control del ordenador extrae las instrucciones de la memoria, instrucción tras instrucción y se ejecutan en el mismo orden en que fueron almacenadas.

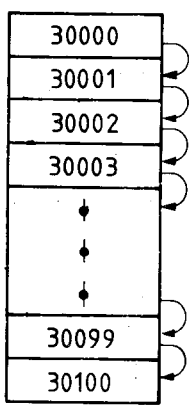


Fig. 3.6. Desarrollo lineal de un programa.

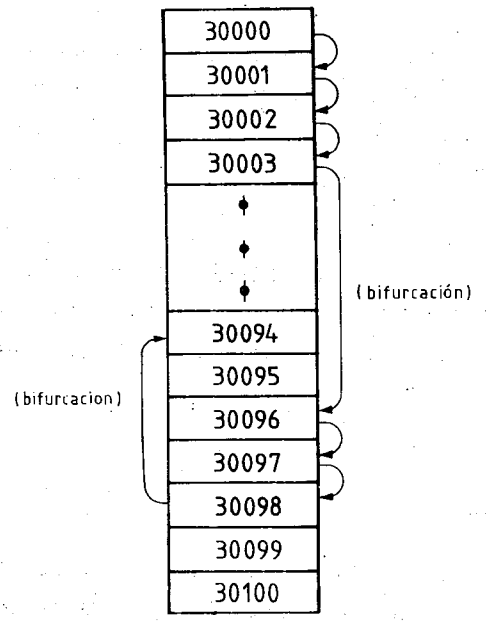


Fig. 3.7. Desarrollo de un programa con instrucciones de bifurcación.

En el proceso de un programa con instrucciones de bifurcación puede romperse la secuencia del programa, de forma que cuando se ejecuta una instrucción de bifurcación puede ejecutarse la siguiente sin seguir el orden en que fueron almacenadas las instrucciones en la memoria del ordenador.

Ejemplo

Supongamos la operación de comparar dos números, A y B. Al efectuar esta comparación se producen dos alternativas:

1) Si A = B, entonces ejecutar la instrucción I1.

2) En caso contrario, si A ≠ B (A distinto de B), entonces ejecutar la instrucción I2.

Comparación	*Switch*	*Instrucción*
A = B	1 (conectado	I1
A ≠ B	0 (desconectado)	I2

En el caso de que A es igual a B, se genera una señal eléctrica que conecta (1) un conmutador o *interruptor* (switch) activándose los circuitos de control para que se ejecute la instrucción I1.

Si A y B son distintos, se desconecta (0) el interruptor y se ejecuta la instrucción I2.

En este ejemplo, si la condición A=B se cumple, se produce una *ruptura de secuencia* (*bifurcación* o abandono de la secuencia del programa) a otro punto del programa, «ejecutar la instrucción I1». Esta sería una instrucción de *bifurcación condicional*. Es decir, se produce una bifurcación cuando se cumple una *condición* determinada.

En otros casos es conveniente abandonar la secuencia de las instrucciones y bifurcar a otro punto del programa sin que sea necesario que se cumpla una determinada condición, para ello se utilizan las instrucciones de *bifurcación incondicional*.

3.2.2. Partes de una instrucción

Una instrucción indica qué tipo de operación (aritmética, lógica, etc.) y el lugar en donde se encuentran los datos que se van a utilizar para poder ejecutarse esta operación.

En general, una instrucción consta de dos partes:

— parte operativa
— parte direccional

La *parte operativa* contiene el *código de operación* que indica el tipo de operación a realizar (suma, resta, comparación, lectura, etc.).

Mediante el código de operación, el ordenador puede interpretar cuál es la operación que debe realizar.

La *parte direccional* indica la *dirección del operando*, es decir, la posición de la memoria en donde se encuentran los datos necesarios para efectuar la operación.

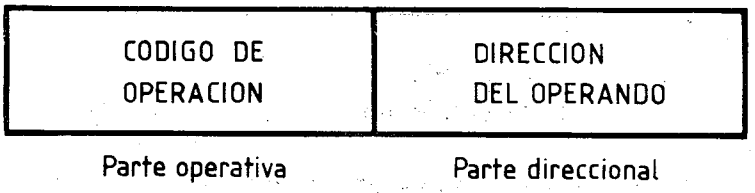


Fig. 3.8. Partes de una instrucción.

Operaciones aritméticas

Cuando el ordenador efectúa una operación de cálculo, necesita generalmente dos operandos. Por ejemplo, si se trata de una suma, necesitará dos sumandos; si es una resta, un minuendo y un sustraendo, etc.

Las operaciones aritméticas las realiza la *unidad aritmética* del ordenador, la cual dispone de unas posiciones o áreas para el almacenamiento temporal de la información que interviene en una operación de cálculo. Estas posiciones se llaman *registros* y suelen tener una longitud de una palabra (32 bits) o de un byte (8 bits), según el tipo de ordenador.

El registro central de la unidad aritmética es el *acumulador* (AC). En este registro se suman los resultados intermedios.

Antes de la ejecución de una instrucción uno de los operandos se almacena en el acumulador, el cual, después de la ejecución de una instrucción aritmética, almacena el resultado.

Ejemplo

En la operación 4+5=9, los dos operandos son 4 y 5.

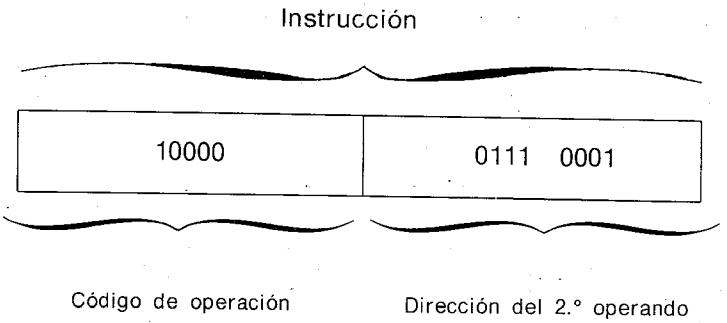
Al empezar la operación, uno de los operandos se encuentra en el acumulador, por ejemplo, el 4.

El otro operando se encuentra en la memoria.

La operación es la siguiente:

«Sumar 5 (extraído de la memoria) al contenido del acumulador (4). El resultado (9) quedará en el acumulador.

La instrucción para efectuar esta operación puede ser la siguiente:



En algunos sistemas, la palabra ADD es el código de la operación *sumar*. Este código se traduce (*decodifica*) al código de operación 10000 (en binario) para que el ordenador pueda interpretarlo.

Los bits 0111 0001 de la parte direccional indican la dirección del segundo operando (5).

Parte operativa

La parte operativa de una instrucción puede estar representada de tres formas, mediante:

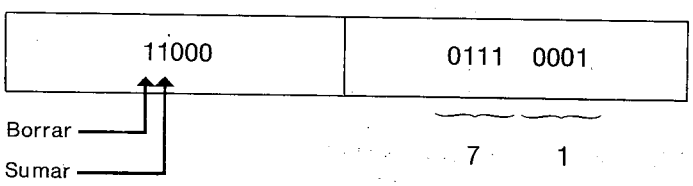
— bits funcionales
— cifras decimales
— símbolos alfabéticos

Bits funcionales

Si la parte operativa está formada por bits, cada uno de ellos significa una operación a realizar.

Ejemplo

Sea la instrucción



El primer bit puede significar: borrar el acumulador.

El segundo bit: sumar al acumulador el contenido de la posición 71.

Cifras decimales

Si se utilizan cifras decimales en la parte operativa pueden codificarse en binario formando tétradas.

Ejemplo:

	Parte operativa	Parte ~~decimal~~ direccional
Sea la instrucción	5 7	1 5 0
esta instrucción quedaría:	0101 0111	0001 0101 000

Símbolos alfabéticos

Pueden utilizarse letras para la parte operativa de la instrucción que han de ser representadas en signos binarios.

B P	1 5 0

Parte direccional

Según la parte direccional, las instrucciones pueden ser tres tipos:

— instrucciones de una dirección
— instrucciones de dos direcciones
— instrucciones de tres direcciones

Instrucciones de una dirección

En las instrucciones de una dirección se indica la dirección del segundo operando. El primer operando se encuentra en el acumulador.

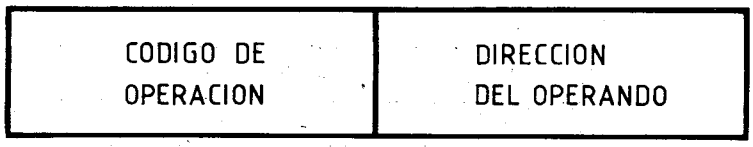


Fig. 3.9. Instrucción de una dirección.

Ejemplo:

La instrucción

SUMAR	1 5 0

significa sumar el contenido de la posición de memoria 150 al contenido del acumulador.

CODIGO DE OPERACION	DIRECCION DEL OPERANDO 1	DIRECCION DEL OPERANDO 2

Fig. 3.10. Instrucción de dos direcciones.

CODIGO DE OPERACION	DIRECCION DEL OPERANDO 1	DIRECCION DEL OPERANDO 2	DIRECCION DEL RESULTADO

Fig. 3.11 Instrucción de tres direcciones.

Instrucciones de dos direcciones

Las instrucciones de dos direcciones son de la forma que muestra la figura 3.10.

En este tipo de instrucciones, se indican las direcciones de los dos operandos que intervienen en la operación de cálculo.

Ejemplo:

SUMAR	1 5 0	2 7 0

Esta instrucción significa: sumar el contenido de la posición 150 al contenido de la posición 270, y el resultado almacenarlo en el acumulador.

Instrucciones de tres direcciones

En estas instrucciones se indican tres direcciones; su representación gráfica puede verse en la figura 3.11.

— la dirección del operando 1
— la dirección del operando 2
— la dirección del operando 3

Ejemplo

SUMAR	1 5 0	2 7 0	3 4 5

Esta instrucción significa: sumar el contenido de la posición 150 al de la posición 270 y el resultado almacenarlo en la posición 345.

EJERCICIOS

1. La mayor parte de los sistemas de proceso de datos utilizan para la representación interna de los datos el sistema:

   a) la representación decimal
   b) la representación octal
   X *c*) la representación binaria

2. En la mayoría de los códigos las cifras y las letras se representan por tétradas que equivalen a:

   a) 3 bits
   X *b*) 4 bits
   c) 8 bits

3. Sean los siguientes números binarios:

   1001, 0010 y 1100.

   ¿Cuáles serían los bits que deben añadirse para el control de paridad impar?

		bit de paridad
X *a*)	1001	1
	0010	0
	1100	1
b)	1001	0
	0010	1
	1100	0
c)	1001	0
	0010	1
	1100	1

4. La operación de clasificar corresponde al tipo de:

   a) información ordenada
   b) información cuantitativa
   c) información de mando

5. La cifra decimal 3 representada con el código BCD es:

   a) 11
   b) 011
   c) 0011

6. La cifra decimal 3 representada en el código EBCDIC es:

   a) 0011
   b) 0000 0011
   c) 1111 0011

7. La cifra decimal 6 representada en código *exceso de tres* es:

   a) 110
   b) 1001
   c) 0110

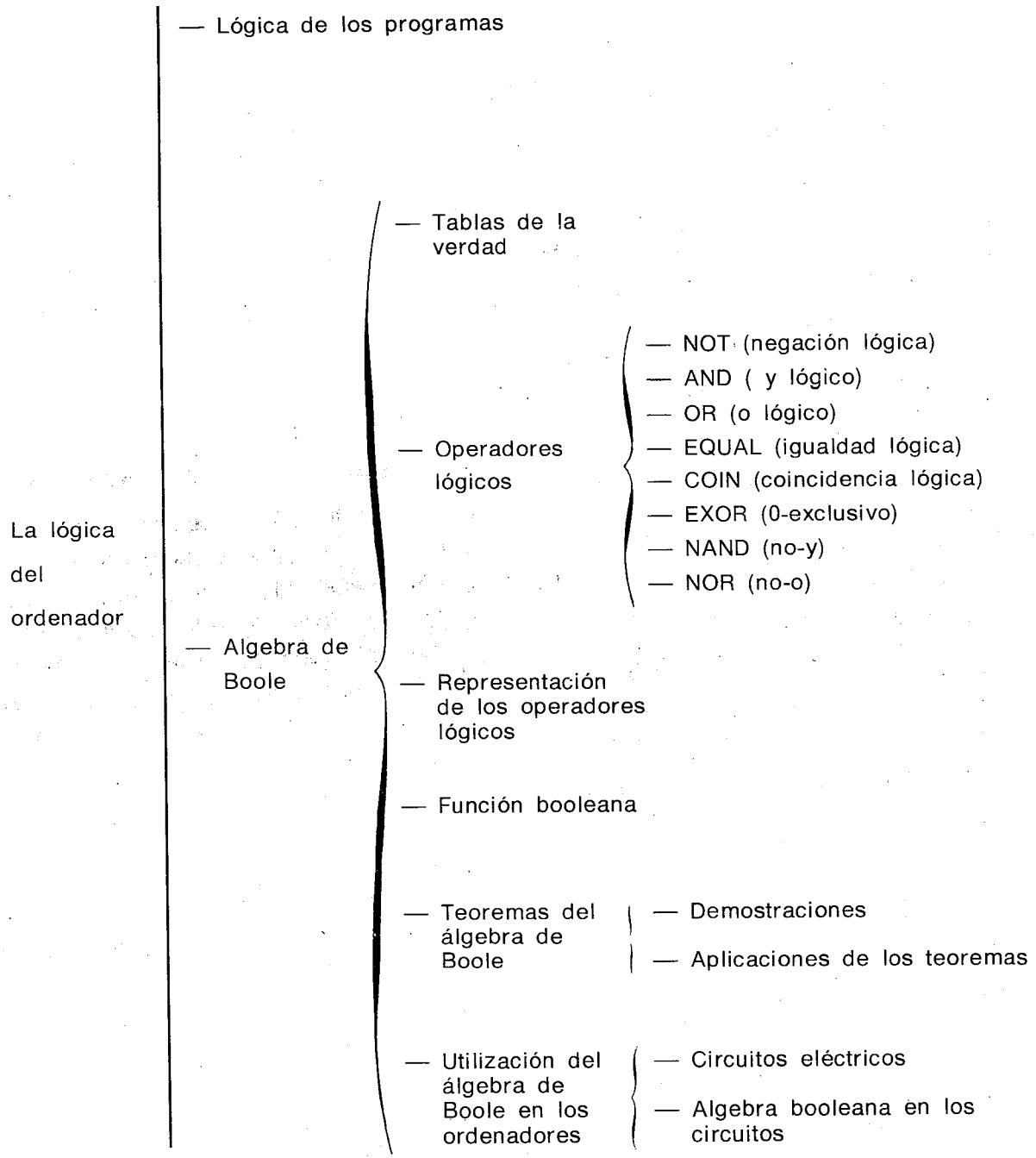
8. ¿Qué es una instrucción de bifurcación?

   a) Es un mecanismo para realizar las operaciones de cálculo.
   b) Una decisión binaria entre dos alternativas que se produce durante el proceso del programa.
   c) Una instrucción para transferir datos a o desde la memoria.

4.

LA LOGICA DEL ORDENADOR

CUADRO RESUMEN DE LA LECCION



4.1. LOGICA DE LOS PROGRAMAS

El hombre es capaz de percibir información (un objeto o idea), formarse una idea de esta información, observar, razonar, clasificar sus observaciones o sus informaciones, realizar operaciones con esas informaciones para llegar a unas conclusiones o resultados

Para llegar a estos resultados se han de establecer unas normas o reglas. El estudio de estas reglas es la *Lógica*.

A los ordenadores se les llama a menudo «cerebros electrónicos» o «máquinas pensadoras» ya que pueden realizar secuencias de operaciones lógicas y aritméticas. El ordenador efectúa estas operaciones con gran velocidad y exactitud y, además, es capaz de tomar decisiones durante la ejecución del programa. Pero estas decisiones no son tomadas por iniciativa propia del ordenador, sino que es el hombre, que lo maneja y quien indica al ordenador, por medio del programa, qué tipo de operaciones debe realizar, en qué debe hacerlo y cuáles son las decisiones que ha de tomar el ordenador.

El ordenador es una máquina electrónica de múltiples aplicaciones con la que se puede obtener la solución de un determinado problema técnico-lógico, pero que no posee la capacidad humana de pensar puesto que tiene que ser programada por el hombre para que pueda realizar su trabajo. Por lo tanto, el programa ha de estar confeccionado de tal forma que los resultados que se exijan sean totalmente válidos y completos. Para ello los datos e instrucciones del programa han de estar organizados de una manera lógica.

Por ejemplo, supongamos un programa para calcular los salarios de los empleados de una determinada empresa. A cada empleado se le aña-

de a su sueldo una cierta cantidad, X, si lleva trabajando en la empresa un tiempo superior a un año. En el programa aparecerán las siguientes condiciones:

a) la antigüedad es superior a un año.
b) la antigüedad *no* es superior a un año.

El programa debe especificar detalladamente las operaciones que ha de realizar el ordenador y establecer unos criterios de comparación para que el ordenador pueda ejecutar unas instrucciones u otras dependiendo de si se cumple una determinada condición o no. Por ejemplo:

1. ¿Antigüedad mayor que 1 año?
   a) *SI*, añadir X a sueldo. Después continuar con el punto 2.
   b) *NO*, entonces continuar con el punto 2.
2. ...

Estas decisiones simples son las que ha de tomar el ordenador. En general, el ordenador se utiliza en trabajos de gran exactitud y muy repetitivos, sin que exista la posibilidad de incertidumbre ni riesgo. Mientras que en situaciones de incertidumbre y riesgo es el hombre quien ha de decidir el tipo de proceso a seguir para resolver esas situaciones.

4.2. ALGEBRA DE BOOLE

El diseño de ciertos elementos básicos de las unidades aritméticas y de control del ordenador está basado en el *álgebra de Boole* o *álgebra lógica*.

El álgebra booleana fue desarrollada en 1854 por el matemático inglés George Boole, y en 1938 Calude Shannon ideó el modo de aplicarla al diseño de los circuitos de los ordenadores electrónicos.

En los ordenadores digitales se utilizan mecanismos y circuitos cuyos estados están representados por dos valores. En el álgebra de Boole sólo existen dos valores, por lo tanto, una *variable booleana* es una cantidad que puede tener uno de los dos valores posibles, es decir, es una *variable binaria*. Estos dos valores son 1 («verdadero») y 0 («falso») y se llaman *constantes lógicas*.

La Lógica matemática, ideada por Aristóteles hace dos mil trescientos años aproximadamente, estudia las relaciones y la validez de las sentencias teniendo en cuenta que las sentencias o son «verdaderas» o son «falsas». Por ejemplo, consideremos las siguientes sentencias:

Sentencia	*Valor*
a) Una semana tiene 8 días	Falsa
b) El Sol es mayor que la Tierra	Verdadera
c) La Luna es mayor que la Tierra	Falsa
d) Una semana tiene 7 días	Verdadera

Estas sentencias no binarias se pueden representar mediante otras cuatro sentencias binarias («verdadero» o «falso»).

Con las sentencias anteriores se pueden componer otras sentencias como por ejemplo:

Sentencia	*Valor*
a) Una semana tiene 8 días y el Sol es mayor que la Tierra.	Falsa
b) Una semana *no* tiene 8 días y el Sol es mayor que la Tierra.	Verdadera
c) Una semana tiene 8 días o el Sol es mayor que la Tierra.	Verdadera
d) Una semana tiene 8 días o el Sol *no* es mayor que la Tierra.	Falsa
e) Una semana *no* tiene 8 días y el Sol *no* es mayor que la Tierra.	Falsa
f) Una semana *no* tiene 8 días *o* el Sol *no* es mayor que la Tierra.	Verdadera

En estos ejemplos se puede comprobar que el valor de las sentencias depende de las palabras *y*, *o* y *no*. Estas sentencias están *compuestas* por otras

sentencias *simples*. La palabra *y* implica que para que la sentencia compuesta sea verdadera han de ser verdaderas las sentencias simples que la componen. La palabra *no* cambia el valor de una setencia simple por su valor opuesto (*complemento*).

Supongamos que representamos las sentencias simples por las letras *A* y *B*, tendremos entonces las reglas siguientes (teniendo en cuenta que la sentencia compuesta se representa por *A y B* o *A o B*):

a) *A y B* es verdadera si y solamente si *A* es verdadera y *B* es verdadera.

b) *A o B* es verdadera si *A* es verdadera o *B* es verdadera, o las dos, *A* y *B*, son verdaderas.

c) *no A* es verdadera si *A* es falsa, y, por tanto, es falsa si *A* es verdadera.

Para simplificar, utilizaremos los siguientes símbolos:

Símbolos	*Representan*
A, B, C, ...	Las sentencias simples
$*$	La palabra *y*
$+$	La palabra *o*
$\bar{A}$, $\bar{B}$, $\bar{C}$, ...	no A, no B, no C, ...
1	verdadero
0	falso

4.2.1. Tablas de la verdad

Una tabla de la verdad contiene el valor (verdader o falso) de los elementos lógicos de entrada (input) y los resultados de la salida (output). Es decir, cada tabla da los valores lógicos de las variables booleanas de entrada y de las variables de salida.

Ejemplo.

Sean *A* y *B* las siguientes sentencias simples:

A = la Tierra es redonda = 1 (verdadero)

B = la nieve es blanca = 1 (verdadero)

Siendo sus complementos $\overline{A}$ y $\overline{B}$:

$\overline{A}$ = la Tierra *no* es redonda = 0 (falso)

$\overline{B}$ = la nieve *no* es blanca = 0 (falso)

Es decir, A puede tener dos valores: 1 y 0, cuando $A = 1$ es verdadero y cuando $A = 0$ es falso, siendo lo mismo para B.

A	*B*	*A y B*
F	F	F
F	V	F
V	F	F
V	V	V

En esta tabla se puede observar que solamente A y B es verdadero cuando A es verdadero y B es verdadero, es decir,

A *y* B = la Tierra no es redonda *y* la nieve no es blanca = falso

A *y* B = la Tierra no es redonda y la nieve es blanca = falso

A *y* B = la Tierra es redonda *y* la nieve no es blanca = falso

A *y* B = la Tierra es redonda *y* la nieve es blanca = verdadero

en los otros tres casos A y B es falso.

Si se sustituyen las palabras *verdadero*, *falso* e *y* por los símbolos 1, 0 y $\times$, se tendrá la siguiente tabla de la verdad:

A	*B*	*A y B*
0	0	0
0	1	0
1	0	0
1	1	1

En el caso de *A o B* la tabla sería la siguiente:

A	*B*	*A o B*
F	F	F
F	V	V
V	F	V
V	V	V

es decir, cuando *A* es verdadero o *B* es verdadero o ambos son verdaderos resulta *A o B* verdadero.

Sustituyendo las palabras *verdadero*, *falso* y *o* por 1, 0 y +, se tiene:

A	*B*	*A+B*
0	0	0
0	1	1
1	0	1
1	1	1

4.2.2. Operadores lógicos

Las palabras *y*, *o* y *no* se llaman *operadores lógicos* y se conocen por las palabras AND, OR y NOT (traducción inglesa de *y*, *o* y *no*).

Los operadores lógicos del álgebra booleana son los siguientes:

Nombre	*Símbolo*	*Ejemplo*	
Negación lógica (NOT)	—	$\bar{A}$	(no A)
Y lógico (AND)	*	A*B	(A y B)
O inclusivo (OR)	+	A+B	(A o B)

Existen otros operadores lógicos, algunos de ellos derivados de los tres anteriores. Estos operadores son: Igualdad lógica (EQUAL), Coincidencia lógica (COIN), O exclusivo (EXOR), NO-Y (NAND) y NO-O (NOR).

Las operaciones con operadores aritméticos dan como resultado un elemento aritmético ($101_2 + 100_2 = 1001_2$). Los resultados de las operaciones lógicas son elementos lógicos: verdadero (1) y falso (0).

Negación lógica (NOT)

La negación lógica (o el complemento lógico) de una variable boolena de entrada, A, es otra variable de salida, $\overline{A}$, de manera que cuando la variable A vale 0, la variable $\overline{A}$ vale 1, y cuando A vale 1, $\overline{A}$ vale 0.

La tabla de la verdad para la negación lógica es la siguiente:

A	$\overline{A}$
0	1
1	0

Por lo tanto, se puede escribir:

$$0 = \overline{1}$$

$$1 = \overline{0}$$

Consideremos el diagrama de Venn de la figura 4.1. La zona rayada de la figura corresponde a la negación lógica, $\overline{A}$ (no A), del conjunto A.

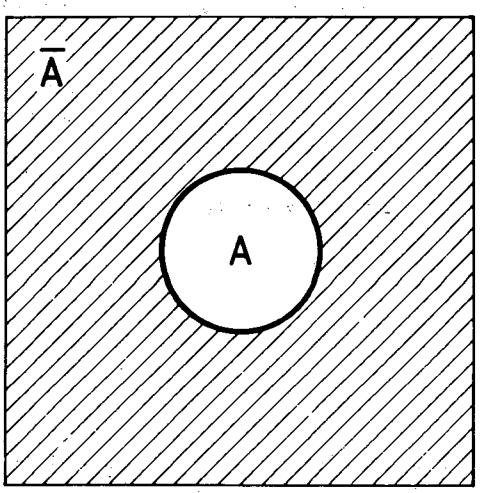


Fig. 4.1. Representación gráfica de la negación lógica ($\overline{A}$).

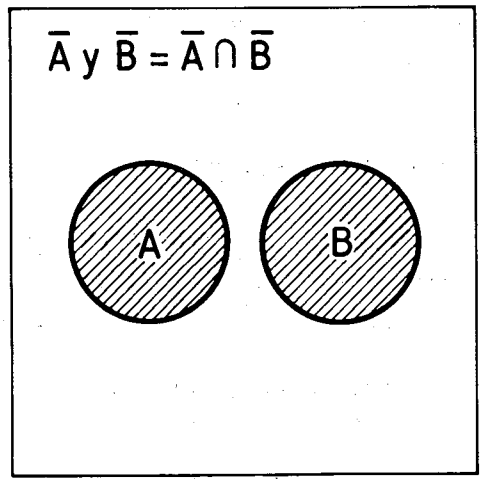


Fig. 4.2. Los elementos que no pertenecen a A *ni a* B *representan la instrucción* ($\bar{A} \cap \bar{B}$). *El símbolo* y *se sustituye por el símbolo de intersección* $\cap$ *(es decir,* $\bar{A}$ y $\bar{B} = \bar{A} \cap \bar{B}$). $\bar{A} \cap \bar{B}$ *corresponde a la zona no rayada.*

Consideremos ahora el diagrama de la figura 4.2.

Este diagrama representa los conjuntos *A* y *B*. La zona rayada de la figura representa la *unión* de los conjuntos:

$$A \cup B$$

es decir, los elementos de A más (*0*) los elementos de B.

La zona no rayada de la figura 4.2 representa la *intersección* de los complementos (negaciones lógicas) de *A* y de *B*:

$$\bar{A} \cap \bar{B}$$

es decir, los elementos de $\bar{A}$ por (*y*) los elementos de $\bar{B}$.

Y Lógico (AND)

El Y lógico de dos variables booleanas de entrada, *A* y *B*, es otra variable de salida, $A * B$. Si una de las dos variables de entrada vale 0, $A * B$ vale 0. Si las dos variables de entrada, *A* y *B*, valen 1, entonces $A * B$ es igual a 1.

La tabla de la verdad para AND será:

A	*B*	*A* * B
0	0	0
0	1	0
1	0	0
1	1	1

Esta tabla significa que:

$$0 * 0 = 0$$

$$0 * 1 = 0$$

$$1 * 0 = 0$$

$$1 * 1 = 1$$

La relación AND (Y lógico) se llama también *intersección* ya que equivale a la intersección de conjuntos. El diagrama de Venn será el de la figura 4.3.

En la figura 4.3 la zona rayada es la intersección de los conjuntos A y B y corresponde a la relación AND de *A* y *B*.

$$A \cap B = A * B$$

En este caso serán conjuntos de elementos binarios ya que los conjuntos *A* y *B* sólo pueden tener dos valores, 0 y 1.

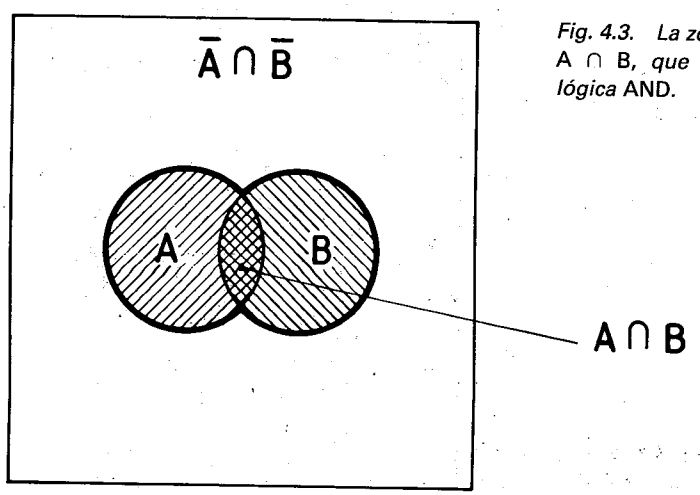


Fig. 4.3. La zona cuadriculada representa a A ∩ B, *que corresponde a la operación lógica* AND.

O Inclusivo (OR)

El O inclusivo lógico de dos variables booleanas de entrada, *A* y *B*, es una nueva variable, $A+B$. Si alguna de las variables, A y B, vale 1 la variable $A+B$ vale 1, y cuando *A* y *B* valen 0, $A+B$ es igual a 0.

A	*B*	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Podemos escribir, entonces:

$$0+0 = 0$$
$$0+1 = 1$$
$$1+0 = 1$$
$$1+1 = 1$$

Nótese que la suma binaria de $1+1$ es igual a 10_2 mientras que en el álgebra de Boole la operación lógica OR de $1+1$ es igual a 1, por lo tanto hay que diferenciar la suma binaria de la operación lógica OR. Sin embargo, los resultados de la multiplicación binaria de dos bits sí corresponden a los resultados de la operación lógica AND.

La relación OR equivale a la *unión* de dos conjuntos de elementos binarios.

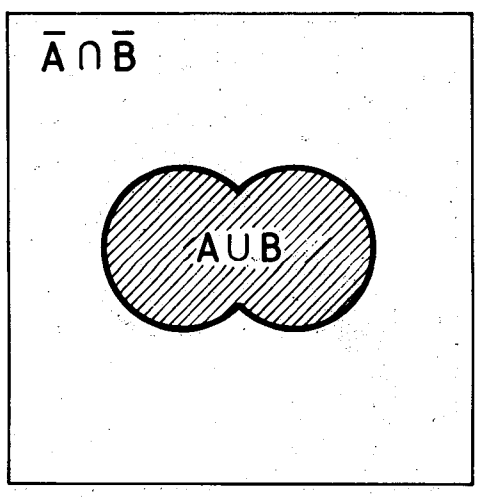


Fig. 4.4. La operación lógica OR es equivalente a la unión *de dos conjuntos* (A ∪ B).

Igualdad lógica (EQUAL)

La igualdad lógica de una variable booleana, *A*, es otra variable, *B*, cuyo valor es siempre igual al de *A*.

La tabla de la verdad será:

A	*B*
0	0
1	1

es decir, A = B.

Coincidencia Lógica (COIN)

La coincidencia lógica de las variables booleanas, *A* y *B* es otra variable A ⊛ B que vale 1 cuando A y B coinciden (o tienen el mismo valor), y vale 0 cuando no coinciden.

A	*B*	A ⊛ B
0	0	1
0	1	0
1	0	0
1	1	1

por lo tanto,

$$0 \circledast 0 = 1$$

$$0 \circledast 1 = 0$$

$$1 \circledast 0 = 0$$

$$1 \circledast 1 = 1$$

El COIN lógico se puede representar por:

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

O Exclusivo (EXOR)

El O exclusivo de dos variables A y B, es otra variable, A⊕B que es igual a 1 cuando *A* y *B* tienen valores distintos, y es 0 cuando *A* y *B* tienen los mismos valores (esta operación es la inversa de la coincidencia lógica).

A	B	$A \oplus B$	
0	0	0	$0 \oplus 0 = 0$
0	1	1	$0 \oplus 1 = 1$
1	0	1	$1 \oplus 0 = 1$
1	1	0	$1 \oplus 1 = 0$

El EXOR se puede representar por la siguiente expresión:

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

NO - Y Lógico (NAND)

El NAND corresponde a la negación lógica de AND, es decir, $\overline{A * B}$.

A	B	$A * B$	$\overline{A * B}$	
0	0	0	1	$\overline{0 * 0} = 1$
0	1	0	1	$\overline{0 * 1} = 1$
1	0	0	1	$\overline{1 * 0} = 1$
1	1	1	0	$\overline{1 * 1} = 0$

NO - O Lógico (NOR)

El NOR corresponde a la negación lógica del OR, $\overline{A + B}$.

A	B	$A + B$	$\overline{A + B}$	
0	0	0	1	$\overline{0 + 0} = 1$
0	1	1	0	$\overline{0 + 1} = 0$
1	0	1	0	$\overline{1 + 0} = 0$
1	1	1	0	$\overline{1 + 1} = 0$



Fig. 4.5. Esquema de la representación por bloques de los operadores lógicos.

4.2.3. Representación de los operadores lógicos

Los operadores lógicos se pueden representar gráficamente, por bloques (Figura 4.5):

En la página siguiente se incluyen las representaciones gráficas, por bloques, de los operadores lógicos estudiados.

4.2.4. Función booleana

Una función booleana de una o más variables muestra la relación lógica de estas variables.

La función booleana constituye también otra variable booleana.

Por ejemplo: sea F la función booleana de la variable A tal que:

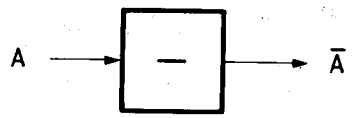
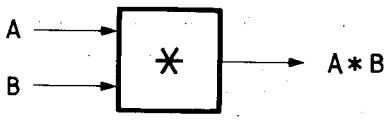
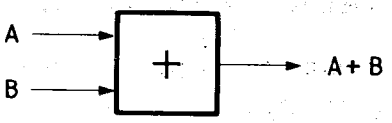
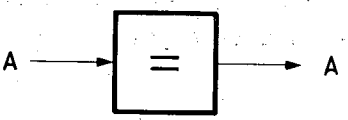
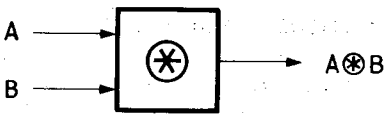
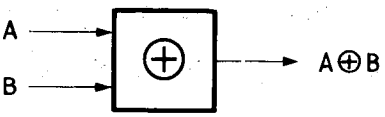
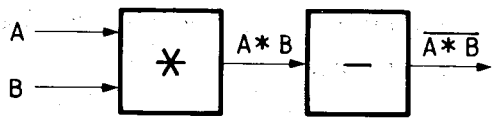
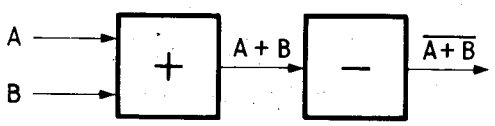
$$F(A) = A + \bar{A}$$

La variable A puede aparecer de dos formas: en forma directa (A) o complementada ($\bar{A}$).

Supongamos ahora que F es la función booleana de las variables booleanas A y B:

$$F(A, B) = A + B$$
o $$F(A, B) = A * B$$
o $$F(A, B) = \bar{A} * B + A * \bar{B}$$ etc.

siendo $F(A, B)$ la función booleana de las variables A y B.

Operadores	Representación por bloques
a) NOT	
b) AND	
c) OR	
d) EQUAL	
e) COIN	
f) EXOR	
g) NAND	
h) NOR	

Ejemplo:

Supongamos la función booleana:

$$F(A,\ B)=\overline{A}*\overline{B}+A*B$$

que representa a la operación lógica COIN (coincidencia lógica)

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

si los valores de A y B son $A=0$ y $B=0$, entonces $F(A,\ B)$ será:

$$F(A,\ B)=\overline{0}*\overline{0}+0*0$$

como $\overline{0}=1$, se tiene

$$F(A,\ B)=1*1+0*0$$

$$F(A,\ B)=1+0$$

$$F(A,\ B)=1$$

Si los valores de A y B son:

$A=1$ y $B=1$, entonces

$$F(A,\ B)=\overline{1}*\overline{1}+1*1$$

$$=0*0+1*1$$

$$=0+1$$

$$=1$$

si $A=0$ y $B=1$ tendremos:

$$F(A,\ B)=\overline{0}*\overline{1}+0*1$$

$$=1*0+0*1$$

$$=0+0$$

$$=0$$

y por último, si $A=1$ y $B=0$, entonces:

$$F(A,\ B)=\overline{1}*\overline{0}+1*0$$

$$=0*1+1*0$$

$$=0+0$$

$$=0$$

A partir de esta función booleana puede construirse la siguiente *tabla de la verdad:*

A	B	F *(A, B)*
0	0	1
0	1	0
1	0	0
1	1	1

Por lo tanto, $F(A, B) = \overline{A} * \overline{B} + A * B$ corresponde a *coincidencia lógica* (COIN).

4.2.5. Teoremas del álgebra de Boole

Los teoremas del álgebra booleana son los siguientes:

NOT

1) $(\overline{\overline{A}}) = A$

la negació d'una negació és una afirmació
No del NO = Sí

OR

2) $A + 0 = A$

3) $A + 1 = 1$

4) $A + \overline{A} = 1$

5) $A + A = A$

6) $\overline{A + B} = \overline{A} * \overline{B}$

7) $A + B = B + A$

8) $A + (B + C) = (A + B) + C$

9) $A + B * C = (A + B) * (A + C)$

10) $A * B + A * \overline{B} = A$

11) $A + A * B = A$

12) $A + \overline{A} * B = A + B$

AND

13) $A * 1 = A$

14) $A * 0 = 0$

15) $A * \overline{A} = 0$

16) $A * A = A$

17) $\overline{A * B} = \overline{A} + \overline{B}$

18) $A * B + A * \overline{B} = A$

19) $A * (B * C) = (A * B) * C$

20) $A * (B + C) = A * B + A * C$

21) $(A + B) * (A + \overline{B}) = A$

22) $A * (A + B) = A$

23) $A * (\overline{A} + B) = A * B$

Demostración de los teoremas

Los teoremas del álgebra booleana pueden dividirse en tres tipos: los teoremas de la negación lógica, NOT, los de la operación OR y los de la operación AND.

NOT

---

1) $\overline{\overline{A}} = A$

La negación lógica de $\overline{A}$, es decir, $\overline{A}$ es A.

Las variables booleanas sólo pueden tener los valores 0 y 1, por lo tanto puede construirse la siguiente tabla de la verdad:

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

OR

---

2) $A+0=A$

A	$A+0$
0	0
1	1

3) $A+1=1$

A	$A+1$
0	1
1	1

---

4) $A+\overline{A}=1$

A	$\overline{A}$	$A+\overline{A}$
0	1	1
1	0	1

5) $A+A=A$

A	$A+A$
0	0
1	1

---

6) $\overline{A+B}=\overline{A}*\overline{B}$

Este teorema junto con el teorema 17 constituyen los llamados *teoremas de Morgan*.

En estos teoremas se sustituyen los operadores + por los operadores *, los operadores * por los operadores +, las variables sin complementar, *A* y *B*, por las variables complementarias, $\overline{A}$ y $\overline{B}$, y las variables complementadas por las variables sin complementar.

La tabla de la verdad será la siguiente:

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}*\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

En esta tabla de la verdad queda demostrado que

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

7) $A+B=B+A$

Las funciones booleanas cumplen la *propiedad conmutativa*: el orden de los términos no altera el resultado.

A	B	$A+B$	$B+A$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

8) $A+(B+C)=(A+B)+C$

Este teorema permite que se utilicen los términos sin paréntesis (*propiedad asociativa*).

A	*B*	*C*	*B+C*	*A+(B+C)*	*A+B*	*(A+B)+C*
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

9) $A+B*C=(A+B)*(A+C)$

Mediante este teorema se pueden desarrollar los términos de una función booleana (propiedad distributiva).

A	*B*	*C*	*A+B*	*A+C*	*B*C*	A+B*C	*(A+B)*XA+C)*
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

10) $A * B + A * \overline{B} = A$

Este teorema permite simplificar las funciones booleanas.

A	B	$\overline{B}$	$A * B$	$A * \overline{B}$	$A * B + A * \overline{B}$
0	0	1	0	0	0
0	1	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1

11) $A + A * B = A$

A	B	$A * B$	$A + A * B$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

12) $A + \overline{A} * B = A + B$

A	B	$\overline{A}$	$\overline{A} * B$	$A + \overline{A} * B$	$A + B$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

AND

13) $A * 1 = A$

A	$A * 1$
0	0
1	1

14) $A * 0 = 0$

A	$A * 0$
0	0
1	0

15) $A * \overline{A} = 0$

A	$\overline{A}$	$A * \overline{A}$
0	1	0
1	0	0

16) $A * A = A$

A	$A * A$
0	0
1	1

17) $\overline{A * B} = \overline{A} + \overline{B}$

Este teorema y el teorema 6 visto anteriormente son los teoremas de Morgan. Estos teoremas se utilizan cuando una función booleana está complementada.

La tabla de la verdad siguiente demuestra este teorema:

A	B	$A * B$	$\overline{A * B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

18) $A * B = B * A$ *(propiedad conmutativa).*

A	*B*	$A * B$	$B * A$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

19) $A * (B * C) = (A * B) * C$ *(propiedad asociativa).*

A	*B*	*C*	$B * C$	$A * B$	$A * (B * C)$	$(A * B) * C$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

20) $A * (B + C) = A * B + A * C$ *(propiedad distributiva).*

A	*B*	*C*	$B + C$	$A * (B + C)$	$A * B$	$A * C$	$A * B + A * C$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

21) $(A+B)*(A+\overline{B})=A$

A	B	$\overline{B}$	$A+B$	$A+\overline{B}$	$(A+B)\otimes(A+\overline{B})$
0	0	1	0	1	0
0	1	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1

22) $A*(A+B)=A$

A	B	$A+B$	$A*(A+B)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

23) $A*(\overline{A}+B)=A*B$

A	B	$\overline{A}$	$\overline{A}+B$	$A*(\overline{A}+B)$	$A*B$
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

Aplicación de los teoremas

Veamos unos ejemplos de funciones booleanas y su desarrollo, utilizando los teoremas del álgebra de boole.

Ejemplo 1:

Obtener el complemento de la función booleana siguiente:

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

la función booleana complementada es:

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

aplicando el teorema de Morgan tendremos

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

según el teorema 1

$$\bar{\bar{B}}=B$$

entonces,

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

es decir, el complemento de la función booleana $A+\bar{B}$ es la función $\bar{A}*B$.

Ejemplo 2:

Demostrar que el complemento de la operación lógica EXOR ($A\oplus B$) es la operación lógica COIN ($A\otimes B$)

$$A\oplus B=A*\bar{B}+\bar{A}*B$$

$$A\otimes B=A*B+\bar{A}*\bar{B}$$

$$\overline{A\oplus B}=\overline{A*\bar{B}+\bar{A}*B}$$

$$=\overline{(A*\bar{B})}*\overline{(\bar{A}*B)}$$

$$=(\bar{A}+\bar{B})*(\bar{A}+\bar{B})$$

$$=(\bar{A}+B)*(A+\bar{B})$$

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

$$=0+B*A+\bar{A}*\bar{B}+0$$

$$=B*A+\bar{A}*\bar{B}=A\otimes B$$

4.2.6. Utilización del álgebra booleana en los ordenadores

En 1938 Shannon demostró que se podía aplicar el álgebra de Boole a los circuitos del ordenador.

El funcionamiento de los núcleos magnéticos, de los transistores y del resto de los componentes de los circuitos del ordenador están basados en la lógica binaria.

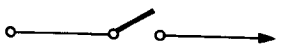
Circuitos eléctricos

Los circuitos eléctricos los representamos gráficamente del siguiente modo:

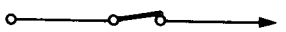


La flecha de la figura indica el sentido de la corriente.

Los circuitos pueden tener uno o varios interruptores (Switch) para su conexión y desconexión, es decir, para que en un momento determinado se pueda interrumpir o activar el paso de la corriente por el circuito.



a) Interruptor abierto

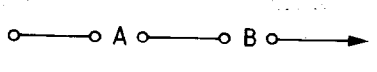


b) Interruptor cerrado

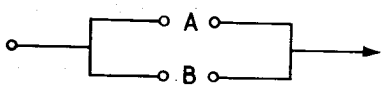
a) Cuando está abierto el interruptor, no pasa la corriente.

b) Cuando el interruptor está cerrado hay circulación de corriente.

Si en un circuito hay varios interruptores, éstos pueden estar colocados en serie o en paralelo, de forma que si están colocados en serie deben estar todos los interruptores cerrados para que circule la corriente y si estan en paralelo es necesario que sólo un interruptor esté cerrado para que la corriente circule.



a) Circuito en serie



b) Circuito en paralelo

a) para que circule la corriente es necesario que *A y B* estén cerrados.

b) para que circule la corriente, *A o B* deben estar cerrados.

En un circuito en serie se pueden presentar las siguientes situaciones:

Interruptor A	*Interruptor B*	*Corriente*
abierto	abierto	no circula
abierto	cerrado	no circula
cerrado	abierto	no circula
cerrado	cerrado	circula

Si sustituimos las palabras «abierto» y «no circula» por 0; «cerrado» y «circula» por 1; y «corriente» por la operación lógica A $\times$ B (AND), tendremos la siguiente tabla:

A	*B*	*A* $\times$ B
0	0	0
0	1	0
1	0	0
1	1	1

En un circuito en paralelo se pueden presentar las situaciones siguientes:

Interruptor A	*Interruptor B*	*Corriente*
abierto	abierto	no circula
abierto	cerrado	circula
cerrado	abierto	circula
cerrado	cerrado	circula

que corresponde a la operación lógica A+B (OR):

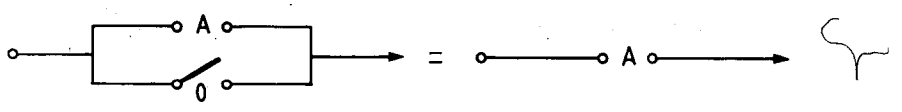
A	*B*	*A*+*B*
0	0	0
0	1	1
1	0	1
1	1	1

El álgebra booleana en los circuitos

Las propiedades de los circuitos son:

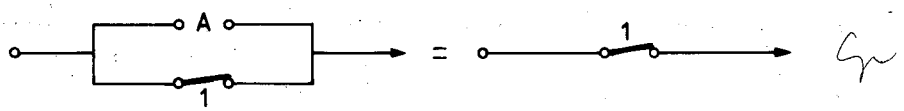
---

1) $A+0=A$



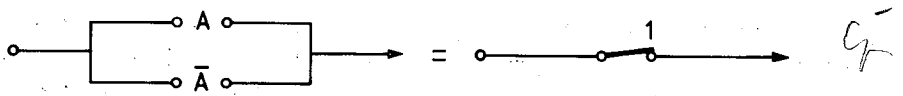
---

2) $A+1=1$



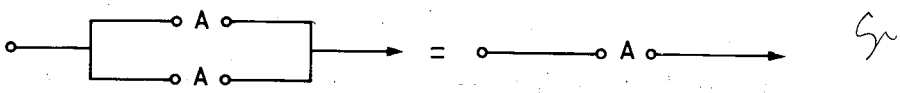
---

3) $A+\overline{A}=1$



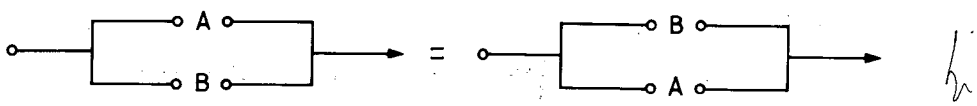
---

4) $A+A=A$



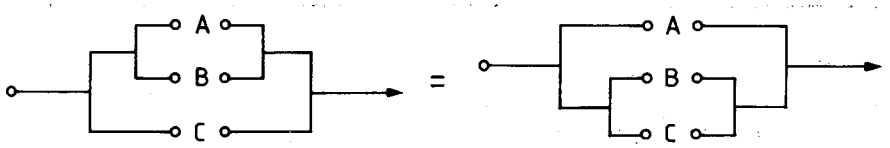
---

5) $A+B=B+A$

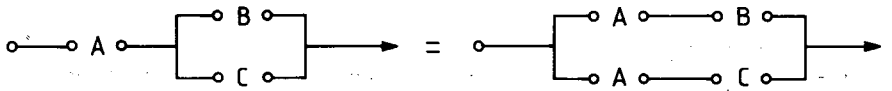


---

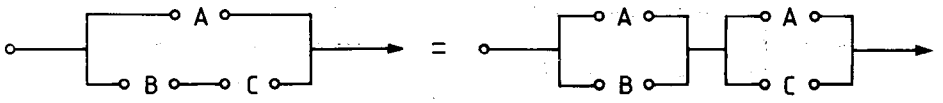
6) $(A+B)+C=A+(B+C)$



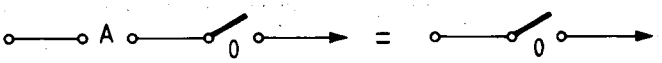
7) $A*(B+C)=A*B+A*C$



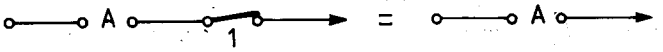
8) $A+B*C=(A+B)*(A+C)$



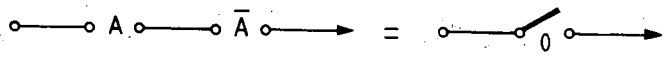
9) $A*0=0$



10) $A*1=A$



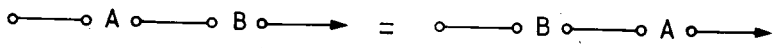
11) $A * \overline{A} = 0$



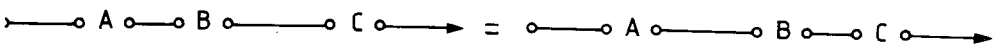
12) $A * A = A$



13) $A * B = B * A$



14) $(A * B) * C = A * (B * C)$

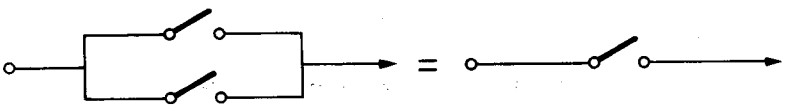


Existen otras propiedades derivadas de las anteriores.

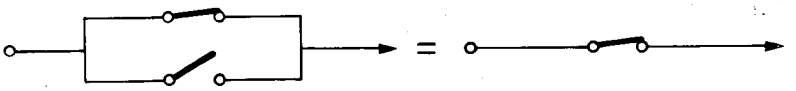
A partir de estos ejemplos es fácil deducir los esquemas de las restantes propiedades de los circuitos.

1) $A+0=A$

$A=0$

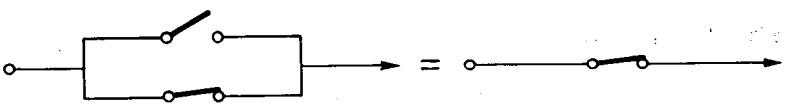


$A=1$

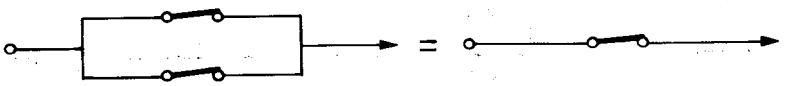


2) $A+1=1$

$A=0$

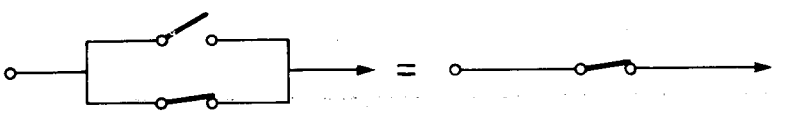


$A=1$

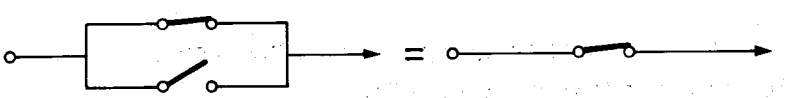


3) $A+\overline{A}=1$

$A=0$

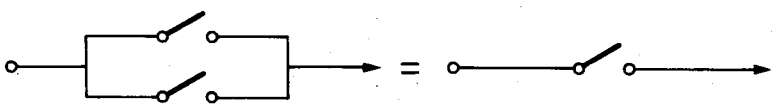


$A=1$

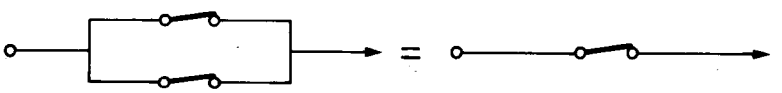


4) $A+A=A$

$A=0$



$A=1$



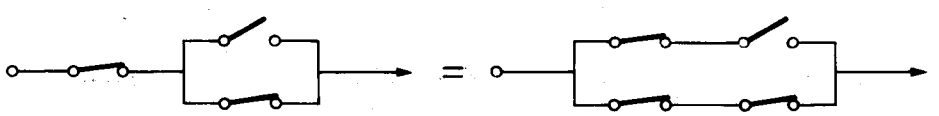
5) $A*(B+C)=A*B+A*C$

Supongamos los valores siguientes para A, B y C:

$A=1$

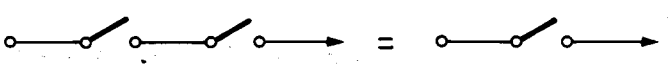
$B=0$

$C=1$

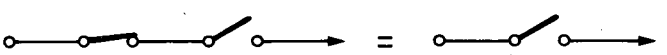


6) $A*0=0$

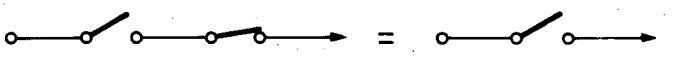
$A=0$

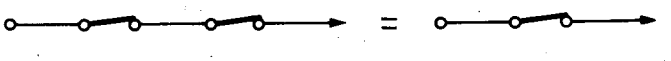


$A=1$

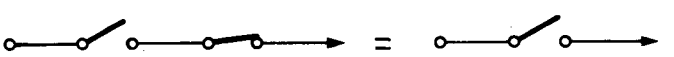


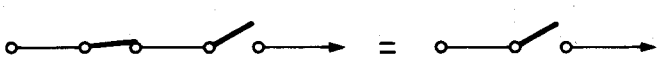
7) $A * 1 = A$

$A = 0$ 

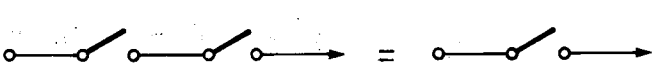
$A = 1$ 

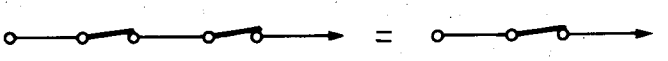
8) $A * \overline{A} = 0$

$A = 0$ 

$A = 1$ 

9) $A * A = A$

$A = 0$ 

$A = 1$ 

EJERCICIOS

1. Los ordenadores poseen la facultad de pensar como un ser humano:

   a) verdadero
   b) falso

2. El programa está compuesto por una serie de instrucciones ordenadas y enlazadas de una forma lógica.

   a) verdadero
   b) falso

3. El ordenador puede tomar decisiones por iniciativa propia.

   a) verdadero
   b) falso

4. Las funciones del ordenador se caracterizan principalmente por su incertidumbre y riesgo.

   a) verdadero
   b) falso

5. El programa debe especificar detalladamente las operaciones que va a realizar el ordenador.

   a) verdadero
   b) falso

6. Construir la tabla de la verdad de la operación lógica COIN, teniendo en cuenta que:

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

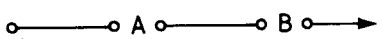
7. Construir la tabla de la verdad de la operación lógica EXOR, teniendo en cuenta que:

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

8. Efectuar la representación por bloques de la operación lógica COIN, teniendo en cuenta la fórmula anterior.

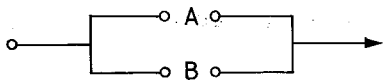
9. El mismo ejercicio para la operación lógica EXOR.

10. Dado el circuito de la figura.



a) ¿En qué condiciones circulará la corriente?
b) ¿A qué operación lógica corresponde?

11. El mismo ejercicio anterior para el circuito:



12. Construir la tabla binaria correspondiente a tres interruptores en serie.

13. Construir la tabla binaria para tres interruptores en paralelo.

14. Calcular el valor de las siguientes funciones booleanas:

a) $1+0+0*1$ 1
b) $1*(0+1*0+0)+1$ 1
c) $0*(1+1*0+1)$ 0
d) $1+1*1+0*1$ 1

SOLUCIONES DE LOS EJERCICIOS

1. Conceptos matemáticos básicos

1. *a*) $\frac{1}{2}$; *b*) 3; *c*) 5; *d*) 128; *e*) 13824; *f*) 4; *g*) 0; *h*) 7; *i*) 35

2. *a*) $A \cup B = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

   b) $A \cap B = \{3, 6\}$

3. *a*) $\bar{A} \cap B = \{5, 6, 7\}$

   b) $A \cap \bar{B} \cap C = \{2\}$

   c) $A \cap B \cap C = \{\varnothing'\}$

   d) $A \cup B \cup C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} = R$

   $\bar{A} \cup A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} = R$

   $\bar{A} \cap A = \{\varnothing\}$

2. Sistemas de numeración

1. *a*) 78; *b*) 57; *c*) −63; *d*) 143; *e*) 488; *f*) 526.

2. *a*) 100110; *b*) 100000011; *c*) 10010110010; *d*) 1111111; *e*) 1000000000000; *f*) 1000001101100

3. a) 343; *b*) 2110; *c*) 16392

4. a) 45226; *b*) 10000; *c*) 1147

5. a) 41227; *b*) 225485; *c*) 64158

6. a) CCE; *b*) 49FD; *c*) 138AD

7. a) 101010111; *b*) 11110000; c) 1101000100

8. a) 47; b) 146; c) 1065

9. a) 101011000001; *b*) 10010011000; c) 11000100001111

10. a) 1B5; *b*) 3D6; *c*) 5A2

11. a) 1010000100; *b*) 1111010001; *c*) 11011001

12. a) 100101110; *b*) −1010; *c*) 111011.01

13. a) 1110110010; *b*) 1010000010; *c*) 11000.100101

14. a) cociente = 111, resto = 1000; *b*) cociente = 1111, resto = 110; *c*) 101100.

3. Representación de la información

1. c); *2.* b); *3.* a); *4.* a); *5.* c); *6.* c); *7.* b); *8.* a); *9.* b).

4. La lógica del ordenador

1. *b*; *2.* *a*; *3.* *b*; *4.* *b*; *5.* *a*.

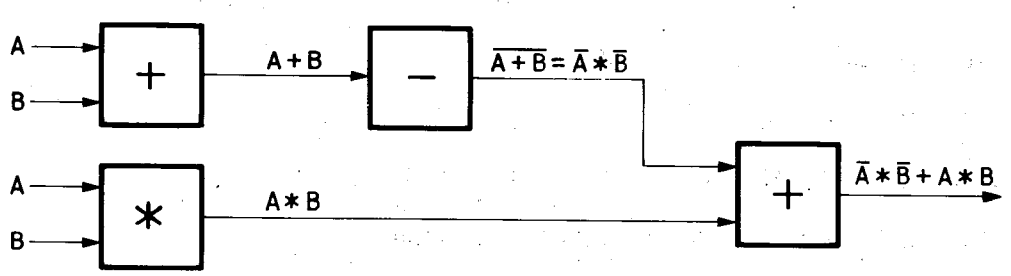
6.

A	B	$\overline{A}$	$\overline{B}$	$A * B$	$\overline{A} * \overline{B}$	$\overline{A} * B + A * B$
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

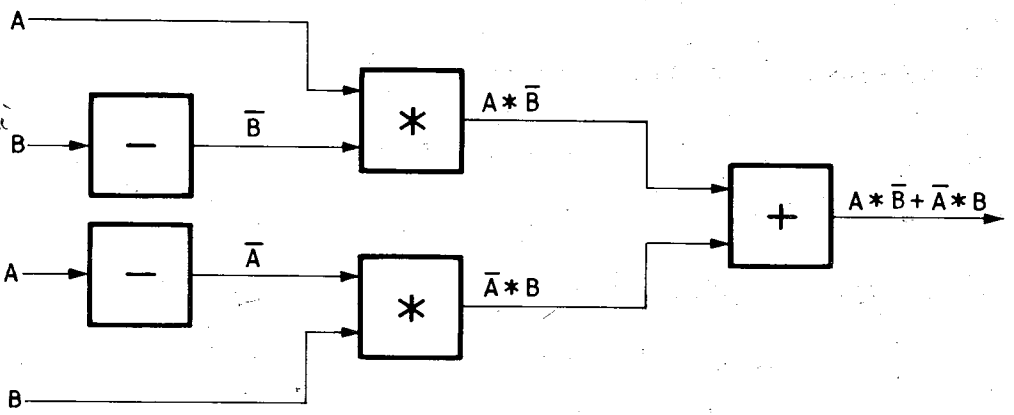
7.

A	B	$\overline{A}$	$\overline{B}$	$A \times \overline{B}$	$\overline{A} \times B$	$\overline{A} \times \overline{B} + A \times B$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

8.



9.



10. *a*) Si los dos interruptores están cerrados: A = 1 y B = 1.

b) $A \times B$

11. *a*) Si uno de los interruptores está cerrado, es decir, si A = 1 o B = 1.

b) A + B

12.

A	*B*	*C*	$A * B * C$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

13.

A	*B*	*C*	$A+B+C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

14. *a*) 1; *b*) 1; *c*) 0; *d*) 1.

Indice General

Contenido 7

Introducción 9

1. CONCEPTOS MATEMATICOS BASICOS

Cuadro resumen 13
1. Conceptos matemáticos básicos 14
1.1. Potencia 14
1.1.1. Reglas de la potenciación 16
1.1.2. Formas de potencia 18
1.1.3. Aplicaciones 18
1.2. Logaritmos 19
1.2.1. Definición 19
1.2.2. Reglas de los logaritmos 20
1.2.3. Cambio de base 21
1.3. Factorial 21
1.4. Coeficiente binómico 22
1.5. Conjuntos 23
1.5.1. Conjunto unidad 24
1.5.2. Conjunto vacío 24
1.5.3. Conjunto universal 24
1.5.4. Subconjuntos 25
1.5.5. Igualdad 27

1.5.6. Conjunto de las partes 28
1.5.7. Complementos 28
– Complemento del conjunto universal 31
– Complemento de un complemento 31
1.5.8. Unión 32
1.5.9. Intersección 33

Ejercicios 36

2. SISTEMAS DE NUMERACION

Cuadro resumen 39
2. Sistemas de numeración 40
2.1. Sistema decimal 40
2.1.1. Números reales 40
2.1.2. Números complejos 43
2.1.3. Representación de los números decimales 43
2.2. Sistema binario 44
2.2.1. Conversión de binario a decimal 45
2.2.2. Conversión de decimal a binario 46
2.2.3. Aritmética binaria 47
– Adición 48
– Sustracción 52
– Complementos de los números binarios 56
– Sustracción utilizando el complemento a 1 58
– Sustracción utilizando el complemento a 2 62
– Multiplicación binaria 65
– División binaria 67
2.3. Sistema octal 69
2.3.1. Conversión de octal a decimal 69
2.3.2. Conversión de decimal a octal 69
2.3.3. Conversión de octal a binario 71
2.3.4. Conversión de binario a octal 72
2.3.5. Aritmética octal 72
2.4. Sistema hexadecimal 73
2.4.1. Conversión hexadecimal a decimal 73
2.4.2. Conversión de decimal a hexadecimal 74
2.4.3. Conversión de hexadecimal a binario 75
2.4.4. Conversión de binario a hexadecimal 76
2.4.5. Conversión de hexadecimal a octal 76

2.4.6. Conversión de octal a hexadecimal 77
2.4.7. Aritmética hexadecimal 77

Ejercicios 78

3. REPRESENTACION DE LA INFORMACION

Cuadro resumen 83
3. Representación de la información 84
3.1. Representación y codificación de los datos 85
3.1.1. Código BCD 86
3.1.2. Código EBCDIC 87
3.1.3. Código ASCII 90
3.1.4. Código AIKEN 91
3.1.5. Código de exceso de tres 92
3.1.6. Código dos de cinco 94
3.1.7. Código Gray 95
3.1.8. Código de ficha perforada 96
3.1.9. Código de cinta perforada 98
3.1.10. Código de cinta magnética 99
3.2. Representación de las instrucciones 100
3.2.1. Tipos de instrucciones 100
3.2.2. Partes de una instrucción 103
– Parte operativa 105
– Cifras decimales 106
– Símbolos alfabéticos 106
– Parte direccional 106
– Instrucciones de una dirección 106
– Instrucciones de dos direcciones 107
– Instrucciones de tres direcciones 108

Ejercicios 109

4. LA LOGICA DEL ORDENADOR

Cuadro resumen 113
4.1. Lógica de los programas 114
4.2. Algebra de Boole 115
4.2.1. Tablas de la verdad 117
4.2.2. Operadores lógicos 119

4.2.3. Negación lógica 120
– Y Lógico (AND) 121
– O Inclusivo (OR) 123
– Igualdad lógica (EQUAL) 124
– Coincidencia lógica (COIN) 124
– O Exclusivo (EXOR) 124
– NO-Y Lógico (NAND) 125
– NO-O Lógico (NOR) 125
4.2.3. Representación de los operadores lógicos 126
4.2.4. Función booleana 126
4.2.5. Teoremas del álgebra de Boole 129
– Demostración de los teoremas 130
– Aplicación de los teoremas 136
4.2.6. Utilización del álgebra booleana en los ordenadores 137
– Circuitos eléctricos 138
– El álgebra booleana en los circuitos 140

Ejercicios 146

Soluciones de los ejercicios 149

En este volumen se describen los fundamentos principales para la introducción de pleno en el campo de la Programación. Los conceptos expuestos aparecen de una forma progresiva que desemboca en la preparación del lector a un más profundo estudio de los campos específicos de la operativa informática.

El proceso de datos, instrumentos, partes, componentes, y relaciones entre todos ellos, vienen descritos de manera que su asimilación y entendimiento sea perfectamente asequible a toda persona que acompañe a la exposición con un mínimo de estudio, constancia e interés.

Al final de los capítulos se expone una serie de ejercicios que sirven para apreciar si la materia estudiada está perfectamente comprendida y el mismo lector puede juzgarlo mediante el solucionario que subsiguientemente se adjunta.

Una característica importante del libro es la subdivisión con que se presentan los temas más importantes que conduce a una exposición explícita y clara, siempre de gran ayuda para el lector.

El dibujo de la cubierta es cortesía de GENERAL DE COMPUTADORES, S.A.