TEMA 2. REPRESENTACIÓN DE LA INFORMACIÓN

2.1. Sistemas de representación de la información
  2.1.1. Necesidad de representar la información
  2.1.2. Proceso de representación de los datos
  2.1.3. Tipos de representación. Proceso a seguir por el diseñador a la hora de elegir una representación
2.2. Representaciones no numéricas
  2.2.1. Codificación de caracteres alfanuméricos
  2.2.2. Codificación de las instrucciones
  2.2.3. Compactación de la información: código dependiente de la frecuencia (código Huffman) y codificación diferencial
2.3. Representaciones numéricas
  2.3.1. Introducción
  2.3.2. Sistemas posicionales
    2.3.2.1. Coma fija sin signo. Binario puro sin signo
    2.3.2.2. Coma fija con signo. Binario puro con signo
    2.3.2.3. Coma fija con complemento a la base. Complemento a 2
    2.3.2.4. Coma fija con complemento restringido a la base. Complemento a 1
    2.3.2.5. Representación en exceso Z
    2.3.2.6. Sistemas decimales codificados en binario (BCD)
    2.3.2.7. Representación de números fraccionarios en coma fija
    2.3.2.8. Representación de números fraccionarios en coma flotante
      2.3.2.8.1. Mantisa entera
      2.3.2.8.2. Mantisa fracción. Normalización
      2.3.2.8.3. Estándar IEEE P754
  2.3.3. Sistemas de residuos
2.4. Representaciones redundantes
  2.4.1. Introducción del concepto de redundancia
  2.4.2. Códigos detectores de error
  2.4.3. Códigos correctores de error
  2.4.4. Códigos polinomiales
2.5. Representaciones de estructuras de datos
  2.5.1. Representación en la máquina de Von Neumann
  2.5.2. Representación en los computadores Burroug B66700. Datos etiquetados
2.6. Representaciones gráficas

Uno de los primeros pasos que hay que dar, necesarios para el diseño de un computador, es el de la selección de el/los sistema/s de representación de la información. Este tema aborda el estudio de las distintas formas de representar la información en el computador.

El tema comienza con una introducción general, haciéndose patente la necesidad de la representación de la información en el computador, así como haciendo referencia al proceso de representación de datos. Se estudian los sistemas de representación no numéricos y numéricos empleados, estos últimos tanto en la forma de coma fija como en la forma de coma flotante, y se profundiza en la representación numérica del estándar IEEE P754. Se aborda el estudio de las representaciones redundantes, capaces de detectar e incluso corregir errores, realizando especialmente el estudio de los códigos polinomiales. Por último se estudian las representaciones gráficas desde un punto de vista general.

2.1. SISTEMAS DE REPRESENTACIÓN DE LA INFORMACIÓN

2.1.1. Necesidad de representar la información

El hombre maneja informaciones de cifras, palabras e imágenes; el computador las maneja de tipo binario exclusivamente. Por lo tanto, es necesario establecer un mecanismo de traducción o codificación que permita la comunicación entre el hombre y el computador.

La selección de una u otra codificación viene condicionada por los costes de: *la traducción*, *el almacenamiento de la información* y *el tratamiento de la información*. Si la información es texto, el coste más importante es el de *traducción*, y la traducción consiste en codificar cada letra por separado; si se trata de cantidades numéricas, podemos encontrarnos con distintas situaciones: tratarla como texto o representarla con formatos de coma fija y de coma flotante.

En todo computador sencillo, existen dos flujos de información: *datos* e *instrucciones* (Figura 2.1). Los *datos de entrada* se procesan según indican las *instrucciones*, para producir los *resultados (o datos de salida)*.

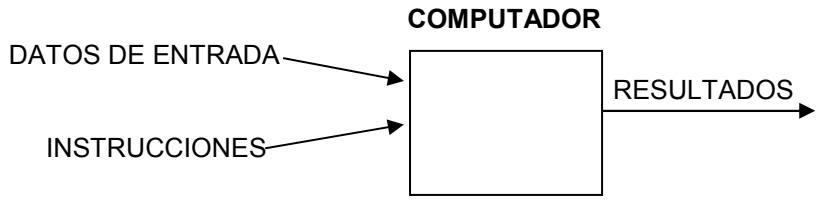


Figura. 2.1. *Procesamiento de la información.*

La elección de los métodos de representación de datos es uno de los primeros pasos en el diseño de un computador, su definición es uno de los aspectos más importantes en el diseño de todo nuevo computador. Las características de un computador se fijan tanto por los tipos de datos que soporta como por las operaciones que se pueden hacer sobre ellos.

2.1.2. Proceso de representación de los datos

El proceso completo de representación de datos parte de los conceptos que queremos manipular y termina en los *elementos materiales del computador*. En la representación en el computador se pueden diferenciar cuatro pasos:

1. Formalizar el ente abstracto a representar. No es una tarea fácil.

2. Representación con elementos lingüísticos. Es decir, representación mediante palabras.
3. Codificación de las palabras siguiendo las normas de un sistema de numeración determinado. Se define una cadena.
4. Se representan las cadenas con información manejable por el computador, cadenas de unos y ceros.

Los datos del computador se guardan en sus registros internos y en posiciones de memoria, constituidas por elementos materiales que determinan el *ESPACIO MATERIAL*. Su número y la estructura de los mismos determinan la **"**capacidad representación del computador**"**.

El *ESPACIO MATERIAL* tiene tres condicionantes fundamentales:

- El espacio material es de tipo binario. Cada elemento material es capaz de almacenar o de tratar un bit o dígito binario.
- El espacio material es finito. Las representaciones tienen que ser acotadas.
- Los elementos materiales presentan tamaños determinados definidos tanto por su construcción como por el ancho de los caminos de acceso en paralelo. La elección de los tamaños determinados (*TAMAÑOS PRIVILEGIADOS*) constituye una de las acciones prioritarias en la definición de la arquitectura de un computador.

Existen en un computador varios *TAMAÑOS PRIVILEGIADOS*:

- RESOLUCIÓN DE ACCESO. La menor cantidad de información accesible y manipulable.
- BYTE O CARÁCTER. Espacio utilizado para representar un carácter alfanumérico.
- PALABRA. Tamaño de información, medido en bits, que trata el computador en paralelo.
- MEDIA PALABRA. Su manipulación en algunos computadores se hace directamente.
- DOBLE PALABRA. Para mejorar la precisión de cálculo. En algunos computadores existen unidades operativas capaces de operar directamente con dobles palabras.
- RESOLUCIÓN DE ACCESO A MEMORIA. Menor tamaño de información que es capaz de leer directamente de la memoria. Esta característica resulta muy importante cuando hay que tratar información de tamaño reducido, como por ejemplo en control de procesos.

Las relaciones de todos los *tamaños privilegiados* correspondientes a un mismo computador son números enteros, siendo la *palabra* el tamaño principal: el *byte* y otros tamaños menores deben dividirla de forma exacta y los tamaños mayores deben ser sus múltiplos.

2.1.3. Tipos de representación. Proceso a seguir por el diseñador a la hora de elegir una representación

Los distintos tipos de representación quedan clasificados teniendo en cuenta los siguientes factores:

- Elemento a representar (número, carácter alfanumérico, gráfico, …)
- Forma de representación (binario puro, coma flotante,...)
- Características de la representación (tolerante a fallos, con paridad,...)

- Representaciones alfanuméricas
- Representaciones numéricas:
    - sistemas posicionales
    - sistemas enteros racionales
    - sistemas de residuos
- Representaciones redundantes:
    - bits de paridad
    - códigos de Hamming
    - códigos polinomiales
- Representación de estructuras de datos (con etiquetas)
- Representaciones gráficas

Las representaciones numéricas que soporta una máquina, de las que existen muchas alternativas, constituyen una de las características más influyentes sobre la arquitectura de la misma.

El *proceso a seguir por el diseñador para la elección de una representación* consta de los siguientes pasos:

1º La *Codificación de caracteres Alfanuméricos*. Ésta determinará el tamaño del *byte* de la máquina.

2º El o los *sistemas numéricos* empleados. La variedad de éstos es amplia.

3º *Conjunto de números* que queremos representar (Enteros positivos, enteros positivos y negativos, reales, reales menores que la unidad, complejos, fracciones, …).

4º Definir la *resolución*. Determinará el *formato de los datos*, es decir, el tamaño y el significado.

2.2. REPRESENTACIONES NO NUMÉRICAS

En un sistema computador hay informaciones que requieren un tratamiento diferente del aritmético. Hay dos tipos de *representaciones no numéricas*:

- Las que se tratan como *datos*, las *representaciones alfanuméricas*.
- Las *instrucciones*.

En el computador Von Neumann no existe diferencia explícita en la representación de las informaciones, tanto los datos como las instrucciones se representan con una combinación de unos y de ceros. La diferencia entre dato o instrucción la establece el contexto en el que se encuentre.

2.2.1. Codificación de caracteres alfanuméricos

Las informaciones de tipo texto se representan con una codificación independiente (byte) para cada carácter. Las características que definen un *sistema de representación alfanumérico* concreto son:

- Tamaño del byte. Éste define el número de caracteres distintos que se pueden representar.
- Codificación de cada carácter.
- Todos los sistemas de codificación alfanuméricos procuran cumplir tres reglas muy útiles:
    - Que sea sencillo diferenciar los caracteres numéricos del resto, así como que se determine fácilmente el valor del mismo.
    - Que se diferencien en un sólo bit la codificación de las mayúsculas y las minúsculas. De esta forma se facilita la comunicación con dispositivos periféricos que no diferencian mayúsculas y minúsculas.
    - Que se distingan claramente las codificaciones de los caracteres numéricos, alfabéticos y de control De esta forma se consiguen simplificar los programas de interpretación de estos elementos.

Cuando se habla de representaciones alfanuméricas, otro aspecto importante a tener en cuenta es la *forma de separar cadenas*. Se pueden emplear las siguientes técnicas:

- Cadenas de longitud fija. Cada conjunto de n bytes contiene una determinada información.
- Cadenas de longitud variable. Con dos variantes:
  - Separación entre datos con un símbolo específico.
  - Comienzo del dato con una cabecera que indica su longitud.

2.2.2. Codificación de las instrucciones

Las instrucciones definen las siguientes informaciones:

- Código de operación.
- Direcciones de registros o de posiciones de memoria.
- Modos de direccionamiento.
- Tipos de operandos.

Se profundizará más adelante en la codificación de las instrucciones, en el tema relativo a la Unidad de Control del computador.

2.2.3. Compactación de la información: código dependiente de la frecuencia (código Huffman) y codificación diferencial

Cuando hay que almacenar o transmitir textos muy grandes conviene buscar una forma de comprimirlos, empleando un número de bits inferior al que se necesitaría con una codificación que emplee el mismo número de bits para representar cada carácter independientemente. La compactación de la información trata de reducir el número de bits necesarios para representar una información. Al comprimir, se reduce el tiempo de transmisión así como el espacio necesario para almacenar la información.

Se puede hablar de dos técnicas básicas de compresión de la información:

- *Códigos dependientes de la frecuencia* o codificación según frecuencia de uso (código Huffman). Con esta codificación, los caracteres más usados tienen asociados códigos más cortos que los menos usados. Antes de establecer la codificación hay que hacer un estudio estadístico sobre la frecuencia de utilización de todos y cada uno de los caracteres a codificar.
- *Codificación diferencial*. Se emplea cuando las unidades de información sucesivas difieren poco entre sí, siendo mejor codificar las diferencias entre estas unidades

de información antes que ellas mismas. Aunque esta codificación presenta un gran ahorro en el número de bits a emplear, tiene los siguientes inconvenientes:

- Hay que realizar restas y sumas adicionales para la codificación y la decodificación.
- Hay propagación de errores.
- Es necesario un mecanismo para que el sistema pueda empezar (no hay diferencia al principio). Este mecanismo deberá también poder tratar diferencias esporádicas mayores que las que se consideren.

2.3. REPRESENTACIONES NUMÉRICAS

Gran parte de la información que procesa un computador corresponde a valores numéricos. Este apartado trata de las diferentes formas que se emplean a la hora de representar los valores numéricos en el sistema computador.

2.3.1. Introducción

Los números reales se pueden clasificar de la siguiente forma

- Naturales. (1, 2, 3…).
- Enteros (naturales positivos y negativos, y el 0).
- Racionales. Aquellos números que pueden representarse como cociente de dos números enteros; y, a su vez, pueden implicar un número finito de decimales, o un número infinito de decimales correspondiéndose con una forma periódica.
- Irracionales. Aquellos números reales que no pueden ser expresados como cociente de dos números enteros; e implican un número infinito de decimales.

Cualquier conjunto de números reales es infinito; por lo tanto, no podemos representarlos todos, dadas las limitaciones del espacio material (éste es finito). Y, de la misma forma, como hay números reales con infinitos dígitos, no podemos representar a estos valores exactamente.

En el computador hay que fijar "n" (número de bits para representar un número, un valor numérico). Este número de bits define tanto el *rango* como la *resolución* del sistema de representación considerado.

- Rango. Conjunto de valores representables. Intervalo comprendido entre el menor y el mayor número representable.
- Resolución. Diferencia entre dos representaciones consecutivas.

Existen diversos métodos de representación de datos numéricos que optimizan la utilización de los n bits seleccionados para determinadas aplicaciones.

Los sistemas de representación más empleados son los siguientes:

- SISTEMAS POSICIONALES:
  - Coma fija sin signo (positivos).
  - Coma fija con signo (enteros).
  - Coma fija con complemento a la base (enteros).
  - Coma fija con complemento restringido a la base (enteros).
  - Coma fija en exceso Z (enteros).
  - Coma fija BCD (enteros).
  - Coma flotante normalizada (racionales).
  - Coma flotante no normalizada (racionales).
- SISTEMAS ENTEROS RACIONALES.
- SISTEMAS DE RESIDUOS.

2.3.2. Sistemas posicionales

En un sistema de representación posicional, los números se representan por cadenas de dígitos, viniendo el valor de cada dígito supeditado a la posición que ocupa en la cadena.

$$X := (...x_3x_2x_1x_0, x_{-1}x_{-2}x_{-3}...)$$

Vector de pesos: $P := (...p_3p_2p_1p_0, p_{-1}p_{-2}p_{-3}...)$

$$V(x) = ... + p_3x_3 + p_2x_2 + p_1x_1 + p_0x_0 + p_{-1}x_{-1} + p_{-2}x_{-2} + p_{-3}x_{-3} + ... = \sum_{i=-\infty}^{+\infty} p_i x_i$$

Normalmente $P := (...b^3b^2b^1b^0, b^{-1}b^{-2}b^{-3}...)$; en este caso:

$$V(x) = \sum_{i=-\infty}^{+\infty} b^i x_i$$

$X := (...x_3x_2x_1x_0, x_{-1}x_{-2}x_{-3}...)_b$ representa a un número en base b; donde los dígitos x_i posibles son:

Dígito x_i | $0 \le x_i < b$

PROPIEDADES IMPORTANTES DE LOS SISTEMAS POSICIONALES:

- Si se tienen dos bases b_1 y b_2 tales que $b_1 = b_2^k$, entonces, los dígitos de la base b_1 se obtienen agrupando los dígitos de la base b_2 en grupos de longitud k.
- Un nº racional puede tener una representación exacta en una base b_1, exigiendo en cambio una representación periódica para una base b_2. Esto plantea un claro problema puesto que hay que encajar este número en una cadena finita de n bits, cometiéndose por lo tanto un error no esperado.

2.3.2.1. Coma fija sin signo. Binario puro sin signo

Es el formato más simple para representar números enteros positivos.

$$X := (x_{n-1}x_{n-2}...x_1x_0)$$

Se trata de un sistema posicional con base 2 y sin parte fraccionaria.

Valor de cualquier representación:

$$V(x) = \sum_{i=0}^{n-1} x_i 2^i$$

Rango: $0 \le x \le 2^n - 1$

Resolución: 1

Cambio de signo: No

Extensión de signo: Se rellenan las posiciones sobrantes por la izquierda con ceros

Dificultades:

- El resultado de una suma puede necesitar n+1 bits. Por lo tanto hay que detectar el desbordamiento. También con el producto puede haber desbordamiento.
- Por no poderse representar números negativos, al hacerse la operación de resta hay que comprobar previamente si el minuendo es mayor que el sustraendo.

2.3.2.2. Coma fija con signo. Binario puro con signo

En este caso se reserva un bit para indicar el signo del valor. Es decir, de los n bits disponibles, uno indica el signo y los n-1 restantes indican la magnitud (representación ***signo-magnitud***). Normalmente el bit de la izquierda suele ser el bit de signo.

$$+ \begin{cases} 00.....0 \rightarrow 0 \\ \vdots \\ 01.....1 \rightarrow 2^{n-1}-1 \end{cases}$$

$$- \begin{cases} 10.....0 \rightarrow -0 \\ \vdots \\ 11.....1 \rightarrow 1-2^{n-1} \end{cases}$$

Valor de cualquier representación:

$$V(x) = (1-2x_{n-1})\sum_{i=0}^{n-2} x_i 2^i$$

Rango: $(1-2^{n-1})\cdots(2^{n-1}-1)$

Resolución: 1

Cambio de signo: Se complementa el bit de signo.

Extensión de signo: El bit de signo original pasa a ser el bit de signo de la nueva representación, los bits de la magnitud son los mismos que los bits extremos derechos de la nueva representación, y las posiciones sobrantes se rellenan con ceros.

Dificultades:

- Dos representaciones para el valor “0”.
- La operación a realizar, suma o resta, depende de los operandos. Con la multiplicación y la división no existe este problema puesto que se opera con los signos y las magnitudes independientemente.
- Hay posibilidad de desbordamiento al operar con sumas, restas y multiplicaciones.

2.3.2.3. Coma fija con complemento a la base. Complemento a 2

El complemento a 2 es el caso particular del complemento a la base cuando ésta vale 2. Con n bits se desean representar tanto números enteros positivos como negativos. Se reservan la mitad de los códigos para los números positivos (los que tienen como primer bit de la izquierda un 0) y la otra mitad para los negativos (los que tienen como primer bit de la izquierda un 1). Los números positivos se representan en binario puro, y los números negativos se representan con su complemento a 2 (restando de $\mathbf{2^n}$ la magnitud del número a representar).

$$+ \begin{cases} 00.....0 \rightarrow 0 \\ \vdots \\ 01.....1 \rightarrow 2^{n-1}-1 \end{cases}$$

$$- \begin{cases} 10.....0 \rightarrow -2^{n-1} \\ \vdots \\ 11.....1 \rightarrow 1 \end{cases}$$

Valor de cualquier representación:

$$V(x) = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2_i$$

Rango: $-2^{n-1} \cdots (2^{n-1}-1)$

Resolución: 1

Cambio de signo: Se aplica la regla del complemento a 2.

Extensión de signo: Se rellenan las posiciones sobrantes con el mismo valor que el bit extremo izquierdo de la representación original.

Dificultades:

- Rango de representación no simétrico.
- La operación de multiplicación se complica cuando el/los operando/s están complementados
- Hay posibilidad de desbordamiento al operar con sumas, restas y multiplicaciones.

Cálculo práctico del complemento: Consiste en cambiar “unos” por “ceros” y “ceros” por “unos” y al resultado sumarle “uno”.

2.3.2.4. Coma fija con complemento restringido a la base. Complemento a 1

El complemento a 1 es el caso particular del complemento restringido a la base cuando la base vale 2. Con n bits se desean representar tanto números enteros positivos como negativos. Se reservan la mitad de los códigos para los números positivos (los que tienen como primer bit de la izquierda un 0) y la otra mitad para los negativos (los que tienen como primer bit de la izquierda un 1). Los números positivos se representan en binario puro, y los números negativos se representan con su complemento a 1 (restando de $\mathbf{2^n - 1}$ la magnitud del número a representar).

$$+ \begin{cases} 00.....0 \rightarrow 0 \\ \vdots \\ 01.....1 \rightarrow 2^{n-1}-1 \end{cases}$$

$$- \begin{cases} 10.....0 \rightarrow 1-2^{n-1} \\ \vdots \\ 11.....1 \rightarrow -0 \end{cases}$$

Valor de cualquier representación:

$$V(x) = -(2^{n-1}-1)x_{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

Rango: $(1-2^{n-1})\cdots(2^{n-1}-1)$

Resolución: 1

Cambio de signo: Se aplica la regla del complemento a 1.

Extensión de signo: Se rellenan las posiciones sobrantes con el mismo valor que el bit extremo izquierdo de la representación original.

Dificultades:

- Dos representaciones para el valor “0”.
- Comparándolo con el complemento a 2, se complican algo más las operaciones de suma y resta.
- La operación de multiplicación se complica cuando el/los operando/s están complementados
- Hay posibilidad de desbordamiento al operar con sumas, restas y multiplicaciones.

Cálculo práctico del complemento: Consiste en cambiar “unos” por “ceros” y “ceros” por “unos”.

2.3.2.5. Representación en exceso Z

Este sistema se emplea para representar los exponentes en formato de coma flotante. Se representan los números incrementados en Z, en binario natural. Es decir, A se representa por A+Z en binario natural.

Generalmente $Z=2^{n-1}$. Se va a estudiar el sistema de representación considerando ese Z.

$$X := (x_{n-1}x_{n-2}\ldots x_1x_0)$$

$$\text{-}\begin{cases} 00.....0 \rightarrow -2^{n-1} \\ \vdots \\ 01.....1 \rightarrow -1 \end{cases}$$

$$+\begin{cases} 10.....0 \rightarrow 0 \\ \vdots \\ 11.....1 \rightarrow 2^{n-1}-1 \end{cases}$$

Valor de cualquier representación:

$$V(x) = \sum_{i=0}^{n-2} x_i 2^i - 2^{n-1} = -(1-x_{n-1})2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i = -2^{n-1}\overline{x}_{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

Se observa que es una representación muy parecida al complemento a 2; basta invertir el primer bit para que sean iguales.

Rango: $-2^{n-1}\cdots(2^{n-1}-1)$

Resolución: 1

Cambio de signo: Se aplica la regla del complemento a 2 (igual que en el sistema de complemento a 2).

(A) positivo → $2^{n-1} + A$

(-A) negativo → $2^{n-1} - A$

Si le aplicamos el complemento a dos:

$$+ \rightarrow 2^n - (2^{n-1} + A) = 2^{n-1}(2-1) - A$$

$$- \rightarrow 2^n - (2^{n-1} - A) = 2^{n-1}(2-1) + A$$

Extensión de signo

- *Considerando el mismo exceso Z en ambos sistema* Se rellenan las posiciones sobrantes con ceros (igual que en el binario natural).
- *Considerando* $Z = 2^{n-1}$ *para el sistema de partida y* $Z = 2^{m-1}$ *para el sistema final (m > n)* Se invierte el bit n-1 y se rellenan con este valor todos los bits adicionales menos el bit m-1 que ha de tomar el antiguo valor del bit n-1.

Dificultades:

- Rango de representación no simétrico.
- Las operaciones de suma y resta exigen corrección.
- La operación de multiplicación se complica.

- Hay posibilidad de desbordamiento al operar con sumas, restas y multiplicaciones.

Ventajas:

- Presenta la misma ordenación que el binario natural.
- Puede representar enteros positivos y negativos.
- Para el exceso Z = 2^{n-1}, presenta las mismas ventajas para operaciones aritméticas que el sistema de representación en complemento a 2.

A continuación se muestra una tabla comparativa entre los distintos sistemas de representación posicionales vistos hasta el momento, para el caso de n = 3 bits.

Decimal	Binario Puro	Signo-Magnitud	Complemento a 2	Complemento a 1	Exceso Z
+7	111	N.D.	N.D.	N.D.	N.D.
+6	110	N.D.	N.D.	N.D.	N.D.
+5	101	N.D.	N.D.	N.D.	N.D.
+4	100	N.D.	N.D.	N.D.	N.D.
+3	011	011	011	011	111
+2	010	010	010	010	110
+1	001	001	001	001	101
+0	000	000	000	000	100
-0	N.D.	100	N.D.	111	N.D.
-1	N.D.	101	111	110	011
-2	N.D.	110	110	101	010
-3	N.D.	111	101	100	001
-4	N.D.	N.D.	100	N.D.	000
-5	N.D.	N.D.	N.D.	N.D.	N.D.
-6	N.D.	N.D.	N.D.	N.D.	N.D.
-7	N.D.	N.D.	N.D.	N.D.	N.D.

N.D.: No Definido

2.3.2.6. Sistemas decimales codificados en binario (BCD)

En estos sistemas de representación se convierten independientemente cada dígito decimal a binario. Para representar un dígito decimal (0, 1, 2, 3,…, 9) con información binaria se necesitan 4 bits; se desaprovechan el 6/16 de los códigos (37.5%).

Aunque se desaprovechen el 37.5% de los códigos, su utilización viene justificada por:

- La E/S de datos del computador debe realizarse usando la base decimal. Por lo tanto, si los cálculos a realizar por el computador son muy simples, no compensa pasar de decimal a binario, procesar y después pasar de binario a decimal.
- Los errores de redondeo en binario y en decimal son diferentes.

Los sistemas BCD más corrientes son el BCD natural y el BCD exceso 3. Este último simplifica el diseño de la unidad operativa.

En las representaciones alfanuméricas, la representación de los números se corresponde con una representación BCD que emplea 6, 7 u 8 bits por dígito (según el código alfanumérico empleado). Para estos casos cabe hablar de dos formas de representación:

- DECIMAL DESEMPAQUETADO

En este sistema, un número se almacena con un byte por cada una de sus cifras. Cada byte lleva en su cuarteto de la izquierda cuatro unos y en el de la derecha la cifra en BCD. El cuarteto de la izquierda de la última cifra representa el signo:

1 1 0 0 → + (C en hexadecimal)

1 1 0 1 → - (D en hexadecimal)

Ejemplo:

1992 →	1111	0001	1111	1001	1111	1001	1100	0010
		1		9		9	+	2
-1992 →	1111	0001	1111	1001	1111	1001	1101	0010
		1		9		9	-	2

- DECIMAL EMPAQUETADO

En este sistema, se elimina el cuarteto de la izquierda del sistema anterior, que no contenía información salvo en la última cifra. En este caso cada cuarteto lleva una cifra en BCD, salvo el primero por la derecha que lleva el signo:

1 1 0 0 → + (C en hexadecimal)

1 1 0 1 → - (D en hexadecimal)

Ejemplo:

1992 →	0000	0001	1001	1001	0010	1100
		1	9	9	2	+
-1992 →	0000	0001	1001	1001	0010	1101
		1	9	9	2	-

2.3.2.7. Representación de números fraccionarios en coma fija

Hasta el momento se han considerado sistemas de representación de números enteros. También se apunta la posibilidad de representar un número con parte fraccionaria como una *parte entera*, la *coma* y la *parte fraccionaria*; esto no se suele emplear.

Es cierto que, empleando el escalado adecuado, se puede conseguir que cualquier cantidad quede dentro del rango de representación de cualquiera de los sistemas de coma fija vistos (por ejemplo, dinero en céntimos de euro en vez de en euros). Pero hacerlo de esta forma conlleva dificultades de cálculo nada despreciables:

- Productos y divisiones entre cantidades escaladas producen cambios de escala.
- Si no se conocen a priori las magnitudes de los resultados y de los números intermedios, es muy difícil realizar el escalado correcto, que prevenga desbordamientos y no desaproveche el rango de representación.
- Es una labor muy tediosa y propensa a equivocaciones.

Por todas estas razones, los sistemas de coma fija no suelen emplearse para representar números fraccionarios.

2.3.2.8. Representación de números fraccionarios en coma flotante

El computador se va a encargar de la tarea del escalado, de forma dinámica y óptima. La representación en coma flotante divide los n bits en dos partes; concatena con el número un factor de escala:

- **Mantisa**. Contiene los dígitos significativos del número.
- **Exponente**. Indica el factor de escala como una potencia de la **base** r.

La *mantisa* y el *exponente* se representan en algunos de los sistemas de coma fija estudiados anteriormente.

n bits

EXP.(q bits)	MANTISA (p bits)

$V(x) = M \cdot r^E$

M → Valor de la mantisa

r → Base del exponente

E → Valor del exponente

La base de representación de M debe coincidir con r, pues es la única forma de ajustar fácilmente el exponente.

Cabe hablar de dos tipos de mantisa:

- MANTISA ENTERA → Coma a la derecha m_0.
- MANTISA FRACCIÓN → Coma a la izquierda (entre el bit m_{p-2} y m_{p-1})→ el valor entero hay que dividirlo entre 2^{p-1}).

2.3.2.8.1. Mantisa entera

La mantisa entera se emplea cada vez menos. Todos los nuevos sistemas tienden a emplear el estándar IEEE P754, que emplea mantisa fracción.

Para el cálculo del rango, hay que calcular los mayores y menores valores representables en la zona a la derecha del cero (números positivos) y en la zona a la izquierda del cero (números negativos). En el campo de los números positivos, el menor número representable será la menor mantisa positiva afectada del menor exponente, y el mayor número representable será la mayor mantisa (positiva) afectada del mayor exponente; en la zona de los números negativos, el menor número representable (número más alejado del cero por la izquierda) será la menor mantisa negativa (mayor valor absoluto) afectada del mayor exponente, y el mayor número representable (número más próximo a cero por la izquierda) es la mayor mantisa negativa (menor valor absoluto) afectada del menor exponente.

2.3.2.8.2. Mantisa fracción. Normalización

Cuando tenemos la mantisa fracción, la coma implícita se encuentra a la izquierda del bit más significativo de la mantisa. Según esté codificada la mantisa, su valor real vendrá dado por:

- **Mantisa en signo magnitud** → $M = [(1 - 2x_{p-1}) \sum_{i=0}^{p-2} x_i 2^i] / 2^{p-1}$

- **Mantisa en complemento a 2** → $M = [-x_{p-1}2^{p-1} + \sum_{i=0}^{p-2} x_i 2^i] / 2^{p-1}$

- **Mantisa en complemento a 1** → $M = [-x_{p-1}(2^{p-1} - 1) + \sum_{i=0}^{p-2} x_i 2^i] / 2^{p-1}$

Los valores de las mantisas consideradas son válidos nada más para la posición de la coma propuesta (mantisa con p-1 bits a la derecha de la coma), que es caso más corriente; para cualquier otra posición, habrá que dividir los valores enteros por 2^x, siendo x el número de bits que quedarían a la derecha de la coma.

Generalmente la mantisa está **normalizada**. Esta **normalización** consiste en eliminar todos los dígitos nulos inmediatos a la derecha de la coma, ajustando adecuadamente el exponente. Con esta normalización, el rango de mantisas posibles está comprendido entre $0{,}1\ 0\ 0\ \ldots\ 0)_r$ y $0{,}r\text{-}1\ r\text{-}1\ \ldots\ r\text{-}1)_r$, siendo **r** la base de representación considerada para la mantisa y **r-1** el mayor dígito posible para esa base.

El sistema de representación más empleado para la mantisa es el de signo-magnitud; para este caso, la normalización consiste en hacer que el dígito más significativo sea distinto de cero. Hay que tener en cuenta que si la base es $r \neq 2$, el primer bit significativo de una mantisa normalizada puede ser cero pero no puede ser cero el primer dígito significativo de la mantisa.

Cuando en la representación signo-magnitud de la mantisa es r = 2, el primer dígito es el primer bit y, si la mantisa está normalizada, este primer bit tiene que ser uno. Para este caso, si todas las representaciones posibles de la mantisa tienen como primer bit significativo un uno no es necesario incluirlo en el formato de representación del número a la hora de almacenarlo. Cuando no se incluye este bit en el formato se habla de **bit implícito**. Este bit implícito nos va a permitir operar con números que tienen un bit más en la mantisa que los que aparecen representados en el formato, permitiéndonos mejorar la resolución del sistema de representación (con bit implícito → p+q = n+1).

La representación en coma flotante con mantisa normalizada presenta como problema en el rango de representación la existencia de un "hueco" alrededor del cero, ya que no están permitidas las mantisas comprendidas entre el $0{,}0\ 0\ 0\ \ldots\ 01)_r$ y el $0,\ 0\ r\text{-}1\ r\text{-}1\ \ldots\ r\text{-}1)_r$. Esta dificultad está justificada por la facilidad de comparación de dos cantidades representadas en formato de coma flotante con mantisa fracción; para este caso, la comparación de dos cantidades se puede realizar de manera ordenada de la siguiente forma:

1º **Comparación de los signos**. Se comparan los signos del formato y, si son distintos, el positivo es el mayor y se termina la comparación. Si las dos representaciones presentan el mismo signo, hay que seguir comparando los exponentes de los números.

2º **Comparación de los exponentes**. Como el exponente en un formato de coma flotante normalmente se representa en exceso, la ordenación es la misma que en binario natural, y la comparación consiste en comparar los bits de la misma posición de ambos exponentes, en orden de izquierda a derecha de las representaciones; cuando se encuentre la primera diferencia, un bit cero en un exponente y un bit uno en el otro, el exponente de bit uno es el mayor. El número en formato de coma flotante con ese exponente mayor, es el mayor si ambos eran positivos, y es el menor si ambos eran negativos, y se termina la comparación. Si los dos exponentes son iguales, hay que seguir la comparación.

3º **Comparación de las mantisas**. La representación de la mantisa más empleada es la representación signo-magnitud. Con esta representación es muy fácil realizar la comparación de las magnitudes de las mantisas ya que la ordenación de éstas coinciden con la del binario natural, se van comparando los bits de la misma posición de ambas magnitudes de la mantisa en orden de izquierda a derecha, en el primer bit que se diferencien, la magnitud con un uno es mayor que la que tiene el cero. El número en formato de coma flotante con esa magnitud mayor, es el mayor si ambos eran positivos, y es el menor si ambos eran negativos. Si las dos magnitudes son iguales, se concluye que ambos números en formato de coma flotante comparados son iguales.

En resumen, si se define el formato con el orden: **signo-exponente-magnitud**, para la mantisa como signo magnitud y para el exponente en exceso, la comparación se hace de izquierda a derecha hasta encontrar los primeros bits de la misma posición distintos, con los que se terminaría la comparación.

2.3.2.8.3. Estándar IEEE P754

Se encuentra presente en cada computador aparecido a partir de mediados de los 80. Este estándar incluye tanto las representaciones de los operandos como la forma de realizar las operaciones; especifica el formato de los números, las operaciones básicas, las conversiones y las condiciones excepcionales. En este apartado nos centraremos en las representaciones de los operandos.

El estándar considera dos formatos básicos, el de **simple** y el de **doble** precisión (Figura 2.2).

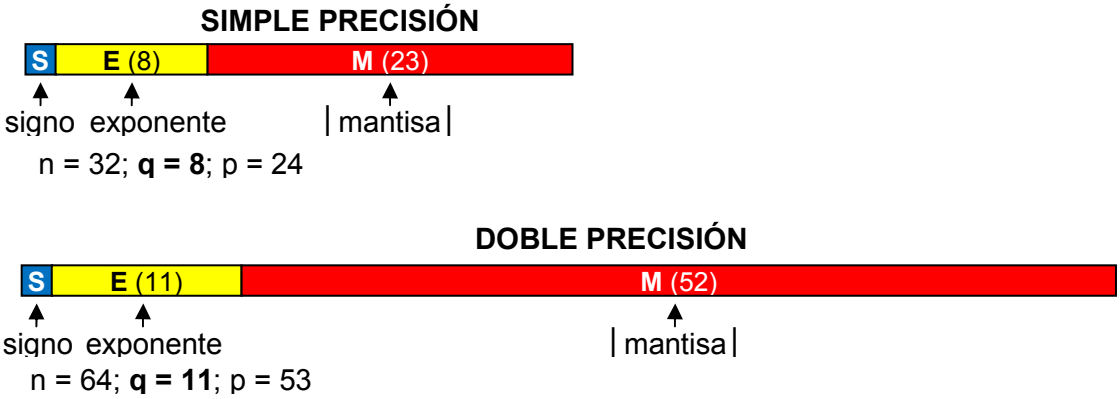


Figura. 2.2. *Formatos de representación en simple y doble precisión del IEEE P754.*

A continuación, cada vez que aparezcan **E** o **M** se referirán a los valores del exponente o la mantisa, respectivamente, en binario natural.

Las características de los formatos definidos por este estándar son:

- **Exponente** con representación en **exceso 2^{q-1}-1**:
  - Simple precisión: exceso $2^{q-1}-1=2^{8-1}-1=$ **127**
  - Doble precisión: exceso $2^{q-1}-1=2^{11-1}-1=$ **1023**
- **Base** del exponente **r = 2**.
- **Mantisa** con representación **signo-magnitud**.
- **Mantisa fraccionaria normalizada con bit implícito** (excepto en la forma desnormalizada). El bit implícito constituye la parte entera de la mantisa.
- Fórmula con **valor de la representación normalizada:**

  - Simple precisión: $\mathbf{V(X)=(-1)^S \cdot 1,M \cdot 2^{E-127}}$

  - Doble precisión: $\mathbf{V(X)=(-1)^S \cdot 1,M \cdot 2^{E-1023}}$

- **Casos especiales**:
  - **E = 0** (en el campo exponente la combinación de bits 0 …0)
    - **M = 0** (en el campo mantisa la combinación de bits 0 … 0) → Representa al valor "**0**"

- **M ≠ 0** → Representaciones de números pequeños en forma **desnormalizada**; su valor viene dado por:

  - Simple precisión: $\mathbf{V(X)=(-1)^S \cdot 0,M \cdot 2^{-126}}$

  - Doble precisión: $\mathbf{V(X)=(-1)^S \cdot 0,M \cdot 2^{-1022}}$

- **E = 255** en simple precisión ó **2047** en doble precisión (en el campo exponente la combinación de bits 1 … 1)

  - **M = 0** (en el campo mantisa la combinación de bits 0 … 0) → Representa el valor de "$+\infty$" para **S = 0** y el de "$-\infty$" para **S=1**

  - **M ≠ 0** → Para indicar que el resultado no tiene sentido (0/0; **∞/∞; …)**

Una vez definidas las características del estándar de representación se va a estudiar el mismo (rango, resolución, error absoluto y error relativo).

En la Figura 2.3 aparece un esquema de representación en la recta real que nos puede ayudar al estudio del sistema de representación definido por el estándar.

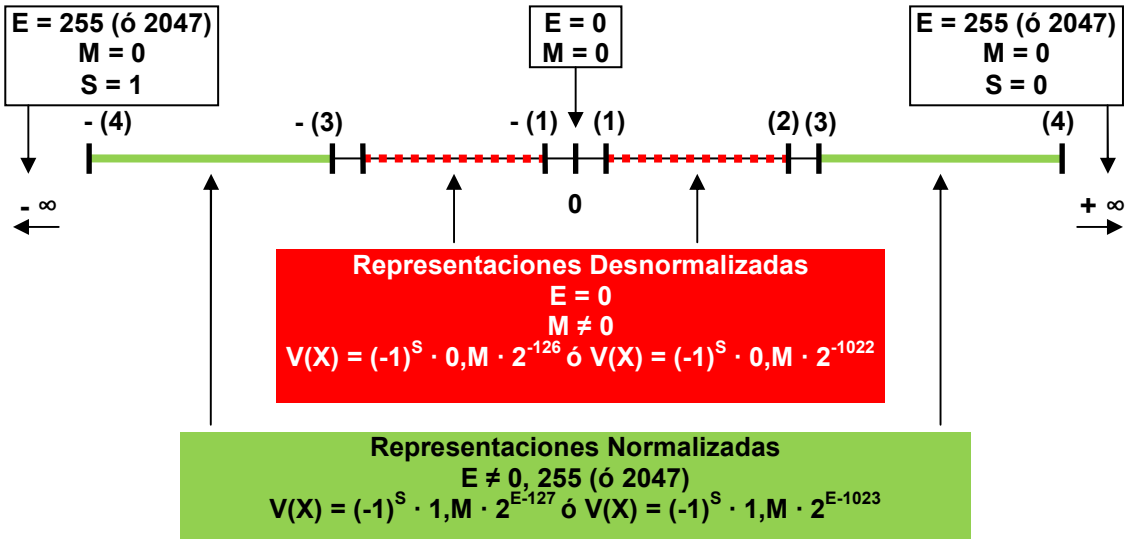


Figura. 2.3. *Esquema de representación del estándar IEEE P754 en la recta real.*

REPRESENTACIONES DESNORMALIZADAS (DN):

- **Rango**:

  - Simple precisión:

    - Punto **(1)**: **Menor M** (0,0 … 01) → $2^{-23} \cdot 2^{-126} = 2^{-149}$

- Punto **(2)**: **Mayor M** (0,1 … 11) → $(1 - 2^{-23}) \cdot 2^{-126}$

▪ Doble precisión:

- Punto **(1)**: **Menor M** (0,0 … … … … 01) → $2^{-52} \cdot 2^{-1022} = 2^{-1074}$
- Punto **(2)**: **Mayor M** (0,1 … … … … 11) → $(1 - 2^{-52}) \cdot 2^{-1022}$

- **Resolución**: (Diferencia entre dos representaciones consecutivas)

▪ Simple precisión:

$$Res_{DN} = |M_1 - M_2| \cdot 2^{-126} = 2^{-23} \cdot 2^{-126} = 2^{-149} \Rightarrow \mathbf{Res_{DN} = 2^{-149}}$$

M_1 y M_2 son mantisas consecutivas

▪ Doble precisión:

$$Res_{DN} = |M_1 - M_2| \cdot 2^{-1022} = 2^{-52} \cdot 2^{-1022} = 2^{-1074} \Rightarrow \mathbf{Res_{DN} = 2^{-1074}}$$

M_1 y M_2 son mantisas consecutivas

La resolución en la zona de representaciones desnormalizadas es uniforme.

- **Error Absoluto**: (Diferencia entre el valor exacto y el valor aproximado. El error absoluto es siempre menor o igual que la Resolución partida por 2)

▪ Simple precisión:

$$E_{ABS\text{-}DN} = |V_{EX} - V_{AP}| \leq Res_{DN} / 2 = 2^{-23} \cdot 2^{-126} / 2 = 2^{-150} \Rightarrow \mathbf{E_{ABS\text{-}DN} \leq 2^{-150}}$$

▪ Doble precisión:

$$E_{ABS\text{-}DN} = |V_{EX} - V_{AP}| \leq Res_{DN} / 2 = 2^{-52} \cdot 2^{-1022} / 2 = 2^{-1075} \Rightarrow \mathbf{E_{ABS\text{-}DN} \leq 2^{-1075}}$$

- **Error Relativo**: (Error absoluto dividido entre el valor exacto de la representación. Para poder operar con números pertenecientes al sistema considerado, en vez de dividir el error absoluto por el valor exacto se divide por el valor aproximado)

▪ Simple precisión:

$$E_{REL\text{-}DN} = |V_{EX} - V_{AP}| / |V_{EX}| \approx |V_{EX} - V_{AP}| / |V_{AP}| \leq Res_{DN} / (2 \cdot |M| \cdot 2^{-126}) =$$
$$= 2^{-24} \cdot 2^{-126} / |M| \cdot 2^{-126} = 2^{-24} / |M| \leq 2^{-24} / 2^{-23} \Rightarrow \mathbf{E_{REL\text{-}DN} \leq 2^{-1}}$$

- Doble precisión:

$$E_{REL\text{-}DN} = |V_{EX} - V_{AP}| / |V_{EX}| \approx |V_{EX} - V_{AP}| / |V_{AP}| \leq Res_{DN} / (2 \cdot |M| \cdot 2^{-1022}) =$$
$$= 2^{-53} \cdot 2^{-1022} / |M| \cdot 2^{-1022} = 2^{-53} / |M| \leq 2^{-53} / 2^{-52} \Rightarrow \mathbf{E_{REL\text{-}DN} \leq 2^{-1}}$$

REPRESENTACIONES NORMALIZADAS (N):

- **Rango**:
  - Simple precisión:

- Punto **(3)**: **Menor M** (**1**,0 … 00) y **Menor E ≠** 0 (0 … 01) → $1 \cdot 2^{1-127} = 2^{-126}$

- Punto **(4)**: **Mayor M** (**1**,1 … 11) y **Mayor E ≠** 255 (1 … 10) →

$$\rightarrow (2 - 2^{-23}) \cdot 2^{254-127} = (2 - 2^{-23}) \cdot 2^{127}$$

  - Doble precisión:

- Punto **(3)**: **Menor M** (**1**,0 … … … … 00) y **Menor E ≠** 0 (0 … … 01) →

$$\rightarrow 1 \cdot 2^{1-1023} = 2^{-1022}$$

- Punto **(4)**: **Mayor M** (**1**,1 … … … … 11) y **Mayor E ≠** 2047 (1 … … 10) →

$$\rightarrow (2 - 2^{-52}) \cdot 2^{2046-1023} = (2 - 2^{-52}) \cdot 2^{1023}$$

- **Resolución**: (Diferencia entre dos representaciones consecutivas)
  - Simple precisión:

$$Res_N = |M_1 - M_2| \cdot 2^{E-127} = 2^{-23} \cdot 2^{E-127} \leq 2^{-23} \cdot 2^{254-127} = 2^{104} \Rightarrow \mathbf{Res_N \leq 2^{104}}$$

M_1 y M_2 son mantisas consecutivas

  - Doble precisión:

$$Res_N = |M_1 - M_2| \cdot 2^{E-1023} = 2^{-52} \cdot 2^{E-1023} \leq 2^{-52} \cdot 2^{2046-1023} = 2^{971} \Rightarrow \mathbf{Res_N \leq 2^{971}}$$

M_1 y M_2 son mantisas consecutivas

La resolución en la zona de representaciones normalizadas no es uniforme, depende de la zona del rango en la que nos movamos.

- **Error Absoluto**: (Diferencia entre el valor exacto y el valor aproximado. El error absoluto es siempre menor o igual que la Res/2)

- Simple precisión:

$$E_{ABS\text{-}N} = |V_{EX} - V_{AP}| \leq Res_N / 2 = 2^{-23} \cdot 2^{E-127} / 2 \leq 2^{-23} \cdot 2^{254-127} / 2 \Rightarrow \mathbf{E_{ABS\text{-}N} \leq 2^{103}}$$

- Doble precisión:

$$E_{ABS\text{-}N} = |V_{EX} - V_{AP}| \leq Res_N / 2 = 2^{-52} \cdot 2^{E-1023} / 2 \leq 2^{-52} \cdot 2^{2046-1023} / 2 \Rightarrow \mathbf{E_{ABS\text{-}N} \leq 2^{970}}$$

- **Error Relativo**: (Error absoluto dividido entre el valor exacto de la representación. Para poder operar con números pertenecientes al sistema considerado, en vez de dividir el error absoluto por el valor exacto se divide por el valor aproximado)

  - Simple precisión:

$$E_{REL\text{-}N} = |V_{EX} - V_{AP}| / |V_{EX}| \approx |V_{EX} - V_{AP}| / |V_{AP}| \leq Res_N / (2 \cdot |M| \cdot 2^{E-127}) =$$
$$= 2^{-24} \cdot 2^{E-127} / |M| \cdot 2^{E-127} = 2^{-24} / |M| \leq 2^{-24} / 1 \Rightarrow \mathbf{E_{REL\text{-}N} \leq 2^{-24}}$$

  - Doble precisión:

$$E_{REL\text{-}N} = |V_{EX} - V_{AP}| / |V_{EX}| \approx |V_{EX} - V_{AP}| / |V_{AP}| \leq Res_N / (2 \cdot |M| \cdot 2^{E-1023}) =$$
$$= 2^{-53} \cdot 2^{E-1023} / |M| \cdot 2^{E-1023} = 2^{-53} / |M| \leq 2^{-53} / 1 \Rightarrow \mathbf{E_{REL\text{-}N} \leq 2^{-53}}$$

En la Figura 2.4 se muestra mediante un gráfico el resumen de valores de esta representación.

En este estándar hay que resaltar que la diferencia {punto (3) – punto (2)} coincide con la resolución en la zona de representaciones desnormalizadas (2^{-149} ó 2^{-1074}), lo que hace que la diferencia existente entre una representación y la siguiente en el rango comprendido [0,2^{-126}] ó [0,2^{-1022}] sea siempre la misma, 2^{-149} ó 2^{-1074}.

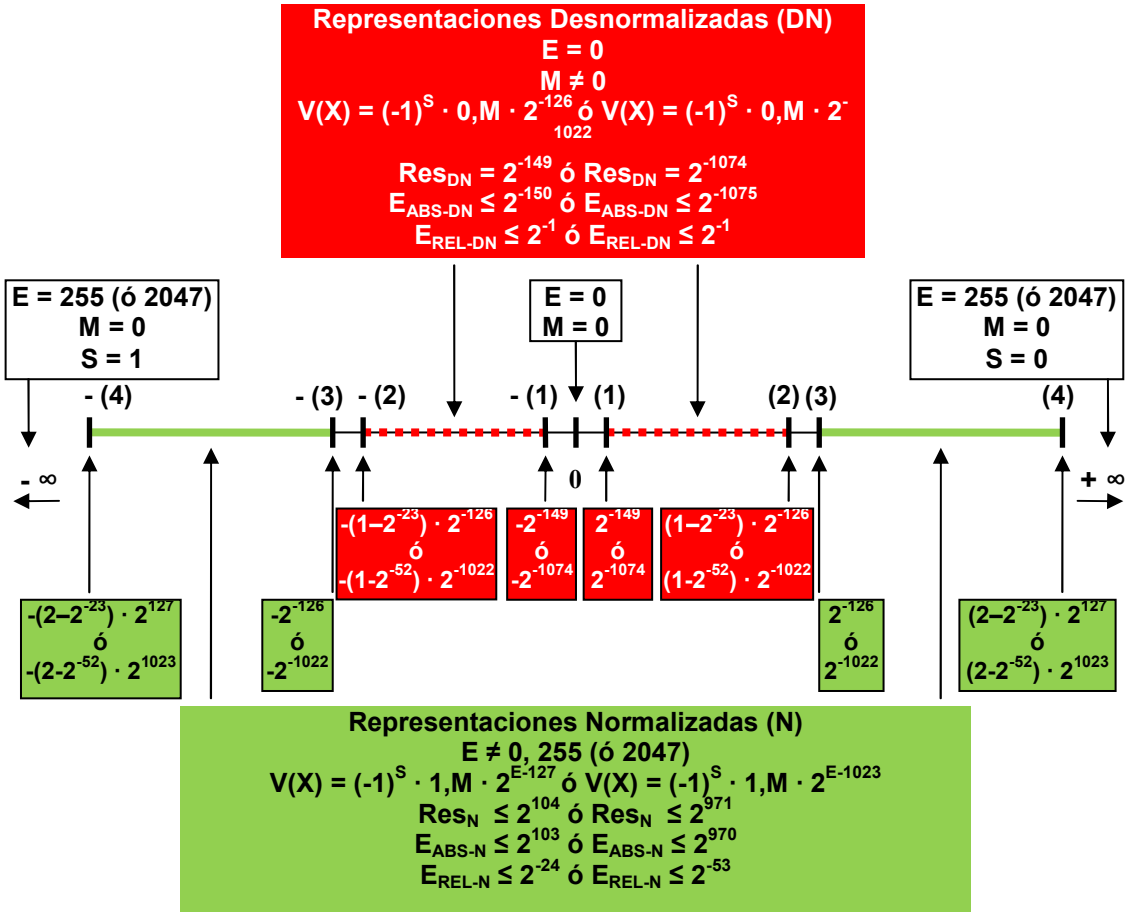


Figura. 2.4. *Resumen de la representación del estándar IEEE P754.*

2.3.3. Sistemas de residuos

El sistema de residuos se basa en el concepto de módulo o residuo de un número. Sea una base m_1, se define como módulo m_1 de X, que se representa como **X mod m_1**, al resto o residuo obtenido al dividir X por m_1.

Con una base m se pueden representar hasta m cantidades distintas.

El interés de estos sistemas estriba en el empleo de varias bases.

$$\left.\begin{array}{l} x_1 = X \bmod m_1 \\ x_2 = X \bmod m_2 \end{array}\right\} X \rightarrow (x_1, x_2)$$

En caso de ser todas las bases primas entre sí, el rango vendrá dado por m1·m2… (el producto de todas las bases.

PROPIEDAD IMPORTANTE DE LOS SISTEMAS DE RESIDUOS:

Las operaciones de suma, resta y multiplicación se realizan de forma independiente sobre los residuos.

$$(x_1, x_2) \overset{+}{*} (y_1, y_2) = ((x_1 \overset{+}{*} y_1) \bmod m_1, (x_2 \overset{+}{*} y_2) \bmod m_2)$$

$$\left.\begin{array}{l} x_1 = X \bmod m_1 \\ y_1 = Y \bmod m_1 \end{array}\right\} \begin{array}{l} \rightarrow X = km_1 + x_1 \\ \rightarrow Y = hm_1 + y_1 \end{array} \left.\right\} \overset{+}{-}{}^{*} \begin{array}{l} \rightarrow X + Y = (k+h) \cdot m_1 + x_1 + y_1 \rightarrow \\ \rightarrow (X+Y) \bmod m_1 = (x_1 + y_1) \bmod m_1 \end{array}$$

Propiedad importante para el diseño de máquinas rápidas, así como en el cálculo con precisión múltiple. También se aplica en el diseño de unidades aritméticas tolerantes a fallos (sirven como códigos redundantes).

INCONVENIENTES FUNDAMENTALES DE LOS SISTEMAS DE RESIDUOS:

1. La conversión desde o hasta la base decimal es compleja.
2. Es difícil realizar la comparación de números.
3. La división resulta muy complicada.

2.4. REPRESENTACIONES REDUNDANTES

2.4.1. Introducción del concepto de redundancia

El objetivo de las representaciones redundantes es salvaguardar la información frente a posibles errores surgidos en su almacenamiento o manipulación. Para ello se añade una información adicional (redundancia) a cada dato, de forma que el conjunto [redundancia, dato] guarde una cierta ley. De esta forma se puede detectar posibles errores e incluso corregirlos si la información redundante es suficiente.

REDUNDANCIA + DATO

↑

Información adicional

Se trata de buscar una ley que permite detectar todos los errores posibles o probables.

Se van a ver los siguientes tipos de códigos redundantes:

- CÓDIGOS DETECTORES DE ERROR
- CÓDIGOS CORRECTORES DE ERROR
- CÓDIGOS POLINOMIALES

2.4.2. Códigos detectores de error

Los **códigos detectores de error** únicamente nos detectan el error o los errores que se produzcan en una combinación. Para que esto sea posible es necesario que se cumplan:

- Que los CÓDIGOS sean NO COMPLETOS. Para que si se produce un error en un bit, se obtenga una combinación no perteneciente al código ← Condición necesaria pero no suficiente.
- Que la distancia mínima del código sea > 1. Se entiende por distancia entre dos combinaciones como el número de bits que hay que modificar en una de ellas para obtener la otra. La distancia de un código es la menor de las distancias entre dos combinaciones cualesquiera del código.

Existen diversos tipos de códigos detectores de error, entre los cuales se tienen:

- CÓDIGOS DE PARIDAD → PAR / IMPAR
- CÓDIGOS DE PESO CONSTANTE → BIQUINARIO / DOS ENTRE CINCO

Peso de una combinación → Nº de unos lógicos de la combinación

Los códigos de paridad se obtienen añadiendo a las combinaciones de los códigos de distancia unidad, un bit llamado *bit de paridad*. Si la paridad obtenida es siempre par diremos que es un **código de paridad par**; si la paridad obtenida es siempre impar diremos que es un **código de paridad impar**.

Entre los códigos de peso constante se encuentra el **código biquinario** que, además, es un código ponderado (cada posición o cifra binaria tiene un peso); emplea siete bits con los pesos “5 0 4 3 2 1 0”.

2.4.3. Códigos correctores de error

Los **códigos correctores de errores** indican, además de la existencia de un error, información de cuál es la cifra o cifras binarias erróneas; de esta forma permiten su corrección sin más que invertir el bit o los bits correspondientes.

Estos códigos se emplean solamente en la transmisión de información y, sobre todo, cuando no es posible volver a enviarla otra vez (en los sistemas digitales que trabajan en tiempo real, en procesos industriales).

Para que un código sea capaz de detectar y corregir al menos un error es necesario que su distancia mínima sea superior a 2. En general, la distancia mínima de un código para que permita corregir errores de n bits ha de ser igual a 2·n+1.

Entre los códigos correctores de error podemos hablar de:

- CONTROL DE DOS ENTRE TRES. Es el modo más sencillo de detectar y corregir errores con una probabilidad alta de acertar. Consiste en transmitir tres veces la misma información y considerar como dato correcto el que se repita más veces.
- CÓDIGOS DE HAMMING. Están basados en la adición de p bits a un código de n bits de distancia mínima 1, obteniéndose un nuevo código de n+p bits. En este nuevo código se realizan p detecciones de paridad en bits seleccionados del mismo, obteniéndose un bit de paridad 1 ó 0 según el número de bits sea par o impar. El conjunto de los p bits de paridad forma un número en el sistema binario natural cuyo equivalente decimal nos indica la posición del bit erróneo. En el caso de que no exista error, dicho número decimal debe ser cero.

2.4.4. Códigos polinomiales

Son los códigos denominados también códigos *redundantes cíclicos* (*CRC*). Permiten detectar errores múltiples en bits consecutivos. Los códigos polinomiales se emplean en aquellas situaciones en que se realiza movimiento de información en serie; Ejemplos: Transmisión de datos; Acceso a un disco o cinta magnética, etc., en todos estos casos, un pequeño fallo, microcorte en la línea de transmisión, mota de polvo en el disco…, produce un error en un conjunto de bits consecutivos.

El procedimiento consiste en añadir al dato su residuo con respecto a una cierta base m_1 [dato, dato mod m_1]. Por ejemplo, para el caso de los diskettes, para protección de la integridad de su información, en cada bloque de información 5 + 128, 5 + 256 ó 5 + 512 bytes se añaden 2 bytes más con el residuo:

mod 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1

↓ polinomio

$P(x) = x^{16} + x^{12} + x^5 + 1$ → estándar CCiTT

Este polinomio permite detectar:

100 % de errores simples

100 % errores dobles

100 % de errores de un número impar de bits

100 % de errores en ráfagas (en una serie sucesiva de bits) de 16 o menos bits

99, 99% de errores en ráfagas de 18 o de más bits

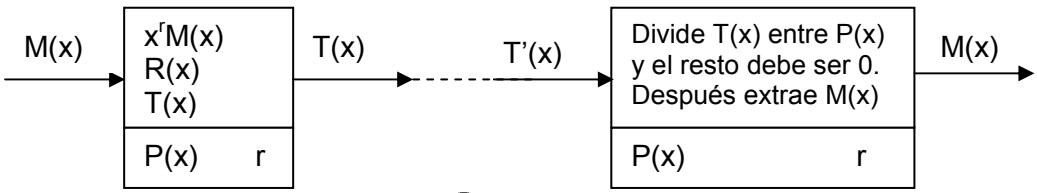
La información que se envía no es M(x), sino T(x), que se obtiene por medio del polinomio generador P(x), de grado r.

M(x) → Información

P(x) → Polinomio generador

r → grado del polinomio generador

$$R(x) \rightarrow \text{resto de } x^r \cdot M(x) / P(x)$$
$$T(x) = x^r M(x) \oplus R(x)$$



DIVIDENDO = COCIENTE x DIVISOR ⊕ RESTO → En esta división (no es una división aritmética; es una división de polinomios en x con coeficientes 0 ó 1) las sumas y restas coinciden (se hacen mediante una operación OR-Exclusiva)

$$x^r M(x) = Q(x)P(x) \oplus R(x)$$
$$x^r M(x) \oplus R(x) = T(x) = Q(x)P(x)$$

Si dividimos T(x) entre P(x) el resto debe ser cero. Esta es la comprobación que se hace. Se puede realizar según van llegando los bits.

Para extraer de T(x) el dato M(x) tan sólo hay que quitar los r bits de menos peso de T(x).

Los polinomios generadores más frecuentes son:

CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$	Se utilizan para la transmisión de tramas de caracteres de 6 bits y genera una FCS de 12 bits.
CRC-16	$x^{16}+x^{15}+x^2+1$	Es habitual en la transmisión de tramas de caracteres de 8 bits.
CRC-CCiTT	$x^{16}+x^{12}+x^5+1$	Es habitual en la transmisión de tramas de caracteres de 8 bits.
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+$ $x^{16}+x^{12}+x^{11}+x^{10}+$ $x^8+x^7+x^5+x^4+ x^2+x+1$	Es muy utilizado en LAN.

2.5. REPRESENTACIONES DE ESTRUCTURAS DE DATOS

Hasta el momento se ha analizado la representación de datos aislados. Ahora bien, estos datos elementales se suelen agrupar en estructuras más o menos complejas.

Estructuras
- Vectores
- Matrices
- Pilas
- Árboles

2.5.1. Representación en la máquina de Von Neumann

En la máquina original de Von Neumann uno de los puntos clave era la no distinción entre datos e instrucciones y, entre los datos, no hacía distinción entre los de un tipo y los de otro. Las estructuras eran IMPLÍCITAS (los programas son los que tienen en cuenta los tipos de datos y las relaciones entre ellos).

2.5.2. Representación en los computadores Burroughs B6700. Datos etiquetados

En estos computadores, los tipos fundamentales de información disponen de representaciones que los identifican. Se asocia a cada información una serie de bits llamados PREFIJO o ETIQUETA (en inglés tag) que define su tipo. → Puede entenderse como una IMPLEMENTACIÓN HARDWARE de la declaración TYPE de muchos lenguajes de alto nivel.

ETIQUETA	INFORMACIÓN

Los BURROUGHS B6700 utilizan etiquetas de tres bits. Permiten diferenciar:

- Datos simple precisión.
- Datos doble precisión.
- Instrucciones.
- Descriptor de matriz,etc…

VENTAJAS DEL ETIQUETADO DE DATOS:

- Al ser el hardware capaz de distinguir los distintos tipos de datos, pueden incrementarse rutinas muy fiables de chequeo y de conversión → simplifican el trabajo del computador (y del programador en ensamblador).
- Se reduce apreciablemente el número de códigos de operación.
- En la fase de puesta a punto de programas se puede disponer de vaciado sensato de memoria.

INCONVENIENTES DEL ETIQUETADO DE DATOS

- Mayor ancho de palabra en rutas de datos y en memoria.
- Se complica el hardware → es casi obligatoria la microprogramación.

2.6. REPRESENTACIONES GRÁFICAS

Las representaciones de los objetos gráficos pueden hacerse en un espacio de dos o tres dimensiones, denominándose respectivamente 2D y 3D.

- En un ESPACIO DE DOS DIMENSIONES (2D), se emplean las representaciones:
  - MATRICIAL (RASTER).
  - VECTORIAL.
- En un ESPACIO DE TRES DIMENSIONES (3D), se emplea la representación:
  - VECTORIAL.

REPRESENTACIÓN MATRICIAL (RASTER)

En la representación matricial se descompone la imagen en una matriz de puntos, asignándose a cada punto uno o varios valores, que determinan sus características visuales.

Cada punto → PIXEL

MAPA DE BITS (BIT MAP)

La CALIDAD DE LA REPRESENTACIÓN depende de:

- DENSIDAD DE PUNTOS (en PUNTOS POR PULGADA – PPP)
- De los POSIBLES VALORES que puede tomar cada PUNTO. Tenemos tres alternativas de posibles valores:

  - BLANCO Y NEGRO.

  - GAMA DE GRIS → diferentes intensidades del negro. Normalmente se asigna un byte por punto → 256 intensidades distintas (0 → BLANCO; 255 → NEGRO).

  - GAMA DE COLORES → Cada punto tiene asignados una serie de valores que expresaron su COLOR e INTENSIDAD. Hay tres formas básicas:

    - RGB (Red, Green y Blue) → Cada punto tiene tres valores que corresponden a las intensidades de tres colores básicos: rojo, verde y azul. Se parte de una imagen negra y se añade una cierta cantidad de cada color básico, por lo que se dice que es un SISTEMA ADITIVO. Si añadimos 100% rojo, verde y azul se produce el blanco. Se adapta bien a los monitores.
    - CMYB (Cyan, Magenta, Yellow y Black) → Cada punto tiene cuatro valores, que corresponden con las intensidades de tres colores primarios: cyan, magenta y amarillo, además del negro. Se trata de un SISTEMA SUSTRACTIVO: se parte del blanco y la intensidad de cada color primario se resta del fondo blanco. El añadir 100% de cada color primario genera un marrón muy oscuro pero no un negro real. Por ello se incluye en color negro para oscurecer las imágenes y añadir detalle. Se adapta bien a las impresoras.
    - HSB (Hue, Saturation y Brightness) → Cada punto viene definido por la CROMINANCIA, la SATURACIÓN y la LUMINANCIA.

  CROMINANCIA → Define el color.

  SATURACIÓN → Define la intensidad del color.

  LUMINANCIA → Define la cantidad de negro presente.

El número de bits que se asigne a cada punto determina las posibilidades de representación; por ejemplo, empleando el sistema RGB y tres bytes por punto, se puede representar hasta 256 x 256 x 256 = 16777216 colores distintos.

REPRESENTACIÓN VECTORIAL

La representación vectorial se caracteriza por dividir la descripción de un objeto gráfico en dos partes:

- Geometría.
- Atributos.

En muchos casos se añade información de:

- Conectividad (relaciona unos objetos con otros).

GEOMETRÍA DE LOS OBJETOS.

La geometría de los objetos se representa mediante una serie de elementos geométricos sencillos, denominados PRIMITIVAS VECTORIALES; las más comunes son:

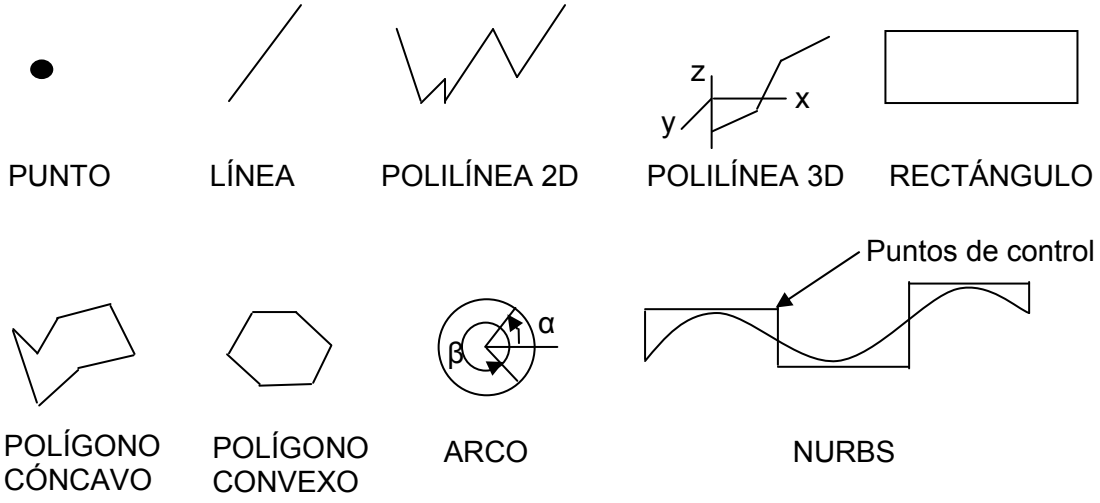


Figura. 2.5. *Primitivas vectoriales*

Hay que descomponer el objeto a representar en un conjunto de estos elementos geométricos para poder representarlo:

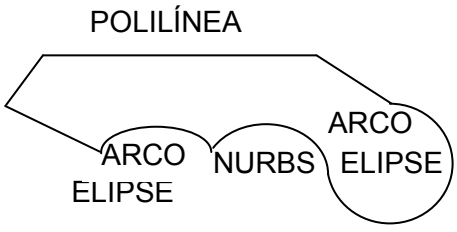


Figura. 2.6. *Objeto vectorial compuesto por cuatro elementos*

Se puede emplear el sistema de coordenadas cartesianas o cualquier otro como el polar. Los valores numéricos de las coordenadas se pueden codificar en cualquiera de los sistemas de representación, siendo muy frecuente el uso de coma flotante.

Las primitivas geométricas más corrientes son:

- PUNTO → Determinado por las dos coordenadas X e Y en 2D y por X,Y,Z en 3D.
- SEGMENTO RECTILÍNEO → Determinado por las coordenadas de los dos puntos extremos.
- QUEBRADA o POLILÍNEA → Formada por un conjunto de segmentos rectilíneos concatenados entre sí. n ≡ nº de tramos

  2D → (1+n)·2 coordenadas

  3D → (1+n)·3 coordenadas

  Aunque la polilínea permite aproximar cualquier curva, no es un buen método cuando tiene grandes curvaturas (cogería muchos segmentos pequeños).
- NURBS (NON-UNIFORM RATIONAL B-SPLINE) → Para representar curvas de muy distinta apariencia, siendo muy flexible en general. Además permite representar exactamente arcos de círculo y de elipse, por lo que se está tendiendo a eliminar esas primitivas.

  Una curva puede expresarse de tres formas:

  - FUNCIÓN EXPLÍCITA → y=f(x)
  - FUNCIÓN IMPLÍCITA → f(x,y)=0
  - DOS FUNCIONES PARAMÉTRICAS → X=X(u), Y=Y(u) → Por sencillez suelen restringirse las funciones X(u) e Y(u) a polinomios en u de un cierto grado n. Se tiene entonces una representación paramétrica polinomial de grado n. Esta representación es la que utiliza la NURBS.

  La NURBS no utiliza directamente los coeficientes de los polinomios X(u) e Y(u) para representar una curva; está basada en una serie de puntos de control que atraen a la curva y una serie de nudos, esto es, una serie creciente de valores del parámetro u. La curva se aproxima a la polilínea

de control formada por los puntos de control. La definición concreta de una NURBS requiere los siguientes elementos:

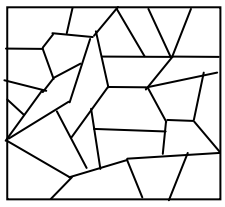
- Orden de la curva (grado de los polinomios+1).
- Número de nudos → Ha de ser igual al número de puntos de control más el orden.
- Lista de nudos.
- Lista de puntos de control con sus coordenadas.

El orden de la curva determina la suavidad de la unión de dos tramos adyacentes: a mayor orden mayor suavidad de unión (aumenta precisión y también aumenta el volumen del cálculo) → orden 3 ó 4 los más empleados.

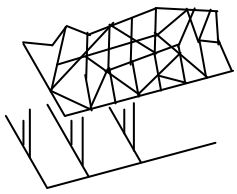
Las NURBS pueden ser 2D o 3D, existiendo también las superficies NURBS, definidas mediante dos parámetros: u y v.

- RECTÁNGULO → Figura 2D que viene definida por las coordenadas de dos vértices opuestos. Sus lados son paralelos a los ejes X e Y.
- POLÍGONO → Figura 2D cerrada limitada por segmentos rectilíneos. Para n lados requiere 2n coordenadas.
- ARCO → Figura 2D de arco de círculo o de elipse. Los datos necesarios para definir un arco de elipses con sus ejes paralelos a X e Y son:
  - Centro.
  - Radios (uno para el círculo y dos para la elipse).
  - Ángulo α de arranque y ángulo β de fin.

    $\alpha = 0$ y $\beta = 2\Pi$ → Curva cerrada
- MALLAS DE POLÍGONOS → Se emplean mucho en cartografía y en representaciones 3D ya que se puede aproximar cualquier superficie mediante una malla de polígonos.



MALLA DE POLÍGONOS 2D



Malla 3D de polígonos regulares

Alturas que definen la malla

Figura. 2.7. *Mallas de polígonos*

Existen tres formas básicas de representar las mallas de polígonos:

- LISTA DE POLÍGONOS → Se almacenan individualmente todos los polígonos, mediante sus coordenadas. Método poco compacto por existir vértices repetidos. No suministra ninguna relación de vecindad. Para obtenerla hay que comparar los vértices de un polígono con todos los demás vértices.
- PUNTEROS A LISTA DE NODOS → Método más compacto que el anterior, pero tampoco establece relaciones de vecindad. Se emplean dos estructuras:
  - Lista de nodos con las coordenadas de todos los vértices de la malla.
  - Lista de polígonos. En la descripción de cada polígono se usan punteros a la lista de nodos, en vez de las coordenadas de los mismos.
- PUNTEROS A LA LISTA DE BORDES → Este método almacena información de vecindad. Se basa en tres estructuras:
  - Lista de nodos con las coordenadas de todos los vértices de la malla.
  - Lista de bordes. Por cada borde se usan dos punteros a los correspondientes nodos. Además cada borde tiene uno o dos punteros que identifican los polígonos que tiene ese borde, de forma que se conocen los polígonos vecinos.
  - Lista de polígonos. En la descripción de cada polígono se usan punteros a la lista de bordes, en vez de las coordenadas de los mismos.

ATRIBUTOS DE LOS OBJETOS GRÁFICOS.

Los atributos son informaciones adicionales sobre los elementos geométricos que componen un objeto gráfico. Los principales atributos de visualización (pueden haber otros tipos) son los siguientes:

- ESTILO DE LÍNEA: continua, de rayas, de punto y raya, etc.
- GROSOR DE LA LÍNEA.
- COLOR DE LA LÍNEA.

- RELLENO O TEXTURA DE POLÍGONOS O SUPERFICIES. El relleno se refiere a una figura o patrón que se repite reiteradamente cubriendo toda la superficie. Puede venir definido en forma vectorial o como un mapa de bits.
- COLOR DE LOS POLÍGONOS O SUPERFICIES.
- PARPADEO.
- TRANSFORMACIONES: traslación, rotación, escalado, sesgado, estirado, simetría,etc.

CONECTIVIDAD.

La conectividad expresa las relaciones entre los distintos objetos gráficos. Existen cuatro tipos de conectividad:

- La AGRUPACIÓN permite definir objetos complejos como un conjunto de primitivas vectoriales y de agrupaciones.
- La VECINDAD expresa que los objetos tienen un elemento común (cara, segmento, punto). Muy empleado en las mallas de polígonos.
- La JERARQUÍA permite establecer dependencias jerárquicas entre objetos gráficos.
- La SUPERPOSICIÓN es muy importante en imágenes que están en un mismo plano (en 2D o en 3D). Permite precisar qué elementos ocultan a otros.

FORMATOS

Son variados y complejos los formatos de representación gráfica.

FORMATO PARA IMÁGENES MATRICIALES:

Las imágenes matriciales suelen ser rectangulares, se componen de los siguientes elementos:

- CABECERA. Se describe la imagen y su forma de almacenamiento.
- MATRIZ. n x m datos. Cada dato de la matriz tendrá el número de bits que corresponda al tipo de imagen deseada (un bit para imágenes en blanco y negro, 8 bits para una gama de grises, etc.).

Las imágenes matriciales son de gran tamaño, por lo que es muy importante el uso de técnicas de compresión.

FORMATO PARA IMÁGENES VECTORIALES:

El formato de las imágenes vectoriales se basa en cuatro componentes:

- CABECERA. Describe la imagen.
- FORMATO DE REPRESENTACIÓN DE LAS PRIMITIVAS VECTORIALES.
- ATRIBUTOS.
- FORMATO DE CONECTIVIDAD.

El formato de las primitivas vectoriales suelen ser de longitud variable (tienen longitud variable algunas). Contienen los siguientes campos:

- Código que especifica la primitiva.
- Longitud.
- Parámetros (coordenadas otros parámetros).

CÓDIGO	LONG.	PARÁMETROS

FORMATO DE LA PRIMITIVA VECTORIAL

Muchos atributos son de utilización poco frecuente. Además, los atributos suelen afectar a más de una primitiva. Por ello no se suelen incluir los atributos en el formato de las primitivas, puesto que sería muy poco eficaz. Se utiliza el concepto de ENTORNO DE ATRIBUTOS, que se refiere a los valores que tienen en un instante todos los atributos. Este entorno puede modificarse con una PRIMITIVA DE MODIFICACIÓN DE ENTORNO.

ENTORNO	LONG.	VARIABLE	PARÁMETROS

FORMATO DE PRIMITIVA DE MODIFICACIÓN DE ENTORNO

Todas las primitivas que se encuentren entre dos modificadores de entorno sucesivos se representarán con los mismos parámetros.

POLILÍNEA
NURBS
NURBS
} Todas tienen los mismos atributos, que habrán sido definidos antes o que tienen los valores por defecto.

MODIFICADOR DE ENTORNO (p.e. COLOR LINEA R=0, G=0 y B=255)

RECTÁNGULO
POLÍGONO
POLILÍNEA
} Este bloque mantiene los atributos del anterior, pero el color de las líneas se ha cambiado al azul.

MODIFICADOR DE ENTORNO (p.e. COLOR RELLENO R=0, G=255 y B=0)

POLÍGONO
NURBS
} Este bloque tiene los mismos atributos que el anterior, menos el color de relleno que se ha cambiado al verde.

TÉCNICAS DE COMPRESIÓN DE IMÁGENES

Dadas las características de los formatos de imágenes, es conveniente aplicar técnicas de compresión de imágenes. Los algoritmos más usuales para la compresión de imágenes son:

- Los **JPEG** (del *Joint Photographic Experts Group*) para imágenes fijas. Se basa en dividir la imagen en celdas de 8x8 píxeles. Cada celda se transforma en 64 frecuencias espaciales mediante la transformación DTC (*Discrete Cosine Transform*), que es parecida a la transformación de Fourier: Dado que la mayoría de la información de la imagen se encuentra en las frecuencias bajas y que el ojo es poco sensible a las altas frecuencias, se puede cuantificar de forma menos precisa estas altas frecuencias, tomando el valor cero para todas aquellas que no superen un umbral α. La secuencia de frecuencias así truncada, que tiene largas cadenas de ceros, se comprime con un algoritmo RLC (*Run Lengh Coding*) y un algoritmo VLC (*Variable Length Coding*). Cuanto más alto sea α mayor será la compresión alcanzada, pero peor la calidad de la imagen comprimida.
- Los **MPEG** (del *Motion Pictures Experts Group*) para vídeo. Se basan en el concepto de diferencia espacio-temporal. Se trata de determinar objetos móviles que aparecen en cuadros o imágenes sucesivas. Para ello se dividen los cuadros en objetos de 16x16 píxeles y se busca cada objeto en el cuadro siguiente. Esta búsqueda se restringe a un entorno cercano a la posición inicial, puesto que se supone que un objeto no se mueve mucho de un cuadro al siguiente. En caso de acierto basta con almacenar, para el cuadro siguiente, el vector de movimiento, esto es el incremento en píxeles [Δx, Δy] entre la posición actual y la del cuadro siguiente.