Tema 2

Codificación binaria de la información

Se ha indicado en el tema introductorio que las computadoras digitales sólo manejan información en forma de ceros y unos. Esto es así porque los dispositivos electrónicos usados para construir las computadoras digitales se diseñan para trabajar en torno a dos valores de tensión. Uno de ellos se asocia con el estado 1 y el otro con el 0. Estos circuitos son capaces de cobijar los valores cero y uno por lo que constituyen celdas de memoria que almacenan estos valores binarios.

Sin embargo estamos acostumbrados a ver que las computadoras permiten utilizar información de muchos tipos: texto, gráficos, sonidos. Se mostrará en este tema que para poder representar la información con ceros y unos es necesario usar un código.

2.1 Sistema binario de numeración

Los números se pueden expresar en distintos sistemas de numeración. Como es sabido, el más usual es el sistema en base 10. En este sistema, llamado decimal, se interpretan las cifras como coeficientes de un polinomio en potencias de 10:

$$D\,|_{10)} = d_n \cdot 10^n + \cdots + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

donde los dígitos d_n a d_0 constituyen el número en base 10. Para aclarar ideas considérese el número 1492. En base 10 se interpreta como: un millar más cuatro centenas más nueve decenas más dos unidades, o lo que es lo mismo:

$$1492 = 1 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 2 \cdot 10^0$$

Los dígitos o coeficientes del polinomio son las cifras 1, 4, 9 y 2. En el sistema decimal se trabaja con cifras del 0 al 9. En el sistema binario la base es el 2, por lo que sólo existen dos posibles coeficientes: el cero y el uno. La interpretación de un número escrito en base dos es la

misma que en decimal, pero cambiando la base:

$$B\ |_{2)} = b_n \cdot 2^n + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0 \tag{2.1}$$

así el número 01001 en base dos (indicado frecuentemente como 01001 $|_{2)}$) se interpreta como la cantidad

$$01001\ |_{2)} = 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9\ |_{10)}$$

Obsérvese que se ha indicado la base utilizada mediante un subíndice. Habitualmente se trabaja en base 10, por lo que se omite dicho subíndice. A los coeficientes d_i de la representación digital se les llama dígitos, a los de la binaria b_i se les llama BITS[1].

Los sistemas decimal y binario son sólo dos ejemplos de una infinidad de posibles sistemas con base b . En general, el Teorema Fundamental de la Numeración proporciona el valor decimal de una cantidad expresada en base b por ciertas cifras. Pero antes de presentar la fórmula general conviene considerar algunos casos particulares. Por ejemplo, la cantidad 23.75 es interpretada como dos decenas más tres unidades más 7 décimas más cinco centésimas. En notación matemática esto se expresa mediante

$$23.75 = 2 \cdot 10^1 + 3 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

A menudo una cantidad como 23.75 se descompone en dos: la llamada parte entera (23) y la parte fraccionaria (.75). La parte entera queda a la izquierda del punto usado como separador y representa unidades completas. La parte fraccionaria queda a la derecha del punto y representa una cantidad menor que una unidad, de ahí el nombre de *fraccionaria*. La parte entera contiene los coeficientes que multiplican a potencias no negativas. La parte fraccionaria abarca los coeficientes de potencias negativas.

En otros sistemas de numeración también pueden usarse potencias positivas y negativas de la base. Por ejemplo en binario o base dos la cantidad 10.11 $|_{2)}$ tiene parte entera 10 $|_{2)}$ y parte fraccionaria .11 $|_{2)}$. La cantidad representada en decimal es

$$1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} = 2 + 0 + \frac{1}{2} + \frac{1}{4} = 2 + 0.5 + 0.25 = 2.75\ |_{10)}$$

Obsérvese que la parte entera es 2 y la parte fraccionaria es .75 y que por otro lado 10 $|_{2)}$= 2 $|_{10)}$ y .11 $|_{2)}$= .75 $|_{10)}$. Es decir las cantidades indicadas por las partes entera y fraccionaria no se modifican al cambiar de base.

Ahora ya se puede dar el salto a una fórmula general. Sean los guarismos g_i que representan un número G en base b de tal modo que

$$G\ |_{b)} = g_p g_{p-1}\ \cdots\ g_0 . g_{-1}\ \cdots\ g_{-n},$$

donde el punto separa la parte entera de la fraccionaria (en la base en cuestión), entonces según el Teorema Fundamental de la Numeración el valor decimal es

$$D\ |_{10)} = \sum_{i=-n}^{i=p} g_i \cdot b^i \tag{2.2}$$

[1] del inglés **b**inary dig**it**

Como ejemplo tómese $32.6\ |_{5)}$, es decir, el número 32.6 en base 5. La cantidad que representa dicho número expresada en el sistema decimal es

$$D\ |_{10)} = 3 \cdot 5^1 + 2 \cdot 5^0 + 6 \cdot 5^{-1} = 15 + 2 + \frac{6}{5} = 18.2\ |_{10)}$$

De este modo se puede concluir que $32.6\ |_{5)} = 18.2\ |_{10)}$.

El funcionamiento de las computadoras actuales está basado en el uso del sistema de numeración binario. Otros sistemas que han tenido cierta importancia son el octal, cuya base es 8 y por tanto usa las las cifras del 0 al 7 y el hexadecimal, de base 16. Este último sistema plantea un problema a la hora de escribir números y es que son necesarios 16 dígitos distintos. En el sistema decimal sólo hay diez dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Para obtener los restantes seis dígitos se utilizan las letras A, B, C, D, E y F para representar los valores 10, 11, 12, 13, 14 y 15 respectivamente. De este modo $3E.1\ |_{16)}$ representa la siguiente cantidad expresada en el sistema decimal

$$3 \cdot 16^1 + 14 \cdot 16^0 + 1 \cdot 16^{-1} = 48 + 14 + \frac{1}{16} = 62.0625\ |_{10)}$$

2.1.1 Conversiones

Las conversiones permiten obtener las cifras correspondientes a una misma cantidad en distintas bases. En informática la conversión más usada es la de decimal a binario y viceversa, por ello se van a describir con detalle.

Al realizar una conversión hay que tener en cuenta que la cantidad representada no cambia, lo que se modifica es la forma en que se simboliza dicha cantidad.

Dado un número en base dos, es fácil hallar su equivalente decimal sin más que aplicar la relación (2.2). Es decir, realizando la suma de potencias de dos. Por ejemplo el número $1\ |_{2)}$ equivale a $1 \cdot 2^0$ que es el uno en base 10, por tanto $1\ |_{2)} = 1\ |_{10)}$. Del mismo modo se obtiene que $101\ |_{2)} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5\ |_{10)}$.

El paso contrario, es decir, el paso de de decimal a binario, es más elaborado y por ello se van a tratar en primer lugar las cantidades enteras. Una posibilidad es dividir sucesivamente el número por dos. Los restos de las divisiones dan el número en binario, leídos en sentido ascendente, tal como se muestra en la figura 2.1 donde se obtiene la representación en base dos del número $214\ |_{10)}$. Una vez obtenida la representación binaria resulta fácil comprobar que el resultado es el correcto sin más que aplicar la fórmula 2.2 lo cual debe dar como resultado el número original. Compruebe que este es el caso para el ejemplo de la figura 2.1.

Resulta interesante conocer las potencias de dos de ciertos números para así realizar conversiones de forma rápida sin usar la regla de la división.

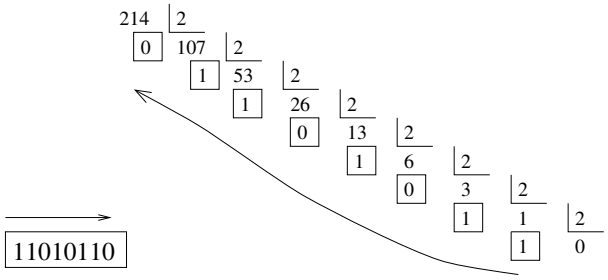


Figura 2.1: Método de la división larga para la conversión de base 10 a base 2.

potencia	decimal	binario
2^0	1	1
2^1	2	10
2^2	4	100
2^3	8	1000
2^4	16	10000
2^5	32	100000
2^6	64	1000000
2^7	128	10000000
2^8	256	100000000
2^9	512	1000000000
2^{10}	1024	10000000000

Ciertas conversiones resultan cómodas de este modo. Por ejemplo, para pasar a binario el número decimal 18 basta con observar que $18\ |_{10)} = 16 + 2$ y por tanto igual a $2^4 + 2^1$, por lo que la representación en base dos resulta $10000\ |_{2)} + 10\ |_{2)} = 10010\ |_{2)}$; es decir, $18\ |_{10)} = 10010\ |_{2)}$.

Los números fraccionarios también pueden pasarse con facilidad de base 10 a base 2. En tal caso se han de calcular los coeficientes de una suma de potencias con exponentes negativos. Para ello en lugar de la división se usa el método de la multiplicación. El método se presenta fácilmente con la ayuda de un ejemplo. Considere el número en decimal $0.6875\ |_{10)}$. Si se multiplica por dos resulta $0.6875\ |_{10)} \cdot 2 = 1.375\ |_{10)}$ que tiene parte entera igual a 1. Esto significa que el número $0.6875\ |_{10)}$ contiene a la primera potencia negativa de dos que es $2^{-1}\ |_{10)} = 0.5\ |_{10)}$. Este 1 se toma como parte de la expresión binaria. El proceso continúa tomando la parte decimal restante y volviendo a multiplicar por dos. Esto se repite hasta que desaparezcan todos los decimales. La tabla siguiente muestra el resultado para la cantidad tomada como ejemplo

cantidad	cantidad × 2	parte entera	parte fraccionaria
0.6875	1.375	1	0.375
0.375	0.75	0	0.75
0.750	1.5	1	0.5
0.5	1	1	0

Los coeficientes fraccionarios de la representación binaria se obtienen de la tercera columna. De este modo se puede afirmar que $0.6875\ |_{10)} = 0.1011\ |_{2)}$. En efecto, basta con comprobar que la suma de potencias $0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4}$ es igual a la cantidad $0.6875\ |_{10)}$.

La relación del método de la multiplicación con el método de la división es evidente con la diferencia de que en lugar de los restos se van tomando las partes enteras resultantes.

Dado que el método de la división permite pasar a binario cantidades decimales enteras y el de la multiplicación cantidades decimales fraccionarias se está ya en condiciones de pasar a binario cualquier cantidad representada en sistema decimal. Tómese por ejemplo la cantidad decimal 278.5625 $|_{10)}$, la parte entera se puede expresar como 278 $|_{10)}$= 100010110 $|_{2)}$, mientras que la parte fraccionaria es .5625 $|_{10)}$= .1001 $|_{2)}$ por lo que 278.5625 $|_{10)}$= 100010110.1001 $|_{2)}$.

Para finalizar conviene llamar la atención sobre el número de coeficientes fraccionarios en las distintas representaciones de una cantidad. En los ejemplos mostrados no se necesitan infinitos coeficientes para expresar las cantidades consideradas. Esto no siempre es así y un simple ejemplo lo deja bien claro. Considere el número 0.1 $|_{3)}$ es decir 0.1 en base 3. Pasando a decimal se tiene que

$$0.1\ |_{3)} = 0 \cdot 3^0 + 1 \cdot 3^{-1} = 0 + \frac{1}{3}\ |_{10)}$$

como es sabido la cantidad $\frac{1}{3}$ expresada en el sistema decimal tiene infinitos decimales por lo que no es posible representarla a menos que se use algún truco como escribir $1.3\widehat{3}$. En la computadora lo normal es escribir tantos decimales como sea posible dentro de la memoria. Este problema se tratará más adelante en este tema.

2.2 Codificación de números enteros

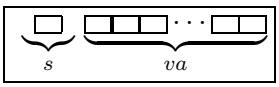
La memoria de una computadora está formada por muchas células elementales agrupadas en registros. Cada celda sólo pueden representar los números 0 ó 1, pero al considerar un grupo de ellas (el registro) se pueden codificar números mayores.

Para representar un número entero positivo mediante un conjunto de bits se puede usar la codificación del número en base 2. De este modo, si se dispone de registros de 16 celdas es posible almacenar los números entre el 0 y el $2^{16} - 1 = 65535$. Si el número de celdas es mayor, digamos 20, es posible ampliar el intervalo representable desde cero hasta $2^{20} - 1 = 1048575$.

La computadoras digitales utilizan registros de una anchura determinada, por lo que existen límites a los números que es posible representar. Este hecho característico de los sistemas digitales no sorprenderá a quien haya utilizado una calculadora.

La representación mostrada que utiliza la base dos es fácil de entender y de usar, por lo que es largamente utilizada. Ahora bien, ¿y los números negativos?. El signo menos tendría que poder expresarse como un cero o un uno. La idea más simple es usar la celda más a la izquierda del registro para indicar el signo del número. Ésta y otras posibles representaciones se explican a continuación.

1. **Modo signo-valor absoluto**. Se reserva una celda del registro (normalmente la que está más a la izquierda) para el signo. Los números positivos tienen un valor 0 para el bit que ocupa esa celda y los negativos un valor 1. Este sistema queda descrito de forma gráfica del siguiente modo:



donde s simboliza el bit de signo, y va la representación en base dos del valor absoluto.

De este modo, con un registro de 16 unidades se pueden representar los números del -32767 hasta el 32767. En general, para un número de bits en la palabra igual a n se pueden representar los enteros comprendidos en el intervalo $\left[-(2^{n-1}-1),\ 2^{n-1}-1\right]$.

Como ejemplo considérense la codificación en un registro de 8 celdas de los números 5 y -4:

$$5 = \boxed{00000101},\ -4 = \boxed{10000100}$$

El número de celdas n del registro afecta a la representación de los negativos, pues el bit de signo debe quedar a la izquierda. A modo de ejemplo véase la representación de los números 5 y -4 codificados en un registro de 6 celdas

$$5 = \boxed{000101},\ -4 = \boxed{100100}$$

Otra característica de este sistema es que tiene dos ceros: el positivo $\boxed{0\ 00\ \ldots\ 0}$ y el negativo $\boxed{1\ 00\ \ldots\ 0}$, que, lógicamente, representan la misma cantidad, cero.

2. **Complemento a 1**. Nuevamente se reserva el primer bit para el signo. El resto de la codificación es: el número en base dos si el signo es positivo o el complemento bit a bit del valor absoluto si el signo es negativo. Es decir, los números positivos se representan igual que en el sistema anterior, mientras que, los negativos sufren el cambio de los unos por ceros y los ceros por unos (excepto el bit de signo).

   Si se utilizan ocho celdas, los números 5 y -4 se representan en complemento a uno del siguiente modo:

   $$5 = \boxed{00000101},\ -4 = \boxed{11111011}$$

   Utilizando seis celdas resulta:

   $$5 = \boxed{000101},\ -4 = \boxed{111011}$$

   Este sistema también tiene dos ceros: el positivo $\boxed{00\ \ldots\ 0}$ y el negativo $\boxed{11\ \ldots\ 1}$. Para un número de celdas del registro igual a n se pueden representar los enteros comprendidos en el intervalo $\left[-(2^{n-1}-1),\ 2^{n-1}-1\right]$.

3. **Complemento a 2**. Los dos sistemas anteriormente vistos presentan dificultades a la hora de realizar sumas y restas mediante circuitos lógicos, por lo que se ha buscado otro método. El complemento a dos de un número es igual al complemento a uno más uno; es decir, los números positivos se representan igual que en el sistema signo-valor absoluto, pero los números negativos se codifican con el bit de signo igual a uno y el resto como 1 más el complemento bit a bit del valor absoluto en base dos. Los números negativos se representan pues igual que en complemento a 1 sumándoles 1.

   El ejemplo habitual con ocho celdas queda:

   $$5 = \boxed{00000101},\ -4 = \boxed{11111100}$$

   mientras que utilizando seis celdas resulta:

   $$5 = \boxed{000101},\ -4 = \boxed{111100}$$

   Los números que se pueden representar usando un registro de n células son los enteros comprendidos en el intervalo: $\left[-2^{n-1},\ 2^{n-1}-1\right]$. Se observa que en el lado de los negativos

hay un número más. Esto es consecuencia de que sólo existe un cero: $\boxed{00\ldots0}$ que usa el espacio de un número positivo. El número -2^{n-1} tiene siempre la representación especial $\boxed{10\cdots0}$ que corresponde al cero negativo en los métodos anteriores.

2.3 Codificación de caracteres

Muchas veces la información que la computadora ha de procesar no son números sino caracteres textuales tales como:

- Letras mayúsculas y minúsculas.
- Dígitos. Para escribir las cifras de un teléfono, o la hora, o números.
- Signos: ? ( ) , { } ♡ [ ] + $ ...
- Códigos sin representación gráfica, pero con funciones de control, por ejemplo el retorno de carro de la impresora, el aviso sonoro, los códigos de mensaje recibido, fin de mensaje, fin de archivo, etc. que se transmiten entre dispositivos.

Para transmitir y almacenar información de este tipo se creó el ASCII (American Standard Code for Information Interchange), que es un código que asigna arbitrariamente un número entero a cada signo. Los números son luego representados en base dos para poder ser tratados por la computadora.

El ASCII usa 7 bits, por lo que se pueden representar $2^7 = 128$ signos distintos. Éstos incluyen números, el alfabeto inglés en mayúsculas y minúsculas, signos matemáticos y de puntuación y algunos caracteres de control. Una versión posterior de este código es el ASCII extendido, que usa 8 bits, por lo que se pueden representar 256 signos. Esto pemite incluir la letra ñ y otras de diversos alfabetos.

El conjunto de signos incluidos en el código recibe el nombre de TABLA ASCII. En la tabla 2.1 se muestran los códigos ascii correspondientes a algunos caracteres. Obsérvese que las letras van en orden alfabético a excepción de la letra **ñ**. Las mayúsculas están colocadas en la tabla antes que las minúsculas. También es de interés observar que existen códigos sin representación gráfica, que se han marcado como cc pues son códigos de control.

Los caracteres de control son combinaciones de ocho dígitos binarios, al igual que el resto de la tabla ASCII. Lo que los hace en cierto modo especiales es el modo en que los periféricos los usan. Al contrario que las letras y signos que aparecen por la pantalla o la impresora, los códigos de control realizan cierta función sobre el dispositivo. La tabla 2.2 resume algunas de las funciones más usadas. La primera columna indica el número dentro de la tabla ascii, la segunda es un nombre abreviado de la función que realiza, la cual se explica en la tercera columna.

Nro.	Signo
0	*cc*
1	*cc*
2	*cc*
3	*cc*
⋮	⋮
46	.
47	/

Nro.	Signo
48	0
49	1
50	2
51	3
⋮	⋮
57	9
58	:

Nro.	Signo
63	?
64	@
65	A
66	B
67	C
⋮	⋮
90	Z

Nro.	Signo
⋮	⋮
97	a
98	b
99	c
100	d
101	e
102	f

Nro.	Signo
⋮	⋮
161	í
162	ó
163	ú
164	ñ
165	Ñ
166	$\underline{a}$

Tabla 2.1: Fragmentos de la tabla ASCII. Los signos indicados como *cc* son caracteres no imprimibles.

		Transmisiones			Pantalla o impresora
1	SOH	comienzo de cabecera	7	BEL	señal audible
2	STX	comienzo de texto	8	BS	retroceso
3	ETX	fin de texto	9	HT	tabulación horizontal
4	EOT	fin de la transmisión	10	LF	avance de línea
5	ENQ	petición de transmisión	11	VT	tabulación vertical
6	ACK	reconicimiento de transmisión	13	CR	retorno de carro

Tabla 2.2: Algunos códigos de control.

2.4 Números fraccionarios

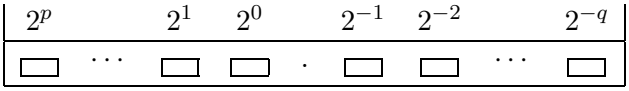
En el sistema decimal de numeración las cantidades no enteras se indican mediante potencias negativas de la base. De este modo la cantidad $3.2\,|_{10)}$ es interpretada como 3 unidades más dos décimas.

Los números con parte no entera también se pueden expresar en binario, usando para ello potencias negativas de dos. Por ejemplo, el número $101.11\,|_2$ representa la cantidad $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 4 + 1 + 0.5 + 0.25 = 5.75\,|_{10)}$.

En las computadoras digitales se plantea el problema de representar el punto fraccionario y de interpretarlo. A continuación se describen dos posibles soluciones.

2.4.1 Punto fijo

Se reservan algunas celdas del registro para la parte entera del número en binario y otras para la fraccionaria; es decir, a partir de una posición predeterminada, los coeficientes multiplican potencias negativas de dos. Este sistema queda descrito de forma gráfica del siguiente modo:



donde se han reservado $p+1$ celdas para la parte entera y q para la fraccionaria. No se ha tenido en cuenta el problema del signo.

Considérese a modo de ejemplo que se tiene un registro de ocho celdas, y el punto decimal está colocado antes de las dos últimas celdas, entonces el registro

00000110

se interpreta como el número 000001.10 $|_{2)}$ que representa la cantidad dada por

$$1 \cdot 2^0 + 1 \cdot 2^{-1} = 1.5\ |_{10)}$$

Del mismo modo, el registro

00001001

representa el número 000010.01 $|_{2)}$ que corresponde a la cantidad decimal 2.25 $|_{10)}$.

Los circuitos de la UAL se construyen de forma que cada parte del registro es tratada correctamente de acuerdo a la posición del punto decimal que es fija. Las operaciones se realizan adecuadamente en cada parte del registro.

En una computadora de punto fijo el número de coeficientes fraccionarios q es fijo por la propria construcción de los circuitos y no se puede cambiar. Esto es un problema pues no se puede trabajar con números muy dispares (por ejemplo 25000.0 y 0.0003). Para comprender esta afirmación piense que $n = p + q + 1$ es un número prefijado por la tecnlogía del momento. Hace años se utilizaban computadoras con $n = 8$ de forma casi exclusiva. Con el paso del tiempo se ha avanzado pasando por $n = 16$, $n = 32$ y recientemente $n = 64$. En cualquier caso si se decide incrementar q eso conlleva un decremento de p y viceversa. Debido a esto, este sistema de punto fijo se usa poco en la actualidad.

2.4.2 Punto flotante

El número (N) a representar se transforma en la pareja (M, E) de forma que $N = M \cdot 2^E$. La primera cantidad, M es llamada mantisa, la segunda E el exponente. Para aclarar el método es mejor pensar en la representación decimal. Un número en base diez con decimales tal y como 340.126 puede expresarse como $3.40126 \cdot 10^2$ o como $0.0340126 \cdot 10^4$ o como $0.340126 \cdot 10^3$.

Se denomina AJUSTE FRACCIONARIO O NORMALIZACIÓN al proceso de selección de la mantisa y el exponente de forma que cumpla con las especificaciones. La normalización tiene por objeto elegir una de las muchas posibles parejas de mantisa y exponente. Una normalización muy usada consiste en imponer que la mantisa tenga parte entera nula y que su primer coeficiente fraccionario no lo sea. Aplicando esta regla al ejemplo decimal anterior se tiene que $0.340126 \cdot 10^3$ es la representación correcta pues la mantisa cumple las normas establecidas.

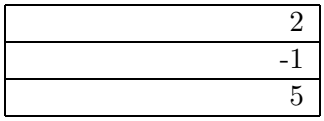
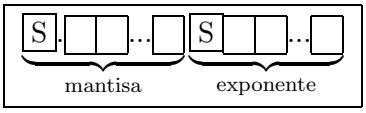


Figura 2.2: Representación de un vector de enteros en la memoria

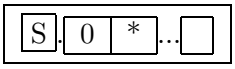
En representación binaria se tiene que $\mid M \mid$ comienza por $0.1\mid_{2)}$. De este modo queda fijado M y por tanto E.

Una vez que el ajuste fraccionario o normalización se ha llevado a cabo cada número M y E se codifica en un trozo de registro. Queda por resolver el problema de representar el signo tanto de la mantisa como del exponente. Para ello basta con utilizar alguno de los métodos vistos, como el signo-valor absoluto o complemento a dos.

Existen varios formatos de punto flotante con sus reglas específicas como IEEE 754. No se va a describir con detalle ninguno de estos formatos, baste saber que todos ellos son muy similares al caso comentado, que puede mostrarse de forma gráfica como:



Dado un número n de celdas el punto flotante permite representar la misma cantidad de valores que el punto fijo pues las posibles combinaciones con n son las mismas. No hay que olvidar que hay combinaciones prohibidas como



Ahora bien, el intervalo de valores representables es mayor en punto flotante, pues el exponente permite alcanzar números muy altos si es positivo o muy cercanos a cero si es negativo. Esta es la gran ventaja del punto flotante frente al fijo y la razón de su uso.

2.5 Otros tipos de información

2.5.1 Vectores

Los vectores son conjuntos de elementos del mismo tipo. Por ejemplo un vector de números enteros de dimensión tres no es más que un trío de números enteros. Para representar un vector en la computadora se pueden usar varias técnicas. La más simple consiste en colocar los elementos del vector en la memoria de forma individual, pero ocupando registros consecutivoss. De este modo el vector de enteros $v = (2, -1, 5)$ se representa simplemente colocando en tres registros de la memoria los números enteros 2, -1 y 5 como se indica en la figura 2.2.

Para manejar los vectores basta con saber cuál es el primer registro utilizado y cuál es la dimensión del vector. Con estos dos datos es posible acceder a cada uno de los elementos del vector.

1
0
-2
4
5
6

Figura 2.3: Representación de una matriz de enteros 2×3 en la memoria

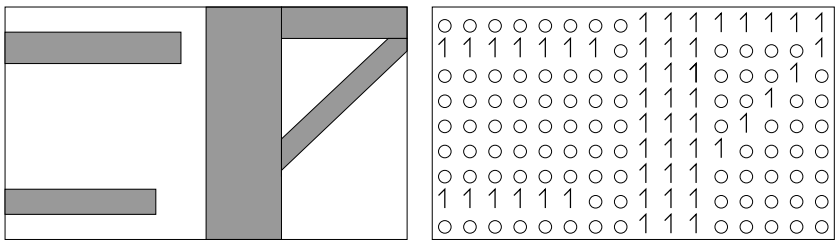


Figura 2.4: La imagen de la izquierda puede describirse usando una matriz de valores binarios como la de la derecha.

2.5.2 Matrices

Las matrices con conjuntos bidimensionales de elementos del mismo tipo. Se pueden considerar como vectores cuyos elementos son vectores. La forma más simple de representar las matrices en una computadora consiste en colocar los elementos por filas. Cada fila es considerada un vector y por tanto sus elementos son colocados en registros consecutivos de la memoria. La fila siguiente ocupa un espacio consecutivo a la fila anterior. A modo de ejemplo considere la matriz:

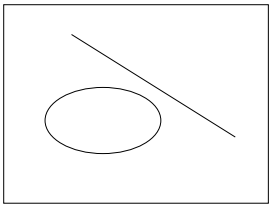
$$A = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 5 & 6 \end{pmatrix}$$

En la figura 2.3 puede verse la matriz A2×3 (dos filas y tres columnas) de enteros colocada en la memoria.

2.5.3 Formatos gráficos

Las imágenes pueden representarse mediante conjunto de puntos con un color o tonalidad de gris dado. Esta es la idea usada por formatos gráficos denominados MATRICES DE PUNTOS (llamados en inglés formatos *raster* y *bitmapped*). En la figura 2.4 se representa una silueta en blanco y negro descrita mediante una matriz de unos y ceros. El cero indica un punto luminoso y el uno un punto oscuro. Resulta imaginable que teniendo una matriz con muchos elementos se pueden conseguir imágenes de gran calidad.

Las imágenes resultantes son en blanco puro y negro puro, como siluetas. Para conseguir grises se puede definir la misma matriz conteniendo números del 0 al 255, de forma que a cada nivel de gris le corresponde un número, siendo 0 el negro y 255 el blanco. Cada elemento de la matriz ahora no es un bit, sino un carácter (8 bits). Para obtener el color se puede realizar la



```
7.500 slw
% Ellipse
n 900 1050 525 300 0 360 DrawEllipse gs col-1 s gr

% Polyline
n 615 270 m 2100 1200 l gs col-1 s gr
```

Figura 2.5: Descripción de una imagen (izquierda) en forma vectorial mediante PostScript.

misma operación 3 veces, una para cada color rojo, verde y azul.

Este formato precisa mucha memoria, por ejemplo, para almacenar una imagen en color de 200×200 puntos con ocho bits para cada color se necesitan $200 \times 200 \times 3 \times 8$ bits, que aproximadamente son 117 kilocaracteres (Kc o Kilobytes Kb).

Un método alternativo consiste en describir la imagen mediante ecuaciones correspondientes a líneas, superficies, etc, de forma que la unión de las partes produzca (aproximadamente) la imagen total. Se habla entonces de GRÁFICOS VECTORIALES. La codificación de la imagen se reduce a la de las ecuaciones de las partes. Estas ecuaciones pueden representarse de muchas formas. Por ejemplo un segmento queda perfectamente definido por las coordenadas de sus puntos extremos, una circunferencia por las de su centro y un punto de la misma, etc.

A modo de ejemplo considérese el dibujo de la figura 2.5 consistente en un segmento y una elipse. Utilizando el formato PostScript estos elementos quedan definidos en la forma en que se muestra a la derecha.

Los gráficos vectoriales son adecuados para dibujos formados por elementos geométricos simples. Ambos métodos (matrices de puntos y gráficos vectoriales) se combinan en los formatos denominados meta archivos (*metafiles*), usados por muchos programas para poder incluir objetos formados por líneas precisas, como planos y esquemas, junto a otros que quedan mejor descritos mediante una matriz de puntos como fotografías.

2.6 Ejercicios propuestos

Los siguientes ejercicios sirven para consolidar las ideas más importantes de este tema.

1. Escriba todos los números enteros que se pueden representar mediante el sistema signo-valor absoluto utilizando registros de cuatro celdas. Indique la cantidad representada por cada número.

2. Repita el ejercicio anterior usando el sistema de representación en complemento a dos para las cantidades negativas.

3. En la tabla siguiente los números de cada fila representan una misma cantidad y en cada columna un mismo sistema de representación. En todos los casos las representaciones binarias emplean 8 celdas. Complete la tabla indicando los valores adecuados.

Decimal	Signo-Valor Absoluto	Complemento a uno	Complemento a dos
-31			
	1 0 0 0 1 0 1 1		
		0 1 0 0 1 0 1 0	
			1 0 1 0 1 0 1 1

4. En la tabla de siguiente los números de cada fila representan una misma cantidad y en cada columna un mismo sistema de representación. En todos los casos las representaciones binarias emplean 10 celdas. Para el punto fijo se utiliza el sistema signo-valor absoluto. Para ello ha de tomarse 1 celda de signo más 5 celdas de parte entera y 4 celdas de parte fraccionaria. Para el punto flotante tenga en cuenta que la mantisa es un número de punto fijo con 1 celda de signo, 0 celdas de parte entera y 5 de fraccionaria. El exponente es un número entero expresado en complemento a dos usando en total 4 celdas. El ajuste fraccionario de la mantisa se lleva a cabo de tal modo que el coeficiente de la primera potencia fraccionaria sea 1. Complete las casillas vacías de la tabla indicando los valores adecuados.

Decimal	Complemento a dos	Punto fijo	Punto flotante
23			
	X X X X X X X X	1 0 0 0 0 0 0 1 0 0	
	X X X X X X X X		0 1 1 0 0 0 0 1 1 1
	1 1 1 0 1 1 0 0 1 0		
-8.75	X X X X X X X X		