2. Representación de la información

Informática

Ingeniería en Electrónica y Automática Industrial

Raúl Durán Díaz Juan Ignacio Pérez Sanz

Departamento de Automática
Escuela Politécnica Superior

Curso académico 2023–2024

Universidad de Alcalá

Contenidos

Universidad de Alcalá

Representación posicional de un número

- La representación posicional se basa en el siguiente teorema:

Teorema

Sea $b > 1$ un entero. Cualquier entero positivo n puede escribirse de modo único como

$$n = \sum_{j=0}^{k} a_j b^j = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0, \qquad (1)$$

con $0 \leq a_j \leq b - 1$ para $j = 0, \ldots, k$, y $a_k \neq 0$.

- Con ello, la representación posicional de n es

$$n = (a_k, a_{k-1}, \ldots, a_0),$$

o, más sencillamente, $a_k a_{k-1} \ldots a_0$.

Universidad de Alcalá

Bases de las representaciones

- Podemos utilizar cualquier base b, fijada, para representar los números. En la vida corriente, se usa la base 10, o *decimal*.

Observación

Dentro de un computador solo podemos almacenar bits; por ello, una de nuestras formas de representar los números (positivos) será la *binaria* en donde los coeficientes a_j de la fórmula (1) solo pueden tomar los valores 0 y 1.

Representación de números racionales

- Los números racionales se escriben canónicamente como cociente de dos números enteros coprimos (o sea, primos entre sí).
- La representación posicional admite también números racionales siempre que podamos conseguir que el denominador pueda escribirse como una potencia de la base b.
- Los irracionales y aquellos racionales que no admitan lo dicho en el punto anterior pueden representarse en notación posicional tomando el racional representable más próximo.

Representación de números racionales

- Supongamos el número $r = \left[\frac{p}{q}\right]$, sea b nuestra base y hemos conseguido que sea $q = b^s$. Entonces podremos escribir:

$$r = \frac{p}{q} = \frac{\sum_{j=0}^{k} p_j b^j}{b^s} = \sum_{j=0}^{k} p_j b^{j-s}.$$

- Si $k > s$, este número podemos escribirlo así:

$$r = (p_k p_{k-1} \cdots p_s, p_{s-1} \cdots p_0),$$

en donde los coeficientes $p_{s-1}, \ldots, p_0$ afectan a potencias negativas de la base b.

Universidad de Alcalá

Rev: 1.28

Cambio de base

- Supongamos dos bases b_1 y b_2. Sea un número real (truncado) u, v (u es la parte entera, v la parte fraccionaria).
- En la base b_1, $u = (p_{k-1}p_{k-2}\cdots p_0)_{b_1}$, $v = (,p_{-1}p_{-2}\cdots p_{-\ell})_{b_1}$, con $\ell > 0$.
- En la base b_2, $u = (q_{K-1}q_{K-2}\cdots q_0)_{b_2}$, $v = (,q_{-1}q_{-2}\cdots q_{-L})_{b_2}$, con $L > 0$.
- Queremos pasar de la representación en b_1 a la representación en b_2.

Universidad de Alcalá

Cambio de base

Procedimiento para obtener la parte entera

Dividir sucesivamente $(u)_{b_1}$ por $(b_2)_{b_1}$. Los restos que vayan saliendo, q_i, son los dígitos de $(u)_{b_2}$, empezando por q_0 hasta q_{K-1}.

Procedimiento para obtener la parte fraccionaria

Multiplicar sucesivamente $(v)_{b_1}$ por $(b_2)_{b_1}$. La parte entera que salga serán los dígitos de $(v)_{b_2}$, empezando por q_{-1} hasta q_{-L}. Quitar la parte entera obtenida antes de volver a multiplicar.

Observación

Como es lógico, el procedimiento para cambiar de base la parte entera es aplicable, en particular, a números enteros.

Ejemplo: cambiar 22,375 de base 10 a base 2

- Parte entera: $u = 22$

dividendo	cociente	resto
22	11	0
11	5	1
5	2	1
2	1	0
1	0	1

- Parte fraccionaria: $v =, 375$

multiplicando	producto	parte entera
0,375	0,75	0
0,75	1,5	1
0,5	1,0	1

- El resultado es 10110,011

Universidad de Alcalá

Cambio de base inverso

- Si nos es más familiar la aritmética en b_2, usamos estos procedimientos:

Procedimiento para obtener la parte entera

Aplicamos la fórmula
$((\cdots((p_{k-1}b_1 + p_{k-2})\,b_1 + p_{k-3})\,b_1 + \cdots)\,b_1 + p_1) + p_0$.

Procedimiento para obtener la parte fraccionaria

«Correr la coma hacia la derecha», multiplicando por b_1^ℓ y luego dividir por b_1^ℓ.

Universidad de Alcalá

Ejemplo: cambiar 10110,011 de base 2 a base 10

- Parte entera: $u = 10110$

$$(((2 \times 1 + 0) \times 2 + 1) \times 2 + 1) \times 2 + 0 = 22.$$

- Parte fraccionaria: $v =, 011$

$$\frac{(,011)_2 \times 2^3}{2^3} = \frac{(011)_2}{2^3} = \frac{(2 \times 0 + 1) \times 2 + 1}{8} = \frac{3}{8} = 0{,}375.$$

El resultado es 22,375.

Universidad de Alcalá

¿Qué es una *codificación*?

- Sean dos conjuntos A y B y sea una función $f : A \to B$.

Definición

Decimos que B *codifica* a A por f si se verifica que f es *biyectiva*.

- Supongamos además que cada conjunto está dotado de su operación interna, es decir, $(A, +)$, $(B, \oplus)$.

Definición

Si $f(a + b) = f(a) \oplus f(b)$ para $a, b \in A$ cualesquiera, entonces tenemos una *representación* (o *codificación*) *fidedigna*.

Universidad de Alcalá

Rev: 1.28

Un poco de matemáticas: operación módulo

Definición

Sea $m > 0$. Definimos la operación módulo con números enteros, $b = a \pmod{m}$, como el resto de la división entera de a entre m. Si $b = a \pmod{m}$, entonces $a = q \cdot m + b$, para cierto entero q, con $b < m$. Observemos que siempre se verifica $b \in \{0, 1, \ldots, m-1\}$.

Ejemplo

Si $m = 12$, y $a = 39$, se tiene que $3 = 37 \pmod{12}$. Solemos usar las horas del reloj módulo 12, por lo que la "hora 25" es, en realidad, la hora 1, pues $1 = 25 \pmod{12}$.

Universidad de Alcalá

Aritmética modular

Si a, b, c, $m > 0$ son enteros tales que $b = a \pmod{m}$, las siguientes fórmulas se verifican

1. $(a + c) \pmod{m} = (b + c) \pmod{m}$,
2. $(a - c) \pmod{m} = (b - c) \pmod{m}$,
3. $(ac) \pmod{m} = (bc) \pmod{m}$.

Representación de números enteros

- El número de bits que usamos en un computador para los números binarios es lo que llamamos «ancho de palabra», w.
- Ordinariamente, 8, 16, 32, o 64 bits.
- Cada uno de estos «anchos» recibe un nombre. Por ejemplo, en lenguaje C:

`char`	$\Rightarrow$	8 bits.
`short int`	$\Rightarrow$	16 bits.
`int`	$\Rightarrow$	32 bits.
`long int`	$\Rightarrow$	64 bits.

Universidad de Alcalá

Cuadro resumen de representaciones

<table>
<tr><td rowspan="4">Coma fija</td><td>Binario sin signo</td><td></td></tr>
<tr><td rowspan="3">Binario con signo</td><td>Con bit de signo</td></tr>
<tr><td>Exceso a Z</td></tr>
<tr><td>Complemento a 2</td></tr>
<tr><td>Complemento a 1</td><td></td><td></td></tr>
<tr><td>Coma flotante</td><td>Mantisa entera</td><td></td></tr>
<tr><td></td><td>Mantisa fraccionaria</td><td></td></tr>
</table>

Formato binario sin signo

- La función de correspondencia que vamos a usar es, simplemente, la del cambio a base 2, es decir:

$$\begin{aligned} f: R &\rightarrow B \\ n &\mapsto (x_{w-1}, \ldots, x_0)_2 \end{aligned}$$

  tal que $n = \sum_{i=0}^{w-1} x_i 2^i$.
- Para un ancho de w bits, el conjunto $R = \{0, 1, \ldots, 2^w - 1\}$ que se codifican como $0 \mapsto (0 \cdots 0)$, $2^w - 1 \mapsto (1 \cdots 1)$. Por tanto, solo números positivos y el 0.
- Realizando las operaciones en R módulo 2^w, esta representación es fidedigna.

Universidad de Alcalá

Formato binario con bit de signo

- Para manejar números enteros con signo, podemos definir:

$$n = (-1)^{x_{w-1}} \times (x_{w-2}, \ldots, x_0)_2 = (-1)^{x_{w-1}} \sum_{i=0}^{w-2} x_i 2^i.$$

  de modo que el conjunto $R = \{-2^{w-1} + 1, \ldots, 2^{w-1} - 1\}$ para un ancho de w bits.
- El signo viene dado por x_{w-1}: si es un 0, el número es positivo; en caso contrario, es negativo.
- Esta representación no es fidedigna pues el 0 tiene dos representaciones (no es biyectiva) y no se conserva la operación de suma.

Universidad de Alcalá

Formato binario con exceso a Z

- Sea $Z > 0$ un número entero. Para un ancho de w bits, podemos representar números del conjunto $R = \{-Z, \ldots, Z-1\}$ aplicando la transformación $n \mapsto n + Z$, $n \in R$. Como $n + Z \geq 0$, usamos después la representación de número binario sin signo, es decir,

$$n + Z = \sum_{i=0}^{w-1} x_i 2^i.$$

- Observemos que, con esta estrategia, a cada número le corresponde una y una sola representación. Típicamente, se elige $Z = 2^{w-1}$.

Universidad de Alcalá

Formato binario con exceso a Z

- La representación no es fidedigna, pues no respeta la operación de suma. En efecto, sean $n, m \in R$. Si los sumamos en $\mathbb{Z}$ y calculamos la representación de la suma, tenemos

$$(n + m) \mapsto (n + m) + Z.$$

Pero si sumamos en las representaciones en B, tenemos

$$\begin{array}{rcl} n & \mapsto & n + Z \\ & & + \\ m & \mapsto & m + Z \\ \hline & & (n + m) + 2Z, \end{array}$$

que no produce el resultado correcto (el de antes).

Universidad de Alcalá

Formato binario en complementos

- Las representaciones más usadas se llaman representaciones «con complemento».
- En realidad se trata de una aritmética modular, con un módulo adecuadamente elegido.
- Como, ordinariamente, trabajamos en base 2, tenemos dos «complementos»:
  - Cuando usamos como módulo 2^w, hablamos de «complemento a 2».
  - Cuando usamos como módulo $2^w - 1$, hablamos de «complemento a 1».

Universidad de Alcalá

Formato en complemento a 2

- Cada número de ese conjunto se puede representar biyectivamente en el conjunto B, de la siguiente forma:

$$
\begin{array}{rcccl}
-2^{w-1} & \mapsto & [2^{w-1}] & \mapsto & (1,0,\ldots,0)\,, \\
-2^{w-1}+1 & \mapsto & [2^{w-1}+1] & \mapsto & (1,0,\ldots,1)\,, \\
 & & \ldots & & \\
-1 & \mapsto & [2^{w}-1] & \mapsto & (1,1,\ldots,1)\,, \\
0 & \mapsto & [0] & \mapsto & (0,0,\ldots,0)\,, \\
1 & \mapsto & [1] & \mapsto & (0,0,\ldots,1)\,, \\
 & & \ldots & & \\
2^{w-1}-2 & \mapsto & [2^{w-1}-2] & \mapsto & (0,1,\ldots,0)\,, \\
2^{w-1}-1 & \mapsto & [2^{w-1}-1] & \mapsto & (0,1,\ldots,1)\,.
\end{array}
$$

Universidad de Alcalá

Rev: 1.28

Formato en complemento a 2

- En este caso, con un ancho w, podemos representar el conjunto $R = \{-2^{w-1}, \ldots, 2^{w-1} - 1\} \subset \mathbb{Z}$.
- Las operaciones se hacen «módulo 2^w».

Universidad de Alcalá

Formato en complemento a 2

- El complemento a 2 es una representación fidedigna, pues es biyectiva y la aritmética modular admite las operaciones de suma y producto (y sus inversas).
- Podemos manejar los números positivos y negativos de una manera homogénea, pero la representación es *asimétrica*: admite una cantidad distinta de positivos que de negativos.
- Esta representación se usa universalmente.

Formato en complemento a 1

- Esta representación es aplicable al mismo conjunto R que el complemento a 2. No es biyectiva, pues el cero va a tener dos representaciones en el conjunto B:

$$\begin{array}{rcccl}
-2^{w-1}+1 & \mapsto & [2^{w-1}] & \mapsto & (1,0,\ldots,0)\,, \\
 & & \ldots & & \\
-1 & \mapsto & [2^w-2] & \mapsto & (1,1,\ldots,0)\,, \\
0 & \mapsto & [2^w-1] & \mapsto & (1,1,\ldots,1)\,, \\
0 & \mapsto & [0] & \mapsto & (0,0,\ldots,0)\,, \\
1 & \mapsto & [1] & \mapsto & (0,0,\ldots,1)\,, \\
 & & \ldots & & \\
2^{w-1}-1 & \mapsto & [2^{w-1}-1] & \mapsto & (0,1,\ldots,1)\,.
\end{array}$$

- Las operaciones se hacen «módulo 2^w-1».

Universidad de Alcalá

Formato en complemento a 1

- La aritmética funciona perfectamente módulo $2^w - 1$.
- No es una representación fidedigna, pues el cero tiene dos representaciones. Ambas, sin embargo, son cero módulo $2^w - 1$. Esto es interesante para ciertas aplicaciones.
- El rango de positivos y negativos es simétrico.
- Es menos usada que la anterior.

Formato de coma fija

- Podemos trabajar con también con números que lleven ≪comas≫.
- Basta con fijar la coma en una posición y realizar las operaciones como si se tratara de enteros normales.
- La ventaja es que los operadores aritméticos ordinarios para enteros son aplicables directamente a la coma fija.
- Puede resultar interesante en ciertas aplicaciones, como la aritmética financiera.

Formato de coma fija

- Para un ancho de w bits, un esquema de coma fija puede provocar errores de truncado muy grandes, en función de la magnitud de los números representados.

Ejemplo

Obsérvese el diferente efecto relativo al suprimir el bit menos significativo en los siguientes números:

$$\begin{aligned} a &= (0000\ 0000,\ 0000\ 1001)_2 \\ b &= (1001\ 0000,\ 0000\ 0001)_2 \end{aligned}$$

Universidad de Alcalá

Formato de coma flotante

- La idea es utilizar una representación en que la coma no está fijada.
- Cada número x se representa como $x = \pm m \times b^e$, donde

x	número representado
m	mantisa
b	base
e	exponente

Ejemplo

$$\begin{aligned} a &= (1{,}001)_2 \times 2^{-5} \\ b &= (1{,}001)_2 \times 2^{7} \end{aligned}$$

Universidad de Alcalá

Rev: 1.28

Formato de coma flotante

- El típico formato para almacenar signo, mantisa, y exponente es:

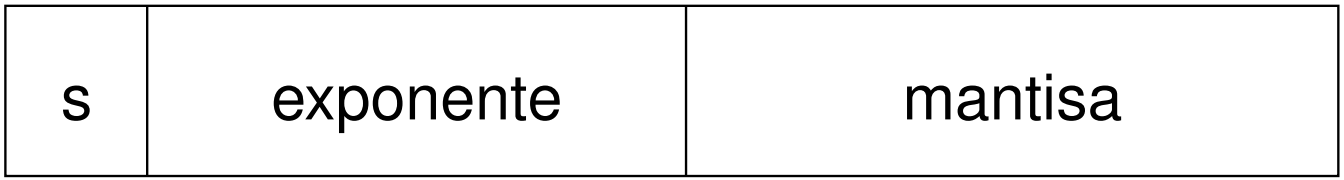

Figura: Formato de coma flotante

Campos del formato de coma flotante

signo	$0 \rightarrow$ positivo, $1 \rightarrow$ negativo.
exponente	entero en formato exceso a Z.
mantisa	entera o fraccionaria

Formatos de mantisa y exponente

- *Entera*: la mantisa es un número entero, en donde usamos alguno de los posibles formatos vistos antes (no se usa).
- *Fraccionaria*: la mantisa es la parte fraccionaria, normalizada de modo que la parte entera sea una sola cifra, distinta de cero.
- *Exponente*: Se almacena en exceso a $Z = 2^{w_e - 1}$, siendo w_e el número de bits dedicados a almacenarlo.

Ejemplos de coma flotante

Ejemplo

Sea $a = (1{,}001)_2 \times 2^{-5}$. La mantisa es $m = 1{,}001$, y el exponente $e = -5$. Esta mantisa ya está normalizada.

Sea $a = (10{,}01)_2 \times 2^{-6}$. La mantisa es $m = 10{,}01$, y el exponente $e = -6$. El valor de a es igual que antes, pero la mantisa no está normalizada, pues la parte entera tiene más de una cifra.

Sea $a = (0{,}1001)_2 \times 2^{-4}$. La mantisa es $m = 0{,}1001$, y el exponente $e = -4$. El valor de a es igual que antes, pero la mantisa no está normalizada, pues, aunque la parte entera tiene una sola cifra, ésta es cero.

Por cierto: $a = \frac{(1001)_2}{2^3} \times \frac{1}{2^5} = \frac{9}{256} = 0{,}03515625$.

Universidad de Alcalá

Representación ANSI/IEEE 754

- Este estándar define la cantidad de bits que se dedican a cada campo.
- Como el bit más significativo de la mantisa es siempre 1, por construcción, nunca se almacena (técnica del *bit oculto*).
- Existen dos tamaños para este tipo de datos:
  - tipo de dato flotante simple, `float`, de 32 bits en total.
  - tipo de dato flotante doble, `double`, de 64 bits en total.

Estándar ANSI/IEEE 754: valores especiales

- El cero no se puede representar: por convenio se elige el número con todos los bits puestos a cero.
- Si el exponente tiene todos los bits a 0 pero la mantisa no, se entiende que está *desnormalizada*. En tal caso, si la mantisa vale m, el número representado es

$$\begin{array}{ll} 0, m \times 2^{-126} & \text{para flotante simple,} \\ 0, m \times 2^{-1022} & \text{para flotante doble.} \end{array}$$

Universidad de Alcalá

Estándar ANSI/IEEE 754: valores especiales

- Se utiliza también dos códigos para representar el valor $\pm\infty$.
- Cuando alguna operación tiene un resultado indefinido (por ejemplo, dividir 0/0), se elige un código especial, que llama ≪NaN≫ (Not a Number).

Universidad de Alcalá

Estándar ANSI/IEEE 754

	simple	**doble**
ancho	32 bits	64 bits
mantisa	$23+1$ bits	$52+1$ bits
exponente	8 bits	11 bits
exceso	$2^7-1=127$	$2^{10}-1=1023$
mínimo	$2^{-126} \simeq 1{,}175 \times 10^{-38}$	$2^{-1022} \simeq 2{,}225 \times 10^{-308}$
máximo	$2^{128}-2^{104} \simeq 3{,}403 \times 10^{38}$	$2^{1024}-2^{971} \simeq 1{,}798 \times 10^{308}$
cero	$exp=0\ ,\ m=0$	$exp=0,\ m=0$
desnormalizado	$exp=0\ ,\ m\neq 0$	$exp=0,\ m\neq 0$
$\pm\infty$	$exp=255,\ m=0$	$exp=2047,\ m=0$
NaN	$exp=255,\ m\neq 0$	$exp=2047,\ m\neq 0$

Universidad de Alcalá

Representación de información alfanumérica

- La información alfanumérica se codifica mediante tablas de caracteres.
- Las tablas de caracteres implementan la función biyectiva entre cada carácter representado y el código que lo representa.
- Cada tabla define cuántos bits codifican cada carácter alfanumérico.
- Existen varias tablas estandarizadas:
  - Estándar ANSI/ASCII.
  - Estándares ISO8859-XX.
  - Estándares Unicode, UTF-8, UTF-16.
  - Estándar IBM/EBCDIC.

Tabla ANSI/ASCII-7

- Se utilizan códigos de 7 bits para codificar hasta 128 caracteres alfanuméricos y de control de impresión.

Ejemplos:

Carácter	"0"	"1"	...	"9"	"A"	...	"Z"
Código ASCII-7	48	49	...	57	65	...	90

Universidad de Alcalá

Tabla ISO8859-15

- Utiliza códigos de 8 bits para codificar hasta 256 caracteres alfanuméricos y de control. Los 128 primeros códigos son totalmente compatibles con la tabla ASCII-7.
- Los otros 128 códigos están pensados para representar caracteres de las lenguas occidentales.

Ejemplos:

Carácter	"é"	...	"è"	...	"û"	...
Código ISO8859-15	130	...	138	...	150	...

Universidad de Alcalá

Tabla UTF-8

- Esta tabla utiliza códigos de longitud variable, 8 o 16 bits.
- Para códigos menores de 128, el bit más significativo es 0 y UTF-8 es totalmente compatible con ASCII-7.
- El resto de caracteres se codifican con dos bytes, el primero de los cuales tiene su bit más significativo a 1. Ello es un indicador de que el código es de 16 bits.
- Observemos que el espacio de codificación es mucho mayor y permite codificar caracteres de muchos idiomas, incluyendo, por ejemplo, los orientales.

Ejemplos de la tabla UTF-8

Carácter	“é”	. . .	“è”	. . .	“û”	. . .
Cód. UTF-8	0xC3A9	. . .	0xC3A8	. . .	0xC3BB	. . .

Universidad de Alcalá

Cadenas de caracteres

- Para poder representar cadenas de caracteres en memoria hay que ponerse de acuerdo en dos cosas:
  - tipo de codificación;
  - cómo codificar la longitud de la cadena.

Cadenas de caracteres

- Método del «terminador».
- Método del «indicador de longitud».
- Método del «descriptor».

Método del «terminador»

- Se usa un código preconvenido para marcar el fin de la cadena. Típicamente se usa el 0.
- Para ubicar la cadena, basta saber la dirección en que reside el primer carácter.

Ejemplo

Para representar la cadena `"Hola"` con tabla ISO8859-15, usamos cinco bytes:

`'H'`	`'o'`	`'l'`	`'a'`	0

Universidad de Alcalá

Método del «indicador de longitud»

- Se conviene que el primer byte (o los dos primeros bytes) de la cadena indica(n) la longitud de ella.
- Para ubicar la cadena, basta saber la dirección en que reside el primer carácter.
- Este método limita la longitud máxima posible de una cadena.

Ejemplo

Para representar la cadena `"Hola"` con tabla ISO8859-15, usamos cinco bytes:

4	'H'	'o'	'l'	'a'

Universidad de Alcalá

Método del «descriptor»

- Los caracteres de la cadena se escriben sin más a partir de una posición de memoria.
- Para ubicar la cadena, hacen falta la dirección en que reside el primer carácter y su longitud.

Ejemplo

Para representar la cadena `"Hola"` con tabla ISO8859-15, usamos cuatro bytes:

`'H'`	`'o'`	`'l'`	`'a'`

Pero necesitamos saber la posición de memoria en donde reside la cadena y, *además*, su longitud. La combinación de estas dos informaciones se suele llamar *descriptor*.