

Tema 1

La información digital

Curso 2008-2009

Objetivos

- Conocer cómo representan los computadores diferentes tipos de información mediante el sistema binario
- Conocer las limitaciones de cada método de representación

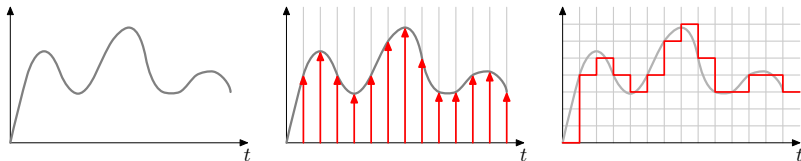
Esquema

- 1 Introducción
 - Señal analógica, digital y binaria
 - Bases de numeración
 - Códigos
 - Prefijos binarios
- 2 Representación y aritmética de enteros
 - Números naturales
 - Enteros en complemento a 2
 - Enteros en signo-magnitud
 - Enteros en complemento a Z
- 3 Representación de reales
 - Coma fija: formato y limitaciones
 - Coma flotante: concepto
 - Coma flotante: norma IEEE-754
 - Coma flotante: rango y precisión
- 4 Representación de caracteres
 - Códigos alfanuméricos
 - ASCII
 - Estándar ISO-8859

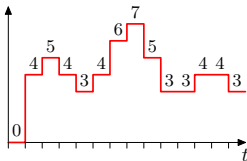
Tipos de señal

- Una señal analógica varía de forma continua en el tiempo
 - Una señal digital es *discreta*
 - Una señal binaria es una señal digital con sólo dos posibles valores
- En el mundo real las señales son analógicas.
 - La electrónica digital trata con señales digitales.
 - Los computadores tratan con señales binarias.

- La señal analógica se transforma en digital mediante *muestreo* y *discretización*.



- La señal digital resultante es una secuencia de enteros.

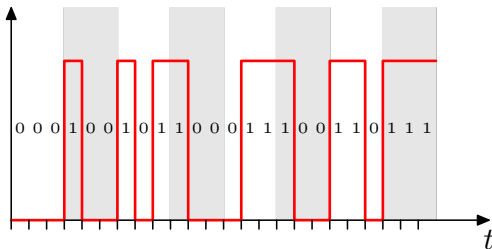


- La señal digital es más inmune al ruido.

- Cada entero puede representarse mediante una secuencia de bits.

000 100 101 100 011 100 110 111...

- La secuencia de bits es una señal binaria.



- La señal binaria es *aún más* inmune al ruido.

Cualquier número natural se puede representar mediante una señal binaria: representándolo en base 2.

Sistema posicional

- En un número de varias cifras, cada posición tiene diferente “peso”:

$$\begin{aligned}1265 &= 1000 + 200 + 60 + 5 \\ &= 1 \times 10^3 + 2 \times 10^2 + 6 \times 10^1 + 5 \times 10^0\end{aligned}$$

- El peso es 10 elevado a la posición que ocupe la cifra
- La idea funciona para números con decimales, si se considera que éstos tienen posiciones negativas.

$$\begin{aligned}3,75 &= 3 + 0,7 + 0,05 \\ &= 3 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}\end{aligned}$$

Lo anterior se resume en la fórmula:

$$\text{Valor} = \sum_{i=-\infty}^{\infty} d_i \times 10^i \quad (1)$$

siendo d_i el dígito que está en la posición i .

La fórmula no es del todo genérica

¿Por qué aparece un 10?

¿Cómo puede generalizarse?

Generalización

$$\text{Valor} = \sum_{i=-\infty}^{\infty} d_i \times B^i \quad (2)$$

siendo B la base de numeración.

Escribir una cantidad N en una base B

Algoritmo para encontrar la secuencia de dígitos d_i

- ① Comenzar con $i = 0$, y con $D = N$.
- ② Se divide D entre B , y se obtiene:
 - Un cociente C_i
 - Un resto, que será la cifra d_i del resultado.
- ③ Si el cociente C_i es cero, hemos terminado.
- ④ Si no, hacer $D = C_i$, $i = i + 1$ y volver al paso 2.

Ejemplo. Escribir la cantidad 37...

- En base 8.
- En base 5.
- En base 2.
- En base 16.

Sobre las cifras

- Al representar un número en base B , cada dígito es una cifra entre 0 y $(B - 1)$.
- En base 10, tenemos cifras del 0 al 9.
- En base 2, se necesitan sólo cifras del 0 al 1.
- En base 16 (hexadecimal) son necesarias cifras del 0 al 15 (?)
Es necesario inventar “nuevas cifras” para 10, 11, 12..
Se usan las 5 primeras letras del abecedario.

Ejemplo

Escribir la cantidad 29 en hexadecimal.

Escribir la cantidad 255 en hexadecimal.

Bases de uso común

- Base 10.
Denotada por el sufijo “d”, o sin sufijo.
- Base 2 (binario).
Denotada por el sufijo “b”.
- Base 16 (hexadecimal).
Denotada por el sufijo “h”.

Conversiones entre bases

- De cualquier base a base 10: Usar ecuación (2).
- De base 10 a cualquier base: Usar algoritmo.
- Entre bases 2 y 16, sin pasar por base 10: Trucos especiales.

Conversión directa entre bases 2 y 16

Hex	Bin
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hex	Bin
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Base 16 a base 2

Convertir cada dígito hexadecimal en cuatro dígitos binarios

Base 2 a base 16

Convertir cada cuatro dígitos binarios en un dígito hexadecimal.

Para representar cualquier información en un computador, debe usarse un *código binario*.

- Se decide cuántos bits se usarán.
- Con N bits se tienen 2^N combinaciones.
- Cada combinación de bits es un *código*.
- Se asigna un código a cada posible valor de la información representada.

Ejemplos

¿Bits para representar los palos de la baraja? .

¿Bits para representar los colores del arcoiris?

¿Cuántos colores puedo representar con 16 bits?

Cualquier información que pueda codificarse con un número entero, puede representarse con bits en un computador.

El número de bits es un límite

Con N bits se pueden representar 2^N valores diferentes

Unidades de información famosas:

N	Nombre	2^N
4	<i>nibble</i>	16
8	<i>byte</i>	256
16	palabra de 16	65 535
32	palabra de 32	4 294 967 296

Kilobites, Megabytes...

- El prefijo *kilo-* (k) es definido por el Sistema Internacional de Unidades como igual a 1000.
- Por tanto, *Kilobyte* (kB) serían 1000 bytes.
- Sin embargo, en muchos contextos KB significa 1024 bytes.
- Análogamente, Mega (M), Giga (G), Tera (T),...
(Deberían significar 10^6 , 10^9 , 10^{12})
(Para bytes pueden significar 2^{20} , 2^{30} , 2^{40}).

Confusión

Estándar Internacional

Para evitar confusión:

- A los prefijos que denoten potencias de 2, se les llamará “Kibi”, “Mebi”, “Gibi”, “Tebi”, etc.
- El símbolo llevará añadida una “i” (Ki, Mi, Gi, Ti,...)

Ejemplo	Se lee	Significa
2 MB	Dos Mega Bytes	2 000 000 bytes (2×10^6)
5 KiB	Cinco Kibi Bytes	6 020 bytes (5×2^{10})
1 MiB	Un Mebi Byte	1 048 576 bytes (2^{20})

Trampa comercial

¿Cuántos GiB son 120 GB?

El código más simple

Cada natural es codificado mediante su representación binaria.

Problema de rango

Con N bits sólo podremos representar naturales en $[0, 2^N - 1]$

Ej: en 1 byte, el rango es $[0, 255]$.

.

Problema de desbordamiento

Una operación entre dos números válidos, puede producir un número no válido (fuera de rango).

Detección del desbordamiento

Sumando en binario: $131 + 141$

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

El resultado no cabe en 8 bits

Se detecta porque el acarreo de la última columna (más a la izquierda) es 1.

Representación de enteros

Problema

Los enteros pueden ser positivos o negativos. El signo es un elemento más a almacenar, pero no es 0 ó 1.

Soluciones

Hay varias:

- Forzar una codificación del signo en un bit extra.
Por ejemplo: 0=positivo, 1=negativo
Método denominado *signo-magnitud*.
- Convertir los negativos en positivos, sumándoles una constante
Método denominado *exceso a Z*
- Complemento a 2.

Complemento a 2

El objetivo

Lograr una representación de positivos y negativos que puedan ser sumados con aritmética de naturales, dando el resultado correcto.

El rango

Dados N bits, se tienen 2^N enteros representables.

- La mitad serán negativos (2^{N-1})
- Uno de ellos será el cero
- La otra mitad menos uno, serán positivos ($2^{N-1} - 1$)

El rango es asimétrico: $[-2^{N-1}, +2^{N-1} - 1]$

El método (con N bits)

- Un positivo se representa directamente en binario natural.
- Un negativo K se representa como la cantidad $2^N - |K|$

Atajo

Método rápido para los negativos:

- Representar $|K|$, usando N bits
- Cambiar $0 \leftrightarrow 1$
- Sumar 1 al resultado.

O también:

- Representar $|K|$, usando N bits
- Comenzando por la derecha, copiar hasta el primer 1, inclusive
- A la izquierda de éste, cambiar $0 \leftrightarrow 1$

Ejemplos

Usando 4 bits

Representar los siguientes enteros en C-2:

- +5
- -3
- -9

Usando 8 bits

Representar los siguientes enteros en C-2:

- 0
- -1
- 127
- -128

Aritmética en complemento a 2

Si se suman dos números en complemento a 2, ignorando el acarreo final, el resultado es correcto en complemento a 2.

Ejemplos (usando 4 bits)

- Sumar $2+3$
- Sumar $(-2)+(-3)$
- Sumar $2+(-3)$
- Sumar $(-2)+3$
- Sumar $4+5$

Detección del desbordamiento (*overflow*)

El acarreo final ya no sirve

Observaciones

- Al sumar dos datos de diferente signo, el resultado siempre estará en el rango.
- El desbordamiento sólo se producirá al sumar dos datos del mismo signo.

Algoritmo de detección de desbordamiento

Comparar bits de signo de datos y resultado:

- Si los datos tienen igual signo, y el resultado diferente
⇒ hubo desbordamiento.
- En otro caso no.

(Continuará...)