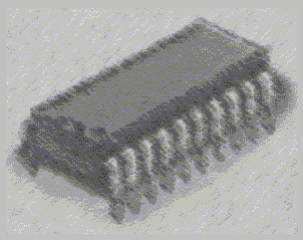
Tema 2. Sistemas de representación de la información



Estructura de Computadores

I. T. Informática de Gestión / Sistemas

Curso 2008-2009

Índice

- Definiciones
- Bases de numeración
- Modos de representación
- Representaciones numéricas
  - Coma fija (números enteros)
  - Coma flotante (números fraccionarios)
- Representaciones alfanuméricas
- Representaciones redundantes

Definiciones

Espacio material: número de bits que se tienen para almacenar el dato (número o carácter)
- Byte (8 bits)
- Palabra (n bits)

Rango de representación: valores máximo y mínimo que se pueden representar en un determinado sistema

Resolución de la representación: diferencia entre un número y el siguiente inmediato

Longitud del código: cuántos elementos diferentes se pueden obtener para una representación con *n* bits de espacio material. La longitud del código para *n* bits es 2^n

Bases de numeración (I)

- Bases más usadas en el computador 2, 8 y 16.

Binario (base 2)	Octal (base 8)	Decimal (base 10)	Hexadecimal (base 16)
0	0 (000)	0 (0000)	0 (0000) A (1010)
1	1 (001)	1 (0001)	1 (0001) B (1011)
	2 (010)	2 (0010)	2 (0010) C (1100)
	3 (011)	3 (0011)	3 (0011) D (1101)
	4 (100)	4 (0100)	4 (0100) E (1110)
	5 (101)	5 (0101)	5 (0101) F (1111)
	6 (110)	6 (0110)	6 (0110)
	7 (111)	7 (0111)	7 (0111)
		8 (1000)	8 (1000)
		9 (1001)	9 (1001)

- Nuestra base es base 10. Cambiar entre bases usa la regla de Horner

Bases de numeración (II)

P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

A cada posición le corresponde un peso

$$Valor = \sum_{i=0}^{n-1} x_i \cdot base^i$$

Unidades
Decenas
Centenas
Unidades de millar
Decenas de millar

Ejemplos:

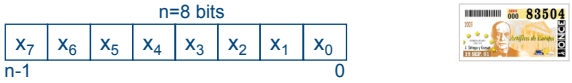
- Consideremos el número binario 10101. Pasado a su valor decimal:
- $1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 21$
- El número 78A en base hexadecimal pasado a decimal:
- $7.16^2 + 8.16^1 + 10.16^0 = 1.930$

Representaciones numéricas (I)
Coma fija (I)

Coma fija:

- **Sin signo:**
  - Binario puro
- **Con signo:**
  - Signo-magnitud
  - Complemento a la base, complemento a 2
  - Complemento restringido a la base, complemento a 1
  - Exceso a M
  - BCD

Representaciones numéricas (II)
Coma fija (II). Binario puro

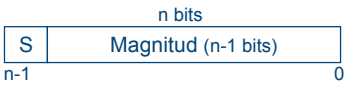


- Sistema posicional de base 2 para números enteros
- Donde los pesos son: $P_i = 2^i$
- Con palabra de longitud *n* se calcula el valor del número como:

$$Valor = \sum_{i=0}^{n-1} 2^i \cdot x_i$$

- Rango: $[0, 2^n - 1]$
- Resolución = 1

Representaciones numéricas (III)
Coma fija (III). Signo-magnitud



- Un bit indica el signo: 0 signo positivo y 1 signo negativo
- Con palabra de longitud n se calcula el valor del número como:

$$Valor = \begin{cases} +\sum_{i=0}^{n-2} 2^i \cdot x_i & \text{si } x_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i \cdot x_i & \text{si } x_{n-1} = 1 \end{cases}$$

- Rango: $[-(2^{n-1} - 1), -0, 0, (2^{n-1} - 1)]$ (rango simétrico)
- Resolución = 1

Representaciones numéricas (IV)
Coma fija (IV). Complemento a 1

- Números positivos comienzan por 0, representados en binario puro
- Números negativos comienzan por 1, representados en Ca1
- El MSB indica el signo, pero se opera con los n bits como un conjunto indivisible
- -A = Ca1(A), *n*=número de bits de la representación
  - $2^n - A - 1$
  - $\bar{A}$

$$Valor = \begin{cases} +\sum_{i=0}^{n-1} 2^i \cdot x_i & \text{si } x_{n-1} = 0 \\ -Valor(Ca1(número)) & \text{si } x_{n-1} = 1 \end{cases}$$

- Rango: $[-(2^{n-1}-1), -0, 0, (2^{n-1} - 1)]$ (rango simétrico)
- Resolución = 1

Representaciones numéricas (V)
Coma fija (V). Complemento a 2

- Números positivos comienzan por 0, representados en binario puro
- Números negativos comienzan por 1, representados en Ca2
- El MSB indica el signo, pero se opera con los n bits como un conjunto indivisible
- -A = Ca2(A), *n*=número de bits de la representación
  - $2^n - A$
  - $\bar{A}+1$

$$Valor = \begin{cases} +\sum_{i=0}^{n-1} 2^i \cdot x_i & \text{si } x_{n-1} = 0 \\ -Valor(Ca2(número)) & \text{si } x_{n-1} = 1 \end{cases}$$

- Rango: $[-2^{n-1},-1, 0, (2^{n-1} - 1)]$ (rango asimétrico)
- Resolución = 1

Representaciones numéricas (VI)
Coma fija (VI). Exceso M

- El número A se representa como A + M en binario puro
- M suele valer 2^{n-1} siendo *n* el número de bits utilizados en la representación.

- Valor: Sea n = 8. M = $2^{n-1} = 2^7 = 128$
  - -16 se representa como -16+128 = 112 0111 0000
  - 0 se representa como 0+128 = 128 1000 0000
  - -128 se representa como -128+128 = 0 0000 0000
  - 32 se representa como 32+128 =160 1010 0000

- Siempre que $M=2^{n-1}$ se verifica que es equivalente escribir el número en Ca2 con *n* bits y negar el MSB
- Rango: $[-2^{n-1}, -1, 0, (2^{n-1} - 1)]$ (rango asimétrico. Idem. a Ca2)
- Resolución = 1

Representaciones numéricas (VII)
Coma fija (y VII). BCD

- Se convierten, uno a uno, los dígitos decimales a binario
- Dos clases:
  - BCD empaquetado
  - BCD desempaquetado (alfanumérico)
- BCD desempaquetado

byte

0000	Dígito BCD

BCD empaquetado

byte

Dígito BCD	Dígito BCD

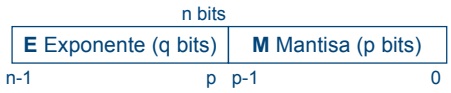
Valor	BCD	Valor	BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Representaciones numéricas (VIII)
Coma flotante (I)

Coma flotante:

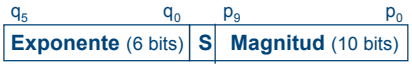
- Con mantisa entera **Mantisa** ,
- Con mantisa fraccionaria: , **Mantisa**
  - No normalizada
  - Normalizada
    - Sin bit implícito
    - Con bit implícito

Representaciones numéricas (IX)
Coma flotante (II)



- Divide los *n* bits de la representación en dos partes: *p* bits para la mantisa y *q* bits para el exponente
- El valor del número = valor(M) x basevalor$^{(E)}$
- Las bases más utilizadas son 2 y 16.
- M y E se pueden representar en alguno de los sistemas de coma fija
  - E suele tener base 2 y se suele representar en exceso 2^{q-1}
  - M puede ser:
    - Entera (regla de Horner para números enteros)
    - Fraccionaria (regla de Horner para números fraccionarios)

Representaciones numéricas (X)
Coma flotante (III). Mantisa entera



- Algunos ejemplos con mantisa entera representada como signo-magnitud sobre 11 bits y con exponente en exceso a 32 sobre 6 bits:

Exponente	Signo	Mantisa	Valor
100000	0	00000 01101	$13 \cdot 2^0 = 13$
100000	1	00000 01100	$-12 \cdot 2^0 = -12$
100010	0	00000 00101	$5 \cdot 2^2 = 20$
011100	1	00000 10100	$-20 \cdot 2^{-4} = -1.25$

- Ya no se usa esta clase de representación

Representaciones numéricas (XI)
Coma flotante (IV). Mantisa fraccionaria (I)

- La representación más corriente para la mantisa fraccionaria es la siguiente

Mantisa fraccionaria no normalizada:

- Se representa la mantisa tal y como queda

Mantisa fraccionaria normalizada:

- Consiste en eliminar todos los dígitos no significativos a la derecha de la coma
- De esta forma se aprovechan al máximo los bits disponibles y cada número tiene una única representación

Representaciones numéricas (XII)
Coma flotante (V). Mantisa fraccionaria (II)

- **Condiciones de normalización:**

$$Signo-magnitud \begin{cases} Números\ Positivos: & ,1xxxxx...x \\ Números\ Negativos: & ,1xxxxx...x \end{cases}$$

$$Complemento\ a\ 1 \begin{cases} Números\ Positivos: & ,01xxxxx...x \\ Números\ Negativos: & ,10xxxxx...x \end{cases}$$

$$Complemento\ a\ 2 \begin{cases} Números\ Positivos: & ,01xxxxx...x \\ Números\ Negativos: & ,10xxxxx...x \end{cases}$$

Representaciones numéricas (XIII)
Coma flotante (VI). Mantisa fraccionaria (III)

- **Técnica del bit implícito:** consiste en "arañar" un bit más para aumentar la precisión del valor representado. Consiste en no representar el primer bit de la mantisa puesto que conocemos su valor y se puede reconstruir.
- Se debe añadir para calcular el valor del número o el rango de representación
- **Condiciones de normalización con bit implícito:**

$$Signo-magnitud \begin{cases} Números\ Positivos: & ,xxxxx...x \\ Números\ Negativos: & ,xxxxx...x \end{cases}$$

$$Complemento\ a\ 1 \begin{cases} Números\ Positivos: & ,1xxxxx...x \\ Números\ Negativos: & ,0xxxxx...x \end{cases}$$

$$Complemento\ a\ 2 \begin{cases} Números\ Positivos: & ,1xxxxx...x \\ Números\ Negativos: & ,0xxxxx...x \end{cases}$$

Representaciones numéricas (XIII)
Coma flotante (VI). Estándar IEEE 754 (I)

- **Exponente:** representado en exceso $2^{q-1} - 1$
- **Mantisa:** representada en signo-magnitud, fraccionaria, normalizada y con la coma situada a la derecha del bit implícito.
- **Simple precisión:**
  - Exponente de 8 bits en exceso $2^{8-1} - 1$
  - Mantisa de 24 bits (1 bit de signo y 23 de magnitud)

Signo	Exponente	Mantisa
1 bit	*8 bits*	*23 bits*

Signo	Exponente	Mantisa
1 bit	*11 bits*	*52 bits*

- **Doble precisión**:
  - Exponente de 11 bits en exceso $2^{11-1} - 1 = 1.023$
  - Mantisa de 53 bits (1 bit de signo y 52 de magnitud)

Representaciones numéricas (XIV)
Coma flotante (VII). Estándar IEEE 754 (II)

Combinaciones especiales de la mantisa y del exponente

- Exponente 0 y mantisa 0. Sirve para representar ± 0.
- Exponente 0 y mantisa diferente de cero. Representa números no normalizados y permite representar números con un exponente muy pequeño pudiendo tratar situaciones de desbordamiento del exponente.
- Exponente 1 y mantisa 0. Sirve para representar $\pm \infty$.
- Exponente 1 y mantisa diferente de 0. Se emplea para señalar condiciones de excepción.

Representaciones numéricas (XV)
Coma flotante (VIII). Estándar IEEE 754 (III)

Ejemplos de números en el estándar IEEE 754 representados en simple precisión

Valor = $28_{(10}$

0	1000 0011	1100 … 00
1 bit	*8 bits*	*23 bits*

Valor = $-9_{(10}$

1	1000 0010	0010 … 00
1 bit	*8 bits*	*23 bits*

Modos de representación alfanumérica (I)

Representaciones alfanuméricas:

- Codifican mediante un grupo de bits (6, 7, 8, 16) cada uno de los caracteres a representar.

- Ejemplos de códigos alfanuméricos:
  - 6 bits (64 caracteres posibles) Fieldata y BCDIC
  - 7 bits (128 caracteres posibles) ASCII
  - 8 bits (256 caracteres posibles) ASCII extendido y EBCDIC
  - 16 bits (65536 caracteres posibles) UNICODE

Modos de representación alfanumérica (II)
Representación de cadenas de caracteres

- Las frases se forman agrupando caracteres. Existen varias alternativas:
- **Cadenas de longitud fija:**
  Se define una longitud máxima para todas las cadenas.
- **Cadenas de longitud variable:**
  - Con carácter separador
  - Con longitud explícita

Modos de representación alfanumérica (III)
Tabla de código ASCII

Representaciones redundantes (I)

- El objetivo de las representaciones redundantes es salvaguardar la información frente a los posibles errores en su almacenamiento o manipulación
- Para ello se añade al dato, información adicional que permite comprobar y, en algunos casos, corregir los errores
- Existen diferentes tipos de códigos redundantes:
  - Detectores
  - Correctores
- Entre los más usados se encuentran:
  - Códigos de paridad (detectores / correctores)
  - Códigos de Hamming (correctores)
- Circuitos que emplean códigos de paridad:
  - ECC - Error Correcting Codes / SEC - Single Error Correcting

Representaciones redundantes (II) Códigos de paridad

- Detecta los posibles errores, añadiendo a cada dato un bit adicional:
- Con paridad par, se añade 0 si el número de unos en el dato es par y 1 si el número de unos es impar.
- Ejemplo: Sólo se detecta 1 error

Número binario	Número de unos	Código de paridad
10010111	impar	1
11001100	par	0
01010101	par	0
00110011	par	0
11011010	impar	1

- **Mejora**:
- Añadir, además, una palabra de paridad para todo un conjunto de palabras (control de paridad horizontal y vertical)
- Detecta dos errores, siendo posible la corrección de uno de ellos

Representaciones redundantes (III) Códigos de Hamming (I)

- Realiza detección y corrección de errores
- Basado en bits de paridad que se colocan en lugares estratégicos
- Debe cumplirse que:

  $2^p \geq n + p + 1$, donde:

- n es el número de bits de datos del código
- p es el número de bits de paridad que se añaden

Representaciones redundantes (IV) Códigos de Hamming (II)

- Ejemplo: deseamos proteger el número 0 1 1 1
- Debe cumplirse que: $2^p \geq p + n + 1$, donde
- n= 4 bits de datos. $2^p \geq p + 4 + 1$ donde $2^p \geq p + 5$.
- El primer valor que cumple la inecuación es: $2^3 \geq 3 + 5 \rightarrow 8 \geq 8$, de donde el número de bits de paridad necesarios es p = 3
- Por tanto, se codificarán 3 + 4 = 7 bits
- La descomposición en potencias de dos indica que bits protege cada uno de los bits de paridad. En nuestro caso será:

  b7 = b4 + b2 + b1
  b6 = b4 + b2
  b5 = b4 + b1
  b4 = b4 , bit de protección
  b3 = b2 + b1
  b2 = b2 , bit de protección
  b1 = b1 , bit de protección

Representaciones redundantes (V)
Códigos de Hamming (III)

- Cada uno de los bits de protección protegerá al bit que lo contenga en su descomposición, así el bit b4 protegerá a los bits: b7, b6 y b5
- El valor del dato es:

b7	b6	b5	b4	b3	b2	b1
0	1	1		1		

- Considerando paridad par, el valor de los bits de protección será:

b7	b6	b5	b4	b3	b2	b1
0	1	1	***0***	0	***0***	***0***

- b1 protege a los bits b3, b5 y b7, siendo el número de unos par → b1 = 0
- b2 protege a los bits b3, b6 y b7, siendo el número de unos par → b2 = 0
- b4 protege a los bits b5, b6 y b7, siendo el número de unos par → b4 = 0

Bibliografía

- Fundamentos de los Computadores. (Capítulo 2)
  Pedro de Miguel Anasagasti
  Ed. Paraninfo
- Arquitectura de Computadores (Anexo A)
  J. Antonio de Frutos, Rafael Rico
  Ed. Universidad de Alcalá
- Arquitectura, programación y diseño de sistemas basados en microprocesadores (8086/80186/80286). (Capítulo 1)
  Yu-Cheng Lu, Glen A. Gibson
  Ed. Anaya Multimedia 86