# Sistemas numéricos

MIA José Rafael Rojano Cáceres

Arquitectura de Computadoras I

## Definición(1)

- Un sistema de representación numérica es un sistema de lenguaje que consiste en:
  - un conjunto ordenado de símbolos (dígitos o cifras).
  - un conjunto de reglas bien definidas para las operaciones aritméticas de suma, resta, multiplicación, división, etc.
- Los números en un sistema de numeración consisten en una secuencia (vector) de dígitos que pueden tener parte entera y parte fraccionaria, ambas separadas por una coma (o punto).

$$(N)_r = [(\text{parte entera}) , (\text{parte fraccionaria})]_r$$

# Definición(2)

- La **base** (r) de un sistema de numeración especifica el número de dígitos o cardinal* de dicho conjunto ordenado. Las bases más utilizadas son:
  - base 2: binaria = {0,1}
  - base 10: decimal = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - base 16: hexadecimal = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
  - base 8: octal = {0, 1, 2, 3, 4, 5, 6, 7}

*El cardinal o cardinalidad indica el número o cantidad de elementos de un conjunto

- El sistema más usual de representación numérica es el **sistema posicional** donde cada dígito del vector numérico tiene un valor distinto dependiendo de su posición concreta en el vector.
- Así, un número N en base r se representa de la siguiente manera:
  - $(N)_r = (a_{p-1}\, a_{p-2} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-q})_r$ ó
  - $N = a_{p-1}\, a_{p-2} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-q}$ *si se sobreentiende que está en base*

Donde

- $\mathbf{a_i}$ son los dígitos,
- **p** es el número de dígitos enteros,
- **q** es el número de dígitos fraccionarios,
- $\mathbf{a_{p-1}}$ es el dígito más significativo,
- $\mathbf{a_{-q}}$ es el dígito menos significativo.
- **r** es la base en la que se representa al vector

- Al ser N un número en base r, sus dígitos deben situarse entre 0 y r-1, es decir:

$$0 \le a_i \le r-1, \ \forall i \text{ con } -q \le i \le p-1$$

# Representación

- **Cada dígito del número es más significativo que el que se encuentra a su derecha**, siendo el valor del número la suma acumulada de los productos de cada dígito por su peso:

$$N = \sum_{i=-q}^{p-1} a_i \cdot r^i$$

- Ejemplo:

$$(N)_r = (a_{p-1}\, a_{n-2} \ldots a_1\, a_0\, , a_{-1}\, a_{-2} \ldots a_{-q})_r$$

$$(1283)_{10} = (a_3 = 1\ a_2 = 2\ a_1 = 8\ a_0 = 3,\ a_{-i} = 0)_{10} =$$

$$= 1\times10^3 + 2\times10^2 + 8\times10^1 + 3\times10^0 = 1\times1000 + 2\times100 + 8\times10 + 3\times1$$

# Conversión entre bases

- **Conversión de base r a base s utilizando la aritmética de la base s:**
  - Esta conversión se usa normalmente para convertir de cualquier base a base 10.
- Dado un número **N** en base **r**: $N = \sum_{i=-q}^{p-1} a_i \cdot r^i = a_{p-1} \cdot r^{p-1} + \ldots + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + \ldots a_{-q} \cdot r^{-q}$
- la conversión consiste en evaluar directamente dicha expresión usando la aritmética de la base **s** tanto para la parte entera como para la fraccionaria.
- Ejemplo: convertir el número binario 1101,01 a base 10

$$(1101{,}01)_2 = (a_3 = 1\ a_2 = 1\ a_1 = 0\ a_0 = 1,\ a_{-1} = 0, a_{-2} = 1)_{10} =$$

.

$$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} =$$

$$= 1\times8 + 1\times4 + 0\times2 + 1\times1 + 0\times0{,}5 + 1\times0{,}25 = 13{,}25_{10}$$

- Ejemplo: convertir el número 14 en base 16 a base 10.

$$(14)_{16} = 1\times16^1 + 4\times16^0 = 1\times16 + 4\times1 = 20_{10}$$

# Conversión entre bases(1)

- **Conversión de base r a base s utilizando la aritmética de la base r:**
  - Esta conversión se usa normalmente para pasar de base 10 a cualquier otra base.
- Proceso de conversión:
  - **la parte entera se convierte mediante divisiones sucesivas entre $(s)_r$**
  - **la parte fraccionaria se convierte mediante productos sucesivos por $(s)_r$**
- Dado un número N en base r con parte entera y parte fraccionaria:

$$(N)_r = (N_E)_r + (N_F)_r$$

- en base s, dicho N se puede escribir del siguiente modo:

$$(N_E)_r = (a_{p-1} \cdot s^{p-1} + \ldots + a_1 \cdot s^1 + a_0 \cdot s^0)_r$$
$$(N_F)_r = (a_{-1} \cdot s^{-1} + \ldots + a_{-q} \cdot s^{-q})_r \qquad \forall i, a_i < s$$

# Conversión entre bases(2)

- **Proceso de conversión de la parte entera** (**divisiones sucesivas entre $(s)_r$**): Aplicando la aritmética de base r, se divide $(N_E)_r$ entre $(s)_r$, obteniéndose:

$$(N_E)_r = (a_{p-1} \cdot s^{p-1} + \ldots + a_1 \cdot s^1 + a_0 \cdot s^0)_r$$
$$\frac{(N_E)_r}{(s)_r} = (\boxed{a_{p-1} \cdot s^{p-2} + \ldots + a_1 \cdot s^0} + \boxed{a_0 \cdot s^{-1}})_r$$

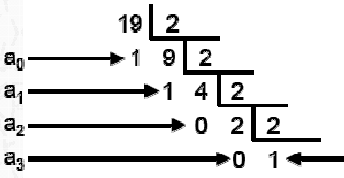
Cociente ← $\boxed{a_{p-1} \cdot s^{p-2} + \ldots + a_1 \cdot s^0}$ ; Resto ← $\boxed{a_0 \cdot s^{-1}}$

$$\frac{D}{d} = c + \frac{r}{d}$$

- El resto ($a_0$) es el primer dígito de $N_E$ en base s, representado en base r.
- Dependiendo de los valores relativos de s y r pueden ocurrir dos cosas:
  - si r > s el resto tiene un solo dígito y su representación es igual en base r que en base s.
  - si r < s el resto puede tener más de un dígito, por lo que hay que sustituir cada dígito por su representación en base s.
- Para calcular los demás dígitos de $N_E$ se repite el proceso de división, dividiendo el cociente entre $(s)_r$ de nuevo

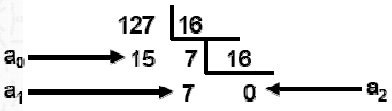
# Ejemplo

Ejemplo (r > s): convertir el número $(19)_{10}$ a binario.




$(19)_{10} = (10011)_2 = (16+2+1)_{10}$

Ejemplo (r < s): convertir el número $(127)_{10}$ a hexadecimal.




$(127)_{10} = (07(15))_{16} = (07F)_{16}$

- **Proceso de conversión de la parte fraccionaria** (**productos sucesivos por $(s)_r$**):
- Aplicando la aritmética de base r, se multiplica $(N_E)_r$ entre $(s)_r$, obteniéndose:

$$(N_F)_r = (a_{-1} \cdot s^{-1} + a_{-2} \cdot s^{-2} + \ldots + a_{-q} \cdot s^{-q})_r$$

$$(N_F)_r \times (s)_r = (\boxed{a_{-1}})_r + (a_{-2} \cdot s^{-1} + \ldots + a_{-q} \cdot s^{-q+1})_r$$

**Parte entera**

- El **primer dígito** de $N_F$ en base s, representado en base r, aparece como la parte entera del producto.
- Para calcular los demás dígitos de $N_F$ se repite el proceso de multiplicación
- **hasta que se obtiene 0 ó** cuando se obtiene el número de cifras que garantiza **la precisión requerida**.
- Es posible que el cambio de base de una fracción exacta producta una fracción periódica.

- Ejemplo: convertir el número $(0,1285)_{10}$ a base 4.

4 0,1285

| | |
|---|---|
| 0 | ,5140 x 4 |
| 2 | ,0560 x 4 |
| 0 | ,2240 x 4 |
| 0 | ,8960 x 4 |
| 3 | ,5840 x 4 |
| 2 | ,3360 x 4 |

$(0,1285)_{10} = (0,020032...)_4$

- Ejemplo: convertir el número $(0,3)_{10}$ a binario.

| | |
|---|---|
| | ,3 x 2 |
| 0 | ,6 x 2 |
| 1 | ,2 x 2 |
| 0 | ,4 x 2 |
| 0 | ,8 x 2 |
| 1 | ,6 x 2 |
| 1 | ,2 x 2 |
| 0 | ,4 x 2 |
| 0 | ,8 x 2 |
| 1 | ,6 x 2 |

$(0,3)_{10} = (0,010011001...)_2$

- Los procesos de conversión entre bases se simplifican cuando las bases de partida y de llegada son una potencia de la otra, es decir que s = rk ó r = sk cuando k>1.
- Casos más frecuentes:
  - Conversión entre binario y octal
  - Conversión entre binario y hexadecimal
- Los sistemas octal y hexadecimal son muy utilizados porque permiten representar grandes cadenas de dígitos binarios de forma abreviada.

# Aritmética

# Aritmética de base 2

- Las operaciones aritméticas en el sistema binario puro se realizan según tablas más sencillas que las equivalentes en el sistema decimal.

**Suma binaria**

| SUMA BINARIA (+) | A 0 | A 1 |
|---|---|---|
| B 0 | 0 | 1 |
| B 1 | 1 | 10 |

**Resta binaria**

| RESTA BINARIA (-) | A 0 | A 1 |
|---|---|---|
| B 0 | 0 | 1 |
| B 1 | 11 | 0 |

acarreo

| | | | |
|---|---|---|---|
| 1 | acarreos | 1 1 1 | |
| 9 | | 1 0 0 1 | |
| + 15 | + | 1 1 1 1 | |
| 24 | | 1 1 0 0 0 | |

| | | | |
|---|---|---|---|
| 83 | | 1 0 1 0 0 1 1 | minuendo |
| - 21 | - | 1 0 1 0 1 | sustraendo |
| 62 | acarreos | 1 1 1 1 | |
| | | 0 1 1 1 1 1 0 | diferencia |

# Producto base 2

**Producto binario**

| PRODUCTO BINARIO (×) | A = 0 | A = 1 |
|---|---|---|
| B = 0 | 0 | 0 |
| B = 1 | 0 | 1 |

```
   12          1 1 0 0   multiplicando
 × 6         ×   1 1 0   multiplicador
 ----        ----------
 72            0 0 0 0
             1 1 0 0     Productos
           1 1 0 0       parciales
         ---------------
         1 0 0 1 0 0 0   resultado
         64 + 8      = 72
```

# División base 2

- La división binaria se puede realizar igual que la decimal.
- En el caso de la binaria es más sencillo porque **se simplifica la elección de cada dígito del cociente** ya que sólo pueden ser 0 ó 1.
  - Si el dividendo parcial es mayor o igual que el divisor, el siguiente dígito del cociente es 1, si no es 0

```
112 | 8          dividendo  1 1 1 0 0 0 0 | 1 0 0 0   divisor
 14                         - 1 0 0 0     | 1 1 1 0   cociente
                             ---------
                             0 1 1 0 0
                           - 1 0 0 0
                             ---------
                             0 1 0 0 0
                           - 1 0 0 0
                             ---------
                             0 0 0 0 0
                           - 0 0 0 0
                             ---------
                             0 0 0 0   resto
```

# Codificación de números

# BCD(1)

- **Codificaciones decimales:**
  - Son las encargadas de representar los números del sistema decimal, representando cada dígito decimal mediante una cadena de dígitos binarios.
  - Para representar 10 dígitos distintos (0, 1, ..., 9) se necesita un número mínimo de 4 bits. Con 4 bits se pueden representar 16 dígitos distintos, luego se pueden asignar distintas combinaciones de bits a los 10 dígitos decimales
- **Codificación en BCD:**
  - Es la codificación decimal más sencilla y representa a los diez dígitos decimales asignándoles el código binario de su representación binaria pura con 4 bits. Con esa representación un número decimal se evalúa mediante la expresión:

$$b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 = b_3 \cdot 8 + b_2 \cdot 4 + b_1 \cdot 2 + b_0 \cdot 1$$

- Por esta razón al código BCD se le conoce también como código 8-4-2-1.

# BCD(2)

- **Equivalencia entre dígitos decimales y código BCD 8-4-2-1:**

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

- Es importante **no confundir la representación de un dígito decimal en BCD con un número binario**, ya que son representaciones distintas.
- Ejemplo: En BCD, el número decimal de dos dígitos 56 se escribe (5) y (6), es decir 0101 0110, mientras que en binario puro se escribe como 111000

# Exceso a 3

- Es una codificación derivada de la codificación BCD sin más que sumar 3 (0011) a la representación BCD de cada dígito decimal.

| BCD | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |
|---|---|---|---|---|---|---|---|---|---|---|
| Exceso-3 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 |

# Código Gray

- También conocido como código reflejado

| Gray 1 bit | Gray 2 bits | Gray 3 bits | Gray 4 bits | Decimal | Binario puro |
|---|---|---|---|---|---|
| 0 | 0 0 | 0 0 0 | 0 0 0 0 | 0 | 0000 |
| 1 | 0 1 | 0 0 1 | 0 0 0 1 | 1 | 0001 |
| | 1 1 | 0 1 1 | 0 0 1 1 | 2 | 0010 |
| | 1 0 | 0 1 0 | 0 0 1 0 | 3 | 0011 |
| | | 1 1 0 | 0 1 1 0 | 4 | 0100 |
| | | 1 1 1 | 0 1 1 1 | 5 | 0101 |
| | | 1 0 1 | 0 1 0 1 | 6 | 0110 |
| | | 1 0 0 | 0 1 0 0 | 7 | 0111 |
| | | | 1 1 0 0 | 8 | 1000 |
| | | | 1 1 0 1 | 9 | 1001 |
| | | | 1 1 1 1 | 10 | 1010 |
| | | | 1 1 1 0 | 11 | 1011 |
| | | | 1 0 1 0 | 12 | 1100 |
| | | | 1 0 1 1 | 13 | 1101 |
| | | | 1 0 0 1 | 14 | 1110 |
| | | | 1 0 0 0 | 15 | 1111 |

# Código Johnson

- Es un código progresivo, continuo y cíclico.

| Johnson 1 bit | Johnson 2 bits | Johnson 3 bits | Johnson 4 bits | Johnson 5 bits | Decimal |
|---|---|---|---|---|---|
| 0 | 00 | 000 | 0000 | 00000 | 0 |
| 1 | 01 | 001 | 0001 | 00001 | 1 |
| | 11 | 011 | 0011 | 00011 | 2 |
| | 10 | 111 | 0111 | 00111 | 3 |
| | | 110 | 1111 | 01111 | 4 |
| | | 100 | 1110 | 11111 | 5 |
| | | | 1100 | 11110 | 6 |
| | | | 1000 | 11100 | 7 |
| | | | | 11000 | 8 |
| | | | | 10000 | 9 |

# Codificación Alfanumérica

- Son las codificaciones encargadas de representar los caracteres alfabéticos, numéricos, signos de puntuación y signos de control mediante cadenas de dígitos binarios.
- Al menos deben representar 26 letras del alfabeto y 10 dígitos, es decir 36 caracteres, luego necesitan un mínimo de 6 bits. En realidad, se necesitan más caracteres, de forma que las codificaciones más utilizadas emplean 7 y 8 bits:

  ASCII y EBCDIC (**E**xtended **BCD I**nterchange **C**ode). Sólo se comenta el ASCII.

**ASCII:**

- El **Código estándar americano para el intercambio de información** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) es el código alfanumérico más extendido.
- El código **ASCII original utilizaba 7 bits** para representar 128 caracteres (0 hasta 7F en hexadecimal (0111 1111)) donde los primeros 32 caracteres son de control (no gráficos o invisibles) y los restantes son gráficos (visibles).

- En la actualidad ASCII es un **código de 8 bits**, también conocido como **ASCII extendido**, que aumenta su capacidad con 128 caracteres adicionales introducidos por IBM para los PC (estándar de facto).
- Estos caracteres van del 80 (1000 0000) hasta el FF (1111 1111) e incluyen caracteres alfabéticos no ingleses, símbolos no ingleses, letras griegas, símbolos matemáticos, caracteres para gráficos, caracteres gráficos de barra y caracteres sombreados.
- Los códigos ASCII extendidos presentan variaciones nacionales. El código más usado en nuestra zona geográfica es el Latin-1.
- Existe otro código para caracteres denominado **Unicode**, que emplea 16 bits por carácter, y tiene bastantes posibilidades de convertirse en el código estándar internacional en algunos años. Unicode no requiere variaciones nacionales, puesto que con 16 bits permite representar 32768 caracteres distintos.