

# ***Tema 2***

Soporte Lógico de los  
Ordenadores

# Índice

---

- Motivación
- Representación interna de la información
- Almacenamiento de datos: Importancia de las BD
- Principios de Sistemas Operativos
- Otras Aplicaciones

# Motivación

---

- Software: Conjunto de programas que dirigen el funcionamiento de un ordenador.
- Soporte Lógico = Software
- Datos-> Ordenador->Información

## **Representación de la Información: Índice**

---

- Representación interna de los datos: caracteres, números enteros, números reales, datos lógicos, imágenes y sonidos, códigos intermedios.
- Representación interna de las instrucciones.

# Representación de la información: Introducción

---

- Un ordenador sólo distingue entre dos estados: 0 y 1 (un bit)
- 0=No hay electricidad, 1=hay electricidad
- Sólo podemos trabajar con bits.
- Código: forma que toma la información que se intercambia.

# Representación de la información: Codificación

---

- Ejemplos codificaciones:
  - Persona-> DNI
  - Coche -> Matrícula
  - Morse
- Código Binario: ceros y unos
- Capacidad de representación:
  - **Digital  $10^n$**
  - **Binario  $2^n$**
- ¿Bits Necesarios para 27 símbolos?

# Codificación caracteres (1)

---

- Necesitamos representar:
  - Alfabeto (a,b,...,z,A,B,...,Z)
  - Números (0,1,2,...,9)
  - Caracteres de control: Espacio, escape, ...
  - Símbolos: (,),=,/,&,% , ....
- Alfanuméricos= Alfabeto+Números
- Codificación ASCII: 8 bits (7 + 1)

## **Codificación caracteres (2)**

---

- Código **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange — *Código Estadounidense Estándar para el Intercambio de Información*)
- ASCII 7+1 bits

## Codificación caracteres (3)

---

- 7 bits para caracteres.  $2^7=128$
- 1 bit detección de errores (código redundante)
- Paridad par:
  - 0= número par de unos
  - 1= número impar de unos
- Paridad impar:
  - 0= número impar de unos
  - 1= número par de unos

| Dec | Hx | Oct | Char                               | Dec | Hx | Oct | Html  | Chr          | Dec | Hx | Oct | Html  | Chr      | Dec | Hx | Oct | Html   | Chr        |
|-----|----|-----|------------------------------------|-----|----|-----|-------|--------------|-----|----|-----|-------|----------|-----|----|-----|--------|------------|
| 0   | 0  | 000 | <b>NUL</b> (null)                  | 32  | 20 | 040 | &#32; | <b>Space</b> | 64  | 40 | 100 | &#64; | <b>@</b> | 96  | 60 | 140 | &#96;  | <b>`</b>   |
| 1   | 1  | 001 | <b>SOH</b> (start of heading)      | 33  | 21 | 041 | &#33; | <b>!</b>     | 65  | 41 | 101 | &#65; | <b>A</b> | 97  | 61 | 141 | &#97;  | <b>a</b>   |
| 2   | 2  | 002 | <b>STX</b> (start of text)         | 34  | 22 | 042 | &#34; | <b>"</b>     | 66  | 42 | 102 | &#66; | <b>B</b> | 98  | 62 | 142 | &#98;  | <b>b</b>   |
| 3   | 3  | 003 | <b>ETX</b> (end of text)           | 35  | 23 | 043 | &#35; | <b>#</b>     | 67  | 43 | 103 | &#67; | <b>C</b> | 99  | 63 | 143 | &#99;  | <b>c</b>   |
| 4   | 4  | 004 | <b>EOT</b> (end of transmission)   | 36  | 24 | 044 | &#36; | <b>\$</b>    | 68  | 44 | 104 | &#68; | <b>D</b> | 100 | 64 | 144 | &#100; | <b>d</b>   |
| 5   | 5  | 005 | <b>ENQ</b> (enquiry)               | 37  | 25 | 045 | &#37; | <b>%</b>     | 69  | 45 | 105 | &#69; | <b>E</b> | 101 | 65 | 145 | &#101; | <b>e</b>   |
| 6   | 6  | 006 | <b>ACK</b> (acknowledge)           | 38  | 26 | 046 | &#38; | <b>&amp;</b> | 70  | 46 | 106 | &#70; | <b>F</b> | 102 | 66 | 146 | &#102; | <b>f</b>   |
| 7   | 7  | 007 | <b>BEL</b> (bell)                  | 39  | 27 | 047 | &#39; | <b>'</b>     | 71  | 47 | 107 | &#71; | <b>G</b> | 103 | 67 | 147 | &#103; | <b>g</b>   |
| 8   | 8  | 010 | <b>BS</b> (backspace)              | 40  | 28 | 050 | &#40; | <b>(</b>     | 72  | 48 | 110 | &#72; | <b>H</b> | 104 | 68 | 150 | &#104; | <b>h</b>   |
| 9   | 9  | 011 | <b>TAB</b> (horizontal tab)        | 41  | 29 | 051 | &#41; | <b>)</b>     | 73  | 49 | 111 | &#73; | <b>I</b> | 105 | 69 | 151 | &#105; | <b>i</b>   |
| 10  | A  | 012 | <b>LF</b> (NL line feed, new line) | 42  | 2A | 052 | &#42; | <b>*</b>     | 74  | 4A | 112 | &#74; | <b>J</b> | 106 | 6A | 152 | &#106; | <b>j</b>   |
| 11  | B  | 013 | <b>VT</b> (vertical tab)           | 43  | 2B | 053 | &#43; | <b>+</b>     | 75  | 4B | 113 | &#75; | <b>K</b> | 107 | 6B | 153 | &#107; | <b>k</b>   |
| 12  | C  | 014 | <b>FF</b> (NP form feed, new page) | 44  | 2C | 054 | &#44; | <b>,</b>     | 76  | 4C | 114 | &#76; | <b>L</b> | 108 | 6C | 154 | &#108; | <b>l</b>   |
| 13  | D  | 015 | <b>CR</b> (carriage return)        | 45  | 2D | 055 | &#45; | <b>-</b>     | 77  | 4D | 115 | &#77; | <b>M</b> | 109 | 6D | 155 | &#109; | <b>m</b>   |
| 14  | E  | 016 | <b>SO</b> (shift out)              | 46  | 2E | 056 | &#46; | <b>.</b>     | 78  | 4E | 116 | &#78; | <b>N</b> | 110 | 6E | 156 | &#110; | <b>n</b>   |
| 15  | F  | 017 | <b>SI</b> (shift in)               | 47  | 2F | 057 | &#47; | <b>/</b>     | 79  | 4F | 117 | &#79; | <b>O</b> | 111 | 6F | 157 | &#111; | <b>o</b>   |
| 16  | 10 | 020 | <b>DLE</b> (data link escape)      | 48  | 30 | 060 | &#48; | <b>0</b>     | 80  | 50 | 120 | &#80; | <b>P</b> | 112 | 70 | 160 | &#112; | <b>p</b>   |
| 17  | 11 | 021 | <b>DC1</b> (device control 1)      | 49  | 31 | 061 | &#49; | <b>1</b>     | 81  | 51 | 121 | &#81; | <b>Q</b> | 113 | 71 | 161 | &#113; | <b>q</b>   |
| 18  | 12 | 022 | <b>DC2</b> (device control 2)      | 50  | 32 | 062 | &#50; | <b>2</b>     | 82  | 52 | 122 | &#82; | <b>R</b> | 114 | 72 | 162 | &#114; | <b>r</b>   |
| 19  | 13 | 023 | <b>DC3</b> (device control 3)      | 51  | 33 | 063 | &#51; | <b>3</b>     | 83  | 53 | 123 | &#83; | <b>S</b> | 115 | 73 | 163 | &#115; | <b>s</b>   |
| 20  | 14 | 024 | <b>DC4</b> (device control 4)      | 52  | 34 | 064 | &#52; | <b>4</b>     | 84  | 54 | 124 | &#84; | <b>T</b> | 116 | 74 | 164 | &#116; | <b>t</b>   |
| 21  | 15 | 025 | <b>NAK</b> (negative acknowledge)  | 53  | 35 | 065 | &#53; | <b>5</b>     | 85  | 55 | 125 | &#85; | <b>U</b> | 117 | 75 | 165 | &#117; | <b>u</b>   |
| 22  | 16 | 026 | <b>SYN</b> (synchronous idle)      | 54  | 36 | 066 | &#54; | <b>6</b>     | 86  | 56 | 126 | &#86; | <b>V</b> | 118 | 76 | 166 | &#118; | <b>v</b>   |
| 23  | 17 | 027 | <b>ETB</b> (end of trans. block)   | 55  | 37 | 067 | &#55; | <b>7</b>     | 87  | 57 | 127 | &#87; | <b>W</b> | 119 | 77 | 167 | &#119; | <b>w</b>   |
| 24  | 18 | 030 | <b>CAN</b> (cancel)                | 56  | 38 | 070 | &#56; | <b>8</b>     | 88  | 58 | 130 | &#88; | <b>X</b> | 120 | 78 | 170 | &#120; | <b>x</b>   |
| 25  | 19 | 031 | <b>EM</b> (end of medium)          | 57  | 39 | 071 | &#57; | <b>9</b>     | 89  | 59 | 131 | &#89; | <b>Y</b> | 121 | 79 | 171 | &#121; | <b>y</b>   |
| 26  | 1A | 032 | <b>SUB</b> (substitute)            | 58  | 3A | 072 | &#58; | <b>:</b>     | 90  | 5A | 132 | &#90; | <b>Z</b> | 122 | 7A | 172 | &#122; | <b>z</b>   |
| 27  | 1B | 033 | <b>ESC</b> (escape)                | 59  | 3B | 073 | &#59; | <b>;</b>     | 91  | 5B | 133 | &#91; | <b>[</b> | 123 | 7B | 173 | &#123; | <b>{</b>   |
| 28  | 1C | 034 | <b>FS</b> (file separator)         | 60  | 3C | 074 | &#60; | <b>&lt;</b>  | 92  | 5C | 134 | &#92; | <b>\</b> | 124 | 7C | 174 | &#124; | <b> </b>   |
| 29  | 1D | 035 | <b>GS</b> (group separator)        | 61  | 3D | 075 | &#61; | <b>=</b>     | 93  | 5D | 135 | &#93; | <b>]</b> | 125 | 7D | 175 | &#125; | <b>}</b>   |
| 30  | 1E | 036 | <b>RS</b> (record separator)       | 62  | 3E | 076 | &#62; | <b>&gt;</b>  | 94  | 5E | 136 | &#94; | <b>^</b> | 126 | 7E | 176 | &#126; | <b>~</b>   |
| 31  | 1F | 037 | <b>US</b> (unit separator)         | 63  | 3F | 077 | &#63; | <b>?</b>     | 95  | 5F | 137 | &#95; | <b>_</b> | 127 | 7F | 177 | &#127; | <b>DEL</b> |

fuelle: [www.tablaascii.com.ar](http://www.tablaascii.com.ar)

## **Tipos codificación caracteres**

---

- **ASCII:** 7+1 bits, representa abecedario inglés.
- **ASCII extendido:** 8 bits, uno para cada un conjunto de lenguas.
- **Unicode:** asigna código único a cada carácter para todos los lenguajes.
- **UTF-8:** *Unicode Trasformation Format*, 8 bits variable. Representa cualquier carácter Unicode. Longitud variable. Incluye ASCII 7 bits.

# ASCII Extendido

|     |   |     |   |     |   |     |   |     |   |     |   |     |   |     |   |
|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|
| 128 | Ç | 144 | É | 160 | á | 176 | ☒ | 193 | ⊥ | 209 | ≠ | 225 | β | 241 | ± |
| 129 | ü | 145 | æ | 161 | í | 177 | ☑ | 194 | ⊤ | 210 | π | 226 | Γ | 242 | ≥ |
| 130 | é | 146 | Æ | 162 | ó | 178 | ☐ | 195 | ⊥ | 211 | ⊥ | 227 | π | 243 | ≤ |
| 131 | â | 147 | ô | 163 | ú | 179 |   | 196 | — | 212 | ⊥ | 228 | Σ | 244 | ∫ |
| 132 | ä | 148 | ö | 164 | ñ | 180 | ⊥ | 197 | ⊥ | 213 | ⊥ | 229 | σ | 245 | ∫ |
| 133 | à | 149 | ò | 165 | Ñ | 181 | ⊥ | 198 | ⊥ | 214 | ⊥ | 230 | μ | 246 | ÷ |
| 134 | â | 150 | û | 166 | ª | 182 | ⊥ | 199 | ⊥ | 215 | ⊥ | 231 | τ | 247 | ≈ |
| 135 | ç | 151 | ù | 167 | º | 183 | ⊥ | 200 | ⊥ | 216 | ⊥ | 232 | Φ | 248 | ° |
| 136 | ê | 152 | — | 168 | ¿ | 184 | ⊥ | 201 | ⊥ | 217 | ∫ | 233 | ⊕ | 249 | . |
| 137 | ë | 153 | Ö | 169 | — | 185 | ⊥ | 202 | ⊥ | 218 | ∫ | 234 | Ω | 250 | . |
| 138 | è | 154 | Û | 170 | ¬ | 186 | ⊥ | 203 | ≠ | 219 | ■ | 235 | δ | 251 | √ |
| 139 | ï | 156 | £ | 171 | ½ | 187 | ⊥ | 204 | ⊥ | 220 | ■ | 236 | ∞ | 252 | — |
| 140 | î | 157 | ¥ | 172 | ¼ | 188 | ⊥ | 205 | = | 221 | ■ | 237 | φ | 253 | ² |
| 141 | ì | 158 | — | 173 | ¡ | 189 | ⊥ | 206 | ⊥ | 222 | ■ | 238 | ε | 254 | ■ |
| 142 | Ä | 159 | f | 174 | « | 190 | ⊥ | 207 | ⊥ | 223 | ■ | 239 | ∩ | 255 |   |
| 143 | Å | 192 | L | 175 | » | 191 | ∫ | 208 | ⊥ | 224 | α | 240 | ≡ |     |   |

fuelle: [www.tablaascii.com.ar](http://www.tablaascii.com.ar)

# Codificación enteros (1)

---

- ¿Caracteres ASCII para representar enteros?
- Mayor consumo de memoria:
  - 10 números:
    - 1 carácter ASCII (8 bits)
    - 4 bits ( $2^4=16>10>2^3=8$ )
  - 100 números:
    - 2 caracteres ASCII (16 bits)
    - 7 bits ( $2^7=128>100>2^6=64$ )
- ALU Trabaja con número binarios

## Codificación enteros (2)

---

- Sistemas de numeración:
  - Binaria: 0 ó 1 (Base 2)
  - Octal; 0,1,2,3,4,5,6 ó 7 (Base 8)
  - Decimal: 0,1,2,3,4,5,6,7,8 ó 9 (Base 10)
  - Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (Base 16)
- Base = Número de dígitos

## Codificación enteros (3)

---

- Pasar a decimal:

- $i$  = posición dígito
  - $d_i$  = dígito  $i$ -ésimo
  - $b$  = base
- $$\sum_{i=0} d_i \times b^i$$

- $255)_{10} = 5 \cdot 10^0 + 5 \cdot 10^1 + 2 \cdot 10^2 = 255)_{10}$

- $255)_8 = 5 \cdot 8^0 + 5 \cdot 8^1 + 2 \cdot 8^2 = 173)_{10}$

- $11010)_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 = 26)_{10}$

## Codificación enteros (4)

---

- Para pasar un número decimal a número en base b.
- Dividir el número sucesivamente por b hasta que el número sea menor que b
- El número está formado por los restos y por el último cociente: Primero el cociente y luego los restos (últimos primero)

$$\begin{array}{r} 26 \quad \underline{) 2} \\ 06 \quad \underline{) 13} \\ \underline{0} \quad \underline{) 1} \\ \quad 6 \quad \underline{) 2} \\ \quad \underline{0} \quad \underline{) 3} \\ \quad \quad 3 \quad \underline{) 2} \\ \quad \quad \underline{1} \quad \underline{) 1} \\ \quad \quad \quad 1 \end{array}$$

## Codificación enteros (5)

---

- Números negativos: Bit de (0=positivo, 1=negativo).
- Números negativos implican Números más pequeños
- 7 bits  $2^7$ =máximo 128 (-128..0..127)
- 8 bits  $2^8$ =máximo 256 (0....256)
- Tamaños usados: 2 bytes, 4 bytes, 8 bytes.

# Codificación reales (1)

---

- Números reales son números con decimales
- Notación científica (normalizada):
  - Signo: positivos y negativos
  - Mantisa: cifras que componen el número
  - Exponente: potencia de 10 asociada
- Representación: 1 bit de signo + Mantisa + Exponente (como números enteros en binario)

## Codificación reales (2)

---

- Ejemplos:
  - $23.4 = 0.234 \times 10^2 = (+, 234, 2) = 0-11101010-010$
  - $-0.0076 = -0.76 \times 10^{-2} = (-, 76, -2) = 1-1001100-110$
  - $15 = 0.15 \times 10^2 = (+, 15, 2) = 0-1111-010$
- Tamaños usados:
  - Simple precisión: 1bit signo + 23bit mantisa + 8bit exponente (4 bytes)
  - Doble precisión: 1bit signo + 52bit mantisa + 11bit exponente (8 bytes)

## Codificación lógicos

- Valores lógicos: Verdadero(1) o Falso(0) [1 bit]
- Uso muy común: ¿Número mayor? ¿se ha pulsado una tecla? ¿ha recibido correctamente un byte?
- AND (Y lógico): ¿Se ha pulsado ALT **y** F4?
- OR (O lógico): ¿Se ha escogido opción salir o se ha pinchado la x?
- NOT (Negación): Not F=V, Not V=F. Ej.: Si no se ha pulsado Enter ....

|     |   |   |
|-----|---|---|
| AND | F | V |
| F   | F | F |
| V   | F | V |

|    |   |   |
|----|---|---|
| OR | F | V |
| F  | F | V |
| V  | V | V |

# Codificación Imágenes

---

- **Cabecera:**
  - Tamaño en píxeles (puntos): Ancho y alto
  - Color o blanco y negro.
  - Transparencias. Formato. Compresión, etc.
- **Puntos:**
  - Rojo, Verde y Azul (RGB=Red, Green, Blue)
  - Nivel de gris
- **Más bits implican colores más precisos**

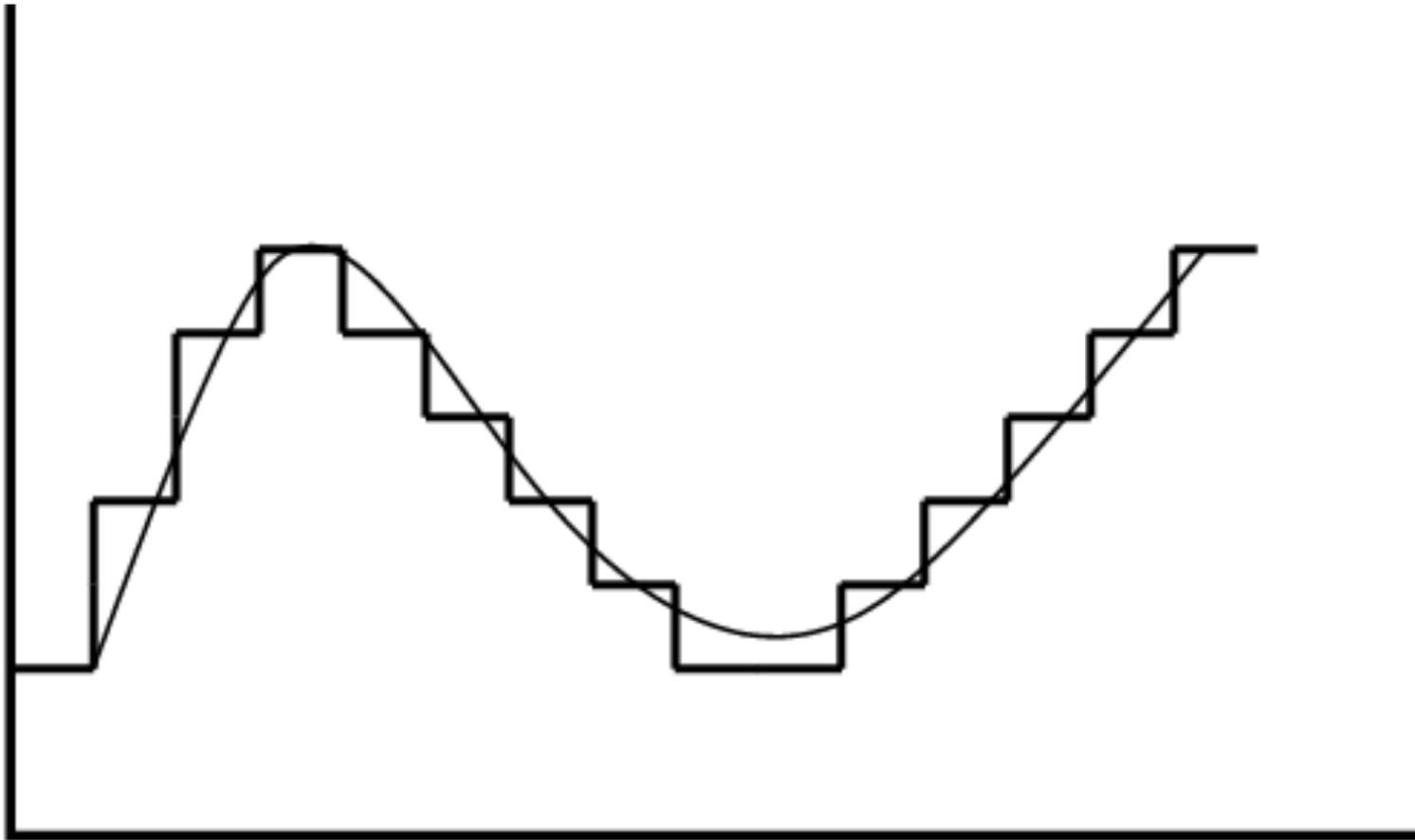
## **Codificación Sonido (1)**

---

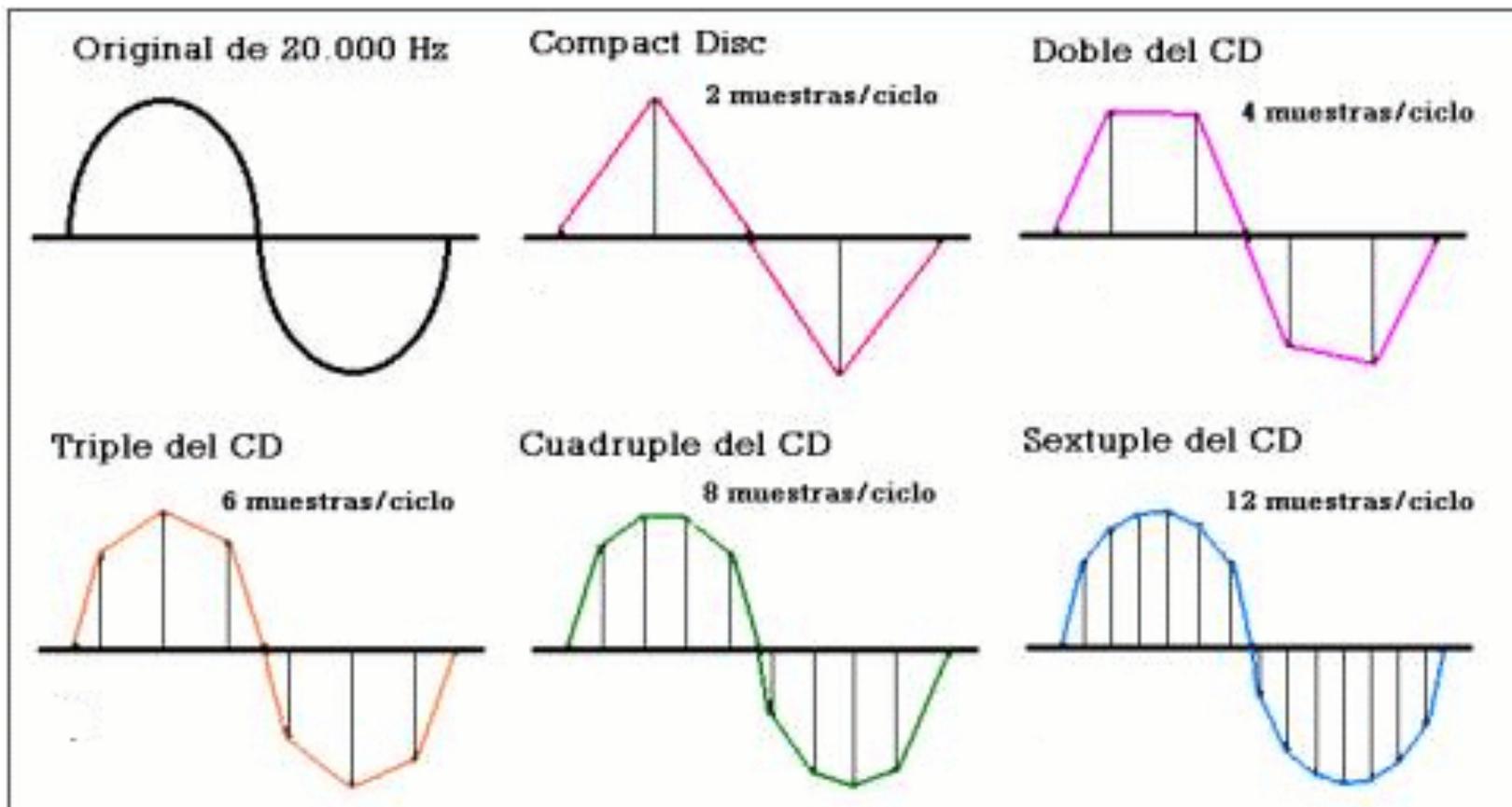
- Sonido es una onda continúa. Frecuencias.
- Se toman muestras y se convierten en números.
- Más muestras y más bits, mayor calidad de sonido.
- También tiene una cabecera indicando formato de sonido, muestras por segundo, calidad de cada muestra, etc.

## Codificación Sonido (2)

---



# Codificación Sonido (3)



## Códigos intermedios: Octal

---

- El código octal permite utilizar los dígitos entre 0 y 7
- Binario es  $2^1 = 2$  Dígitos (0 y 1), octal  $2^3 = 8$  Dígitos (0,1,2,3,4,5,6 y 7).
- Cada dígito en octal se corresponde a 3 dígitos binarios
- $25)_8 = 010101)_2$

| Dígito | Sec. binaria |
|--------|--------------|
| 0      | 000          |
| 1      | 001          |
| 2      | 010          |
| 3      | 011          |
| 4      | 100          |
| 5      | 101          |
| 6      | 110          |
| 7      | 111          |

# Códigos intermedios: Hexadecimal

---

- El código Hexadecimal permite usar 16 dígitos:  
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Cada dígito Hexadecimal ( $2^4=16$ ) se corresponde a 4 dígitos binarios ( $2^1=2$ )
- $25)_{16} = 00100101)_2$

| Dígito | Sec. binaria |
|--------|--------------|
| 0      | 0000         |
| 1      | 0001         |
| 2      | 0010         |
| 3      | 0011         |
| 4      | 0100         |
| 5      | 0101         |
| 6      | 0110         |
| 7      | 0111         |
| 8      | 1000         |
| 9      | 1001         |
| A      | 1010         |
| B      | 1011         |
| C      | 1100         |
| D      | 1101         |
| E      | 1110         |
| F      | 1111         |

# Representación interna instrucciones

---

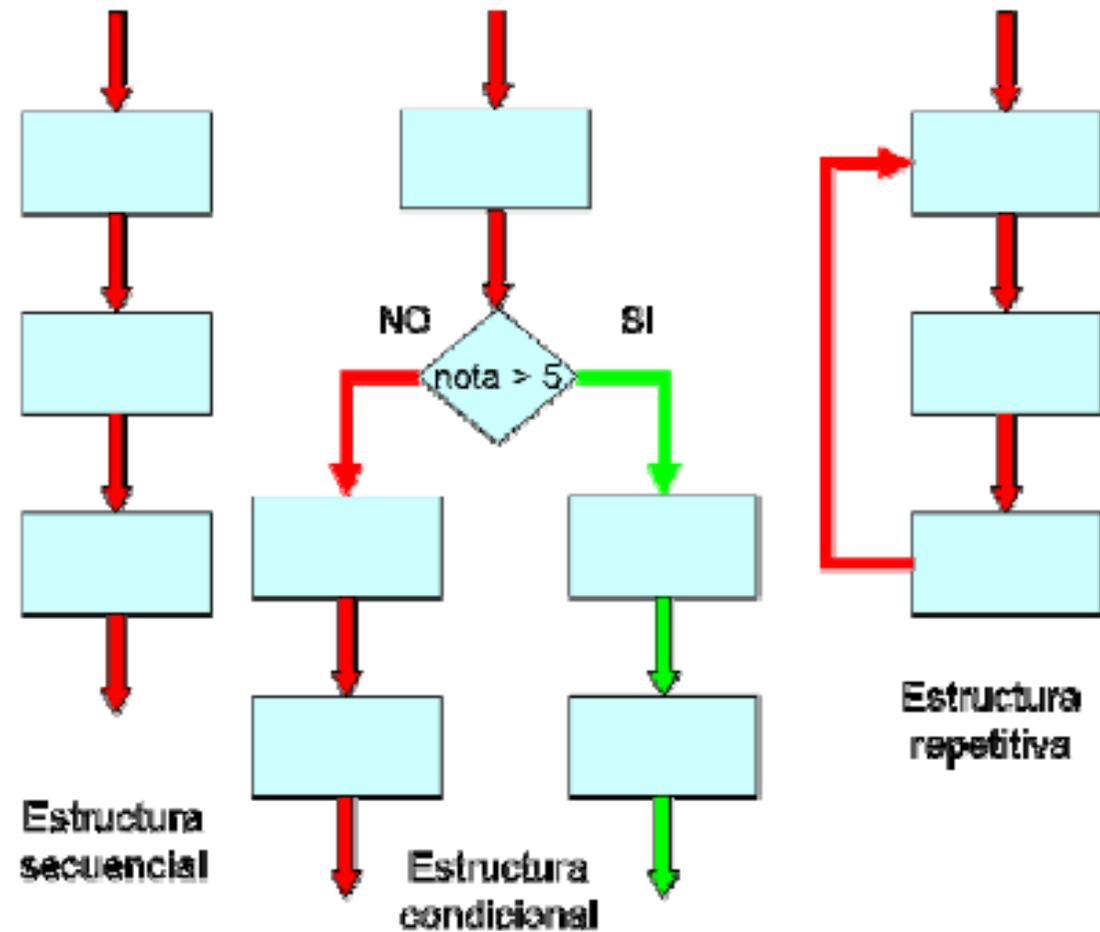
- Un programa se compone de instrucciones.
- Se representan mediante 0's y 1s. Código binario o Código máquina.
- Cada codificación depende del procesador.
- Las instrucciones deben ser sencillas. Por ejemplo, *suma A,B* y no *Dibuja una casa*.

## **Tipos de instrucciones (1)**

---

- Instrucciones de transferencia: Mover los datos de un sitio a otro.
- Instrucciones de tratamiento de datos: Operaciones Aritmético-Lógicas
- Instrucciones de control: Permiten saltar el orden secuencial de un programa.

## Tipos de instrucciones (2)



## Partes de una instrucción

---

- Código de operación: Codifican una instrucción determinada.
- Argumentos: Datos de la instrucción:  
Pueden ser:
  - Direcciones de memoria con datos.
  - Datos a usar directamente.
  - Direcciones de memoria con la siguiente instrucción.

## **Problemas del código máquina**

---

Código máquina, código binario o lenguaje de bajo nivel.

- Secuencia de ceros y unos.
- Dependiente del procesador.
- Incomprensibles.
- Realización de tareas complejas supone un gran esfuerzo.

## **Lenguajes alto nivel y bajo nivel**

---

Si(notas $\geq$ 5)entonces

    nota=aprobado

Sino

    nota=suspenso

Escribe(nota)

Mueve M[22], R1

Mueve 5,R2

Compara R1,R2

SaltaMenor M[10]

Mueve M[30],x41

Mueve M[31],x70

Mueve M[32],x72

Mueve M[33],x6F, etc.

# Lenguajes de alto nivel (1)

---

- Más fáciles de entender.
- Más compatibles.
- Operaciones más complejas.
- Necesidad de un **Traductor** que pase de lenguaje de alto nivel a código máquinas:
  - **Compilador**: Se hace la conversión una vez de forma completa
  - **Intérprete**: Se hace la conversión cada vez que se llama, sentencia a sentencia.

## **Lenguajes de alto nivel (2)**

---

- Ejemplos: Java (Interpretado), C (Compilado), C++(Compilado), C# (Interpretado), Pascal(Compilado), PHP (Interpretado), Perl (Interpretado) , JavaScript (Interpretado), Basic (Interpretado), Cobol(compilado), Fortran(Compilado)

# Entornos de programación

---

- Editor de texto
- Compilador/traductor
- Depurador
- Sistemas de ayuda
- Herramientas varias (generación de documentación, generación de interfaces de usuario, etc.)

## **Almacenamiento de datos**

---

- El usuario decide como guardar sus datos
- Archivo o fichero es la unidad elemental de almacenamiento.
- Carpetas o directorios son agrupaciones de archivos y/o carpetas.

## Almacenamiento de datos (2)

---

- Ordenar datos en registros y, dentro, por campos

| Nombre  | Apellidos         | Edad | Dirección      |
|---------|-------------------|------|----------------|
| Antonio | López<br>Martínez | 25   | C/ Mesones, 37 |
| ...     | ...               | ...  | ...            |
| ...     | ...               | ...  | ...            |
| Juan    | Oliver<br>Pérez   | 20   | C/ Ronda, 18   |

- Por ejemplo, cada registro es un alumno y dentro de cada campo guardamos datos de ese alumno: nombre, apellidos, teléfono, foto, etc.
- Hacer un programa para añadir, borrar, modificar alumnos o sacar listados.

# Importancia de la Bases de Datos

---

- Datos de universidad, matrículas, asignaturas matriculadas, asignaturas impartidas, profesores, cursos, nóminas... ¿Hacer más programas?.
- Utilización de programas para gestión de Bases de Datos: facilitan la construcción, almacenamiento, gestión y utilización de las BD.

## Importancia de la Bases de Datos (2)

---

- Almacenamiento de la información usando **tablas** (ficheros), compuestas por **tuplas** (registros) y éstas compuestas por atributos (campos). Relaciones como tablas.
- Consultas: Permiten recuperar la información fácilmente.
- Formularios: Permiten gestionar fácilmente los datos.
- Informes.

# Principios de Sistemas Operativos

---

Conjunto de programas con los siguientes objetivos principales:

- Facilitar al usuario y aplicaciones el uso del hardware
- Gestionar los recursos eficientemente entre usuarios y programas

## **Tareas del Sistema Operativo (1)**

---

- Uso del procesador: todo programa necesita pasar por la CPU para funcionar.
- Gestión de memoria principal: Repartir los programas en memoria principal.
- Gestión de los dispositivos de almacenamiento masivo (sistema de ficheros)

## **Tareas del Sistema Operativo (2)**

---

- Control y comunicación con los periféricos. Necesidad de Controladores (drivers).
- Protección de los recursos: (memoria, CPU, sistema de ficheros, acceso a los periféricos, etc.)
- Interfaz de usuario: Uso fácil, transparente y unificado para los usuarios

## **Tipos de Sistemas Operativos**

---

- **Multiusuario/Monousuario:** En los multiusuario cada usuario tiene una cuenta (nombre, contraseña, carpeta, permisos, etc.)
- **Monotarea/Multitarea**
- **Interfaz de usuario en modo texto/gráfica.**

## Otras aplicaciones (1)

---

- Procesadores de texto. Ej.: Microsoft Word
- Hojas de Cálculo. Ej.= Microsoft Excel
- Gestores de BD. Ej.=Microsoft Access
- Presentaciones. Ej.=Microsoft PowerPoint
- Paquetes ofimáticos. Ej.= Microsoft Office, OpenOffice, Google Docs, iWorks.

## **Otras aplicaciones (2)**

---

- Aplicaciones de diseño gráfico. Ej.: Adobe photoshop, AutoCad, 3D Studio Max
- Entornos de programación.
- Software educativo y de entretenimiento.
- Otros (antivirus, reproductores multimedia, visualizadores de documentos, Internet, etc.)