Clase 04 - Representación Numérica

IIC1001 - Algoritmos y Sistemas Computacionales

Cristian Ruz – cruz@uc.cl

Jueves 19-Marzo-2026

Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile

Contenidos

Contenidos

Contacto

🐐

vicente.cabra@uc.cl

Vicente Cabra

Ayudante



🐸

alejandro.tapia@uc.cl

Alejandro Tapia

Ayudante



🥇

gabriel.sandoval@uc.cl

Gabriel Sandoval

Ayudante jefe



Contenidos

Temas del curso

Sistemas computacionales

- Representación datos, números y compresión
- Funcionamiento hardware, procesadores y memoria.
- Funcionamiento de sistemas operativos: ejemplo scheduling
- Funcionamiento de Internet
- Herramientas computacionales: github + latex

Algoritmos

- Algoritmos y resolución de problemas
- Eficiencia algorítmica
- Estructuras secuenciales y ordenamiento
- Grafos y árboles

Contenidos

Algoritmos

¿Qué hace este algoritmo?

Algorithm 1.1: A simple Stock Span algorithm.

`SimpleStockSpan(`*quotes*`)` $\rightarrow$ *spans*

Input: *quotes*, an array with n stock price quotes

Output: *spans*, an array with n stock price spans

```
1  spans ← CreateArray(n)
2  for i ← 0 to n do
3      k ← 1
4      span_end ← FALSE
5      while i − k ≥ 0 and not span_end do
6          if quotes[i − k] ≤ quotes[i] then
7              k ← k + 1
8          else
9              span_end ← TRUE
10     spans[i] ← k
11 return spans
```

Un programa

Un programa ... en C

```c
#include <stdio.h>

int main() {
    printf("Hello, world\n");
    return 0;
}
```

Un programa

Un programa … en bits

```
00100011 01101001 01101110 01100011 01101100 01110101 01100100 01100101
00100000 00111100 01110011 01110100 01100100 01101001 01101111 00101110
01101000 00111110 00001010 00001010 01101001 01101110 01110100 00100000
01101101 01100001 01101001 01101110 00101000 00101001 00100000 01111011
00001010 00100000 00100000 00100000 00100000 01110000 01110010 01101001
01101110 01110100 01100110 00101000 00100010 01001000 01100101 01101100
01101100 01101111 00101100 00100000 01110111 01101111 01110010 01101100
01100100 01011100 01101110 00100010 00101001 00111011 00001010 00100000
00100000 00100000 00100000 01110010 01100101 01110100 01110101 01110010
01101110 00100000 00110000 00111011 00001010 01111101
```

Un programa

Un programa … en caracteres

```
 #   i   n   c   l   u   d   e  <sp> <   s   t   d   i   o   .
 35 105 110  99 108 117 100 101  32  60 115 116 100 105 111 46

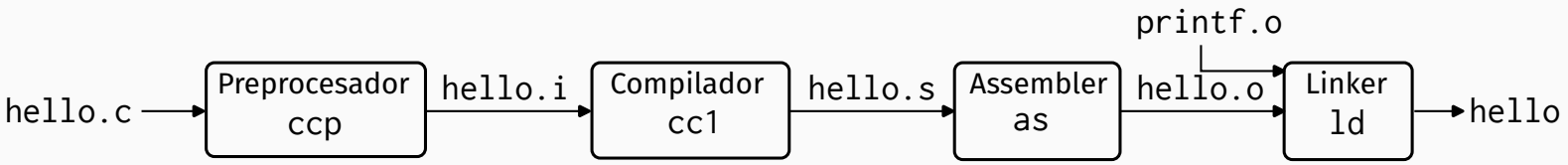
 h   >  \n  \n   i   n   t  <sp> m   a   i   n   (   )  <sp> {
104  62  10  10 105 110 116  32 109  97 105 110  40  41  32 123

\n  <sp><sp><sp><sp> p   r   i   n   t   f   (   "   H   e   l
 10  32  32  32  32 112 114 105 110 116 102  40  34  72 101 108

 l   o   ,  <sp> w   o   r   l   d   \   n   "   )   ;  \n  <sp>
108 111  44  32 119 111 114 108 100  92 110  34  41  59  10  32

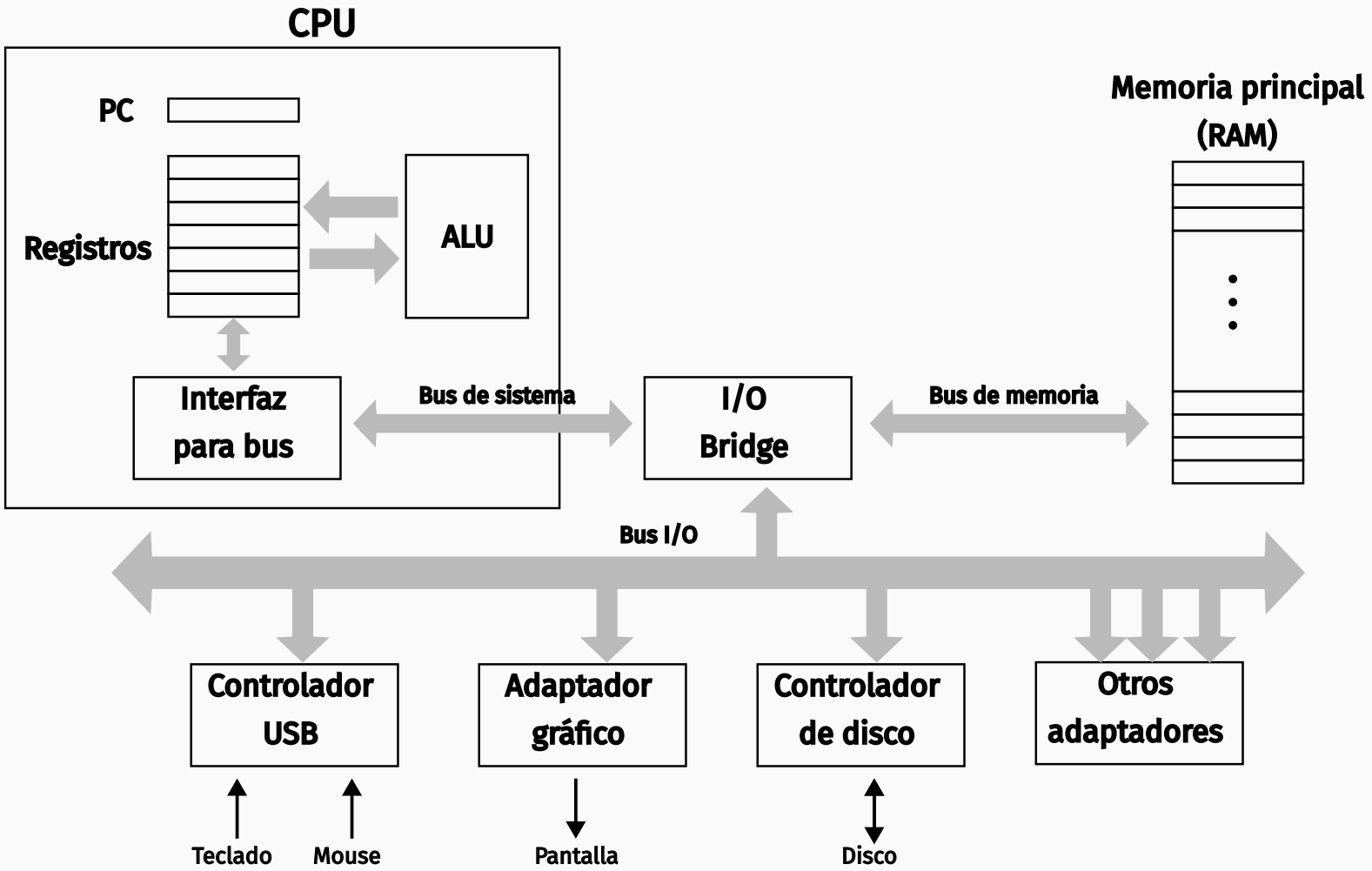
<sp><sp><sp> r   e   t   u   r   n  <sp> 0   ;  \n   }  \n
 32  32  32 114 101 116 117 114 110  32  48  59  10 125 10
```

El proceso de compilación



Arquitectura

Para llegar a ejecutar en ...

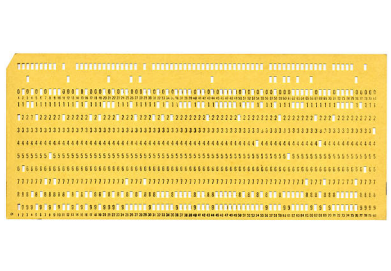


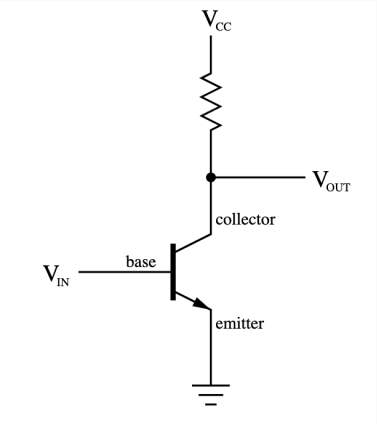
Contenidos

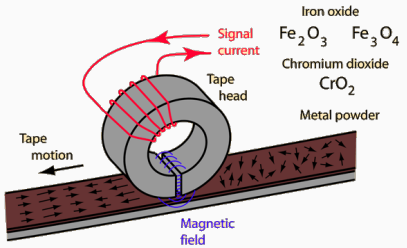
¿Cómo representar información?

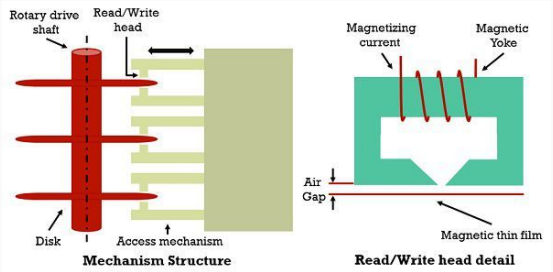
Bits

- Los computadores modernos funcionan en base a **bits**
- Un **bit** puede tener dos valores: 0 ó 1
- Diversas maneras de representar 0 y 1:
  - Hoyos en tarjetas
  - Voltaje alto o bajo en un circuito
  - Campo magnético en un sentido u otro
  - Presencia o ausencia de una señal
- La tecnología moderna permite agrupar miles, millones, billones, trillones de estas unidades de información
  - Poco espacio
  - Alta densidad
  - Bajo consumo energético
  - Fácil de construir









¿Cómo representar información?

Bits

- Los bits son muy convenientes para representar información
- Pero los humanos usamos 10 valores
- Distintos sistemas de representación numérica
  - Computadores: $0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, \ldots$
  - Humanos: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, \ldots$
- Ambos sistemas siguen reglas similiares, pero con diferente cantidad de dígitos
  - Computadores: representación binaria, base 2
  - Humanos: representación decimal, base 10
  - Hay muchas más ...



Sistemas de representación numérica

Los números (naturales) son infinitos.

- Usamos D símbolos (dígitos) para representar lo que podemos.
- Cuando se nos acaban los dígitos, agregamos una **posición** más.
- La nueva posición indica cuántas veces hemos pasado todos los dígitos de la posición anterior.
- En base 10, usamos 10 dígitos ($D = 10$)

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, \ldots$$

- Podría escribirse como:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1 \times D, 1 \times D + 1, 1 \times D + 2, 1 \times D + 3, \ldots, 2 \times D, 2 \times D + 1, \ldots$$

- Un número en base 10 como 13425, es una abreviación de la expresión:

$$1 \times 10000 + 3 \times 1000 + 4 \times 100 + 2 \times 10 + 5 \times 1$$

- Pero también se puede escribir usando la base $D = 10$:

$$1 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

- Estos sistemas se conocen como sistemas de representación posicional

Lo mismo pero con $D = 2$

- Con $D = 2$ solo tenemos dos dígitos: 0 y 1.
- Cuando se nos acaban los dígitos, agregamos una **posición** más.
- Usamos la misma idea que cuando teníamos $D = 10$.

$$0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, \ldots$$

- Podría escribirse como:

$$0, 1, 1\times D, 1\times D+1, 1\times D\times D, 1\times D\times D+1, 1\times D\times D+1\times D, 1\times D\times D+1\times D+1\times 1, \ldots$$

$$0, 1, 1 \times D, 1 \times D + 1, 1 \times D^2, 1 \times D^2 + 1, 1 \times D^2 + 1 \times D, 1 \times D^2 + 1 \times D + 1, \ldots$$

$$0, 1, 1 \times 2, 1 \times 2 + 1, 1 \times 2^2, 1 \times 2^2 + 1, 1 \times 2^2 + 1 \times 2, 1 \times 2^2 + 1 \times 2 + 1, \ldots$$

$$0, 1, 1 \times 2, 1 \times 2 + 1, 1 \times 4, 1 \times 4 + 1, 1 \times 4 + 1 \times 2, 1 \times 4 + 1 \times 2 + 1, \ldots$$

- Un número en base 2 como 101010, es una abreviación de la expresión:

$$1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$$

- Pero también se puede escribir usando la base $D = 2$:

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

Sistemas de representación numérica

Ahora podemos hacer algunas equivalencias:

Base 10	Descomposición	Base 2	Descomposición
0	0×10^0	0	0×2^0
1	1×10^0	1	1×2^0
2	2×10^0	10	$1 \times 2^1 + 0 \times 2^0$
3	3×10^0	11	$1 \times 2^1 + 1 \times 2^0$
4	4×10^0	100	$1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
5	5×10^0	101	$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
6	6×10^0	110	$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
7	7×10^0	111	$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
8	8×10^0	1000	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
9	9×10^0	1001	$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
10	$1 \times 10^1 + 0 \times 10^0$	1010	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
11	$1 \times 10^1 + 1 \times 10^0$	1011	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
12	$1 \times 10^1 + 2 \times 10^0$	1100	$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
13	$1 \times 10^1 + 3 \times 10^0$	1101	$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
14	$1 \times 10^1 + 4 \times 10^0$	1110	$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
15	$1 \times 10^1 + 5 \times 10^0$	1111	$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
16	$1 \times 10^1 + 6 \times 10^0$	10000	$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
17	$1 \times 10^1 + 7 \times 10^0$	10001	$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
18	$1 \times 10^1 + 8 \times 10^0$	10010	$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
19	$1 \times 10^1 + 9 \times 10^0$	10011	$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
20	$2 \times 10^1 + 0 \times 10^0$	10100	$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
100	$1 \times 10^2 + 0 \times 10^1 + 0 \times 10^0$	1100100	$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^2$
1000	1×10^3	1111101000	$1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3$

Sistemas de representación numérica

También podemos abreviar un poco:

Base 10	Descomposición	Base 2	Descomposición
0	0×1	0	0×1
1	1×1	1	1×1
2	2×1	10	$1 \times 2 + 0 \times 1$
3	3×1	11	$1 \times 2 + 1 \times 1$
4	4×1	100	$1 \times 4 + 0 \times 2 + 0 \times 1$
5	5×1	101	$1 \times 4 + 0 \times 2 + 1 \times 1$
6	6×1	110	$1 \times 4 + 1 \times 2 + 0 \times 1$
7	7×1	111	$1 \times 4 + 1 \times 2 + 1 \times 1$
8	8×1	1000	$1 \times 8 + 0 \times 4 + 0 \times 2 + 0 \times 1$
9	9×1	1001	$1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1$
10	$1 \times 10 + 0 \times 1$	1010	$1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$
11	$1 \times 10 + 1 \times 1$	1011	$1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
12	$1 \times 10 + 2 \times 1$	1100	$1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1$
13	$1 \times 10 + 3 \times 1$	1101	$1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$
14	$1 \times 10 + 4 \times 1$	1110	$1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$
15	$1 \times 10 + 5 \times 1$	1111	$1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1$
16	$1 \times 10 + 6 \times 1$	10000	$1 \times 16 + 0 \times 8 + 0 \times 4 + 0 \times 2 + 0 \times 1$
17	$1 \times 10 + 7 \times 1$	10001	$1 \times 16 + 0 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1$
18	$1 \times 10 + 8 \times 1$	10010	$1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$
19	$1 \times 10 + 9 \times 1$	10011	$1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
20	$2 \times 10 + 0 \times 1$	10100	$1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1$
100	$1 \times 100 + 0 \times 10 + 0 \times 1$	1100100	$1 \times 64 + 1 \times 32 + 1 \times 4$
1000	1×1000	1111101000	$1 \times 512 + 1 \times 256 + 1 \times 128 + 1 \times 64 + 1 \times 32 + 1 \times 8$

Conversión de binario a decimal

Tip: usaremos la notación $(x)_2$ o $(x)_{10}$ para indicar si la representación de x es binaria o decimal. Si no se usa nada, será por defecto decimal.

Suponiendo que la posición de más a la derecha es la posición 0, y que van aumentando a medida que nos movemos la izquierda, y que la última posición es la $n-1$, podemos convertir cualquier número binario de n bits a su equivalente decimal usando:

$$\sum_{k=0}^{n-1} s_k \times 2^k$$

donde s_k es el símbolo (bit) que se encuentra en la posición k.

Ejemplo: convertir $(100110101)_2$ a decimal:

$$1 \times 2^0 + 1 \times 2^2 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^8$$

$$= 1 + 4 + 16 + 32 + 256$$

$$= 309$$

Base 16: Hexadecimal

Si bien los computadores utilizan 0 y 1, muchas representaciones son más fáciles de manipular si se trabajan en grupos. La representación en base 16 se conoce como **hexadecimal** y requiere de 16 dígitos. Como no conocemos más que 10, agregamos las primeras 6 letras (*A*, *B*, *C*, *D*, *E*, *F*).

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010
19	13	10011
20	14	10100
42	2A	101010
100	64	1100100
1000	3E8	1111101000

Conversión de hexadecimal a binario

4 bit corresponden a un dígito hexadecimal

- HEX $\rightarrow$ BIN. Cada dígito hexadecimal se convierte a 4 bit, y se reemplazan posicionalmente.
- BIN $\rightarrow$ HEX. Agrupar (de derecha a izquierda) grupos de 4 bit y convertir cada grupo a una dígito hexadecimal. Si se necesitan más bit, agregar 0's a la izquierda.

Ejemplo:

- $(101010)_2 \rightarrow (101010)_2 \rightarrow (00101010) \rightarrow (2A)_{16}$
- $(2A)_{16} \rightarrow (00101010)_2 \rightarrow (00101010)_2$