



Universidad Nacional  
de Entre Ríos

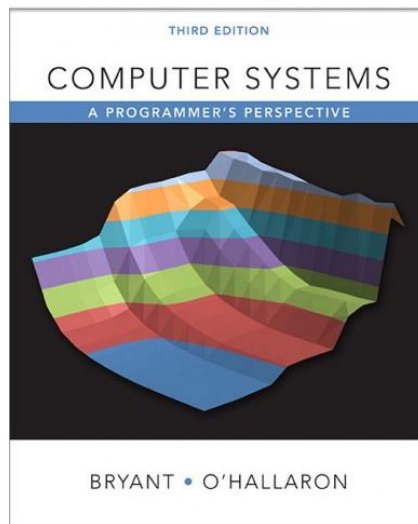
Tecnicatura universitaria en desarrollo web

# ¿Porque estudiar arquitectura?

Semana 1 – Arquitectura de computadoras

# Esta presentación esta basada en el libro de:

- ❑ Randal E. Bryant and David R. O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition, Pearson, 2016.



Archivos de presentación y ejemplos se alojan en:



<https://github.com/ruiz-jose/tudw-arq.git>

## ¿Porque estudiar arquitectura de computadoras?

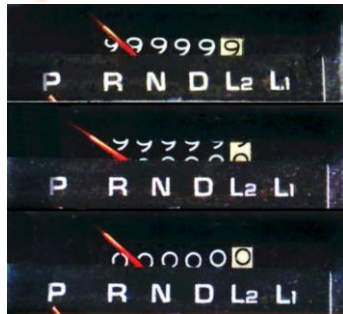
**Cuestiones o realidades de una computadoras que un desarrollador debe conocer**

- **Realidad #1:** Aritmética de la computadora
- **Realidad #2:** Conocer ensamblador
- **Realidad #3:** El acceso a la memoria importa

# Realidad #1: Aritmética de la computadora

**ints**  $\neq$  enteros

¿Qué pasa si el cuenta kilómetros del auto supera los 999999 km?



KM = 000000

No puede representar todos los números, es necesario realizar una abstracción de la realidad!!

**floats**  $\neq$  números reales

¿O si al contar ovejas se utiliza el tipo de datos enteros de una computadora?

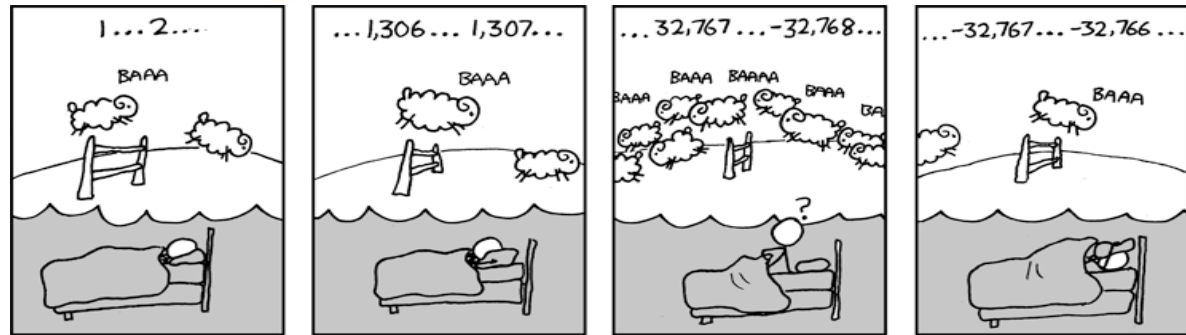


imagen: xkcd.com/571

$$32767 + 1 = -32768$$

## □ Ejemplo 01.c

En C los enteros corto signado (signed short int) ocupan 2 bytes y tienen un rango de  $\{-32768; 32767\}$

$$-2^{n-1} \leq \text{Rango} \leq 2^{n-1} - 1$$

# Realidad #1: Aritmética de la computadora

## ➤ ¿es $x * x \geq 0$ ?

- floats: sí!
- ints: ?

### ❑ Ejemplo 02.c

En C los enteros signado (int) ocupan 4 bytes y tienen un rango de  $\{-2.147.483.648; 2.147.483.647\}$

En clases de matemáticas, esta propiedad se cumple siempre!! **Cuando x es un numero entero o real**

¿Pero en una computadora?

En la computadora, esto depende del tipo de datos de x.

$40000 * 40000 \Rightarrow 1600000000$   
 $50000 * 50000 \Rightarrow ?? \quad -1794967296$

## ➤ ¿es $(x+y)+z = x+(y+z)$ ?

- ints sí! propiedad asociativa
- floats: ?

### ❑ Ejemplo 03.c

$(1E20 + -1E20) + 3,14 \Rightarrow 3,14$   
 $1E20 + (-1E20 + 3,14) \Rightarrow ?? \quad 0.00$

**¿Qué es válido en qué contexto?**  
**No se pueden asumir válidas todas las operaciones matemáticas por ser representaciones finitas.**

## Realidad #2: Conocer ensamblador

- Seguramente nunca escribas código ensamblador
  - Pero, entenderlo es clave para el modelo de ejecución de la máquina.
  - Los compiladores son mejores que vos, y más pacientes.
- ¿Qué multiplicación es más rápida  $x*2$  o  $x \ll 1$ ?

■ C:

```
x=5;  
x=x << 1;
```

Antes: 

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

 = 5

Después: 

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

 = 10

Desplazamiento lógico  
de 1 bit izquierda

■ Ensamblador:

```
mov ax,5  
shl ax,2 ;ax = 20
```

■ Ejemplo 04.c:

- Desplazar 1 bit a la izquierda multiplica un número por 2
- Desplazar n bits a la izquierda multiplica el operando por  $2^n$

## Realidad #2: Conocer ensamblador

$x * 973 \rightarrow \text{mul ax, 973}$

$x * 1024 \rightarrow \text{shl ax, 10} ; 2^{10} = 1024$

Realizar multiplicaciones con potencias de 2 ( $2^n$ ) utilizando desplazamiento de bits a la izquierda aumenta el rendimiento del programa.

➤ La división también es más rápida  $x/2$  o  $x \gg 1$ ?

■ C:

```
x=10;
x=x >> 1;
```

■ Ensamblador:

```
mov ax,10
shr ax,1 ;ax = 5
```

Antes: 0 0 0 0 1 0 1 0 = 10  
Después: 0 0 0 0 0 1 0 1 = 5

Desplazamiento lógico  
de 1 bit derecha

□ Ejemplo 05.c:

- Desplazar 1 bit a la derecha divide un número por 2
- Desplazar n bits a la derecha divide el operando por  $2^n$



## Realidad #3: El acceso a la memoria importa

➤ La memoria no es **infinita**

➤ bugs

☐ Ejemplo 06.c:

- Referencia a elementos de un arreglo fuera de rango
- Detección de destrucción de la pila (**\*\*\* stack smashing detected \*\*\*: terminated**)

➤ El desempeño de la memoria no es uniforme

## Realidad #3: El acceso a la memoria importa

- ❑ Ni siquiera contar la cantidad exacta de instrucciones predice el desempeño
- ❑ ¿Cómo se optimiza el código? Para poder optimizar, se tiene que entender el sistema.

```
void copyij(int src[2048][2048],  
            int dst[2048][2048])  
{  
    int i,j;  
    for (i = 0; i < 2048; i++)  
        for (j = 0; j < 2048; j++)  
            dst[i][j] = src[i][j];  
}
```

4.3ms

2.0 GHz Intel Core i7 Haswell

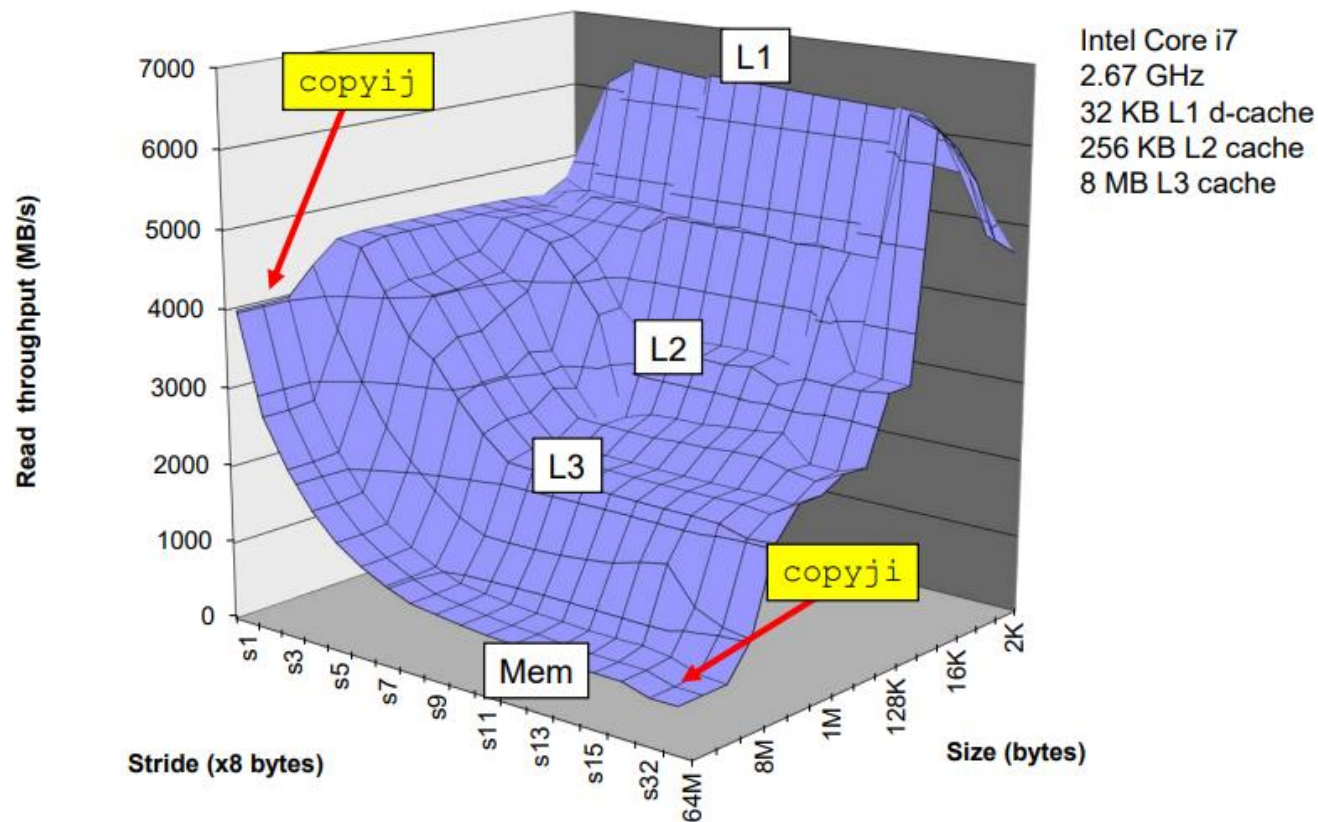
```
void copyji(int src[2048][2048],  
            int dst[2048][2048])  
{  
    int i,j;  
    for (j = 0; j < 2048; j++)  
        for (i = 0; i < 2048; i++)  
            dst[i][j] = src[i][j];  
}
```

81.8ms

Acceso por columnas o por  
filas.. ¿Porque?

- ❑ Organización de la jerarquía de memoria, el desempeño depende de los patrones de acceso a memoria.
- ❑ Los efectos de la caché y la memoria virtual afectan enormemente el desempeño de un programa. Adaptar un programa a las características de la memoria de un sistema puede llevar a grandes mejoras en velocidad.

## Realidad #3: El acceso a la memoria importa



# Preguntas?