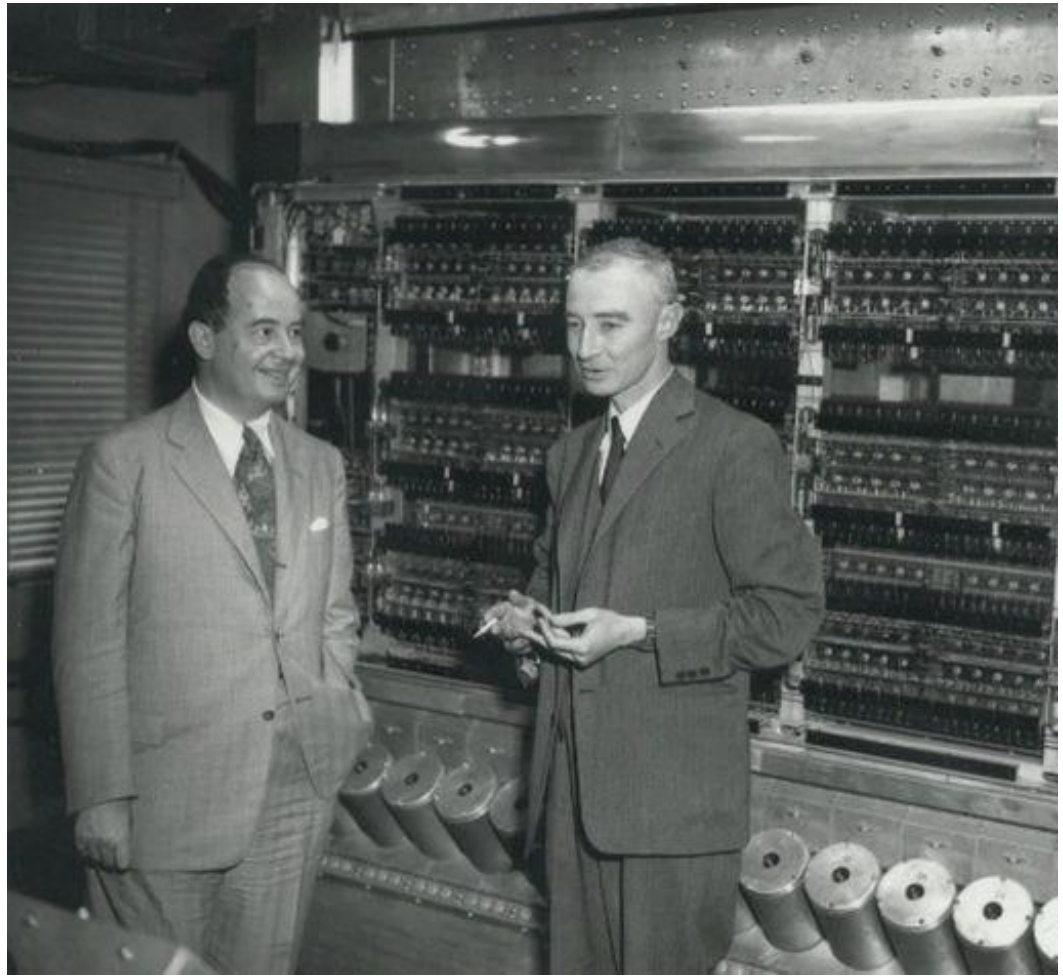
Tema 1

Repaso y definiciones

Aplicación, App, Programa

John Von Newman



- Propuso la adopción del bit
- desarrolló el concepto de los "bits de paridad“ para poder paliar la aparición de errores
- John von Neumann es uno de los responsables de sentar los pilares sobre los que se apoyan los computadores actuales

visión como para crear un modelo estándar que funcionase en cualquier ámbito o aplicación: **separar el software del hardware**, crear un modelo de computador universal.

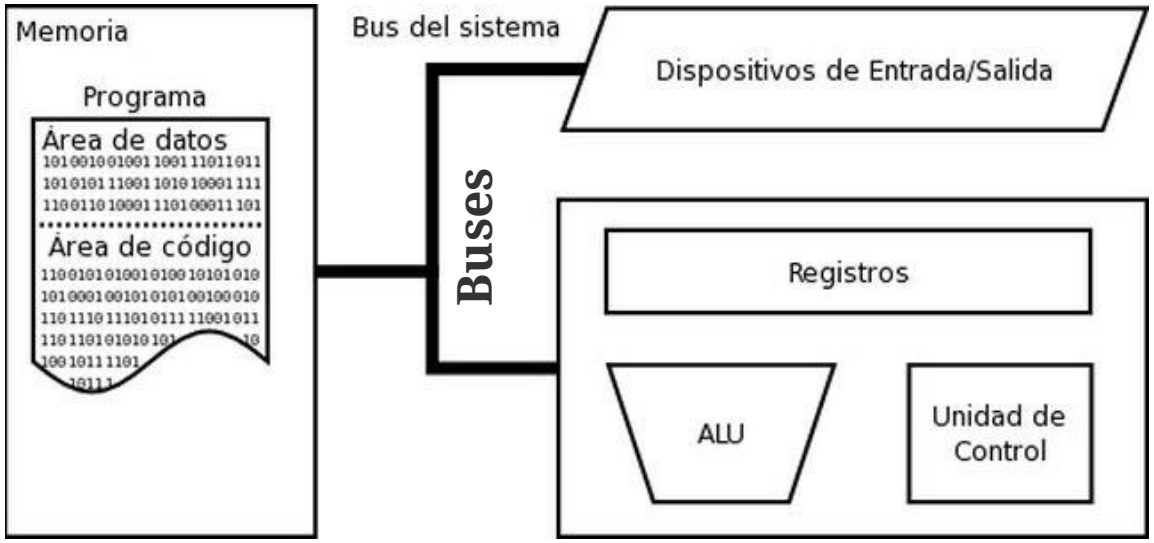
John Von Newman

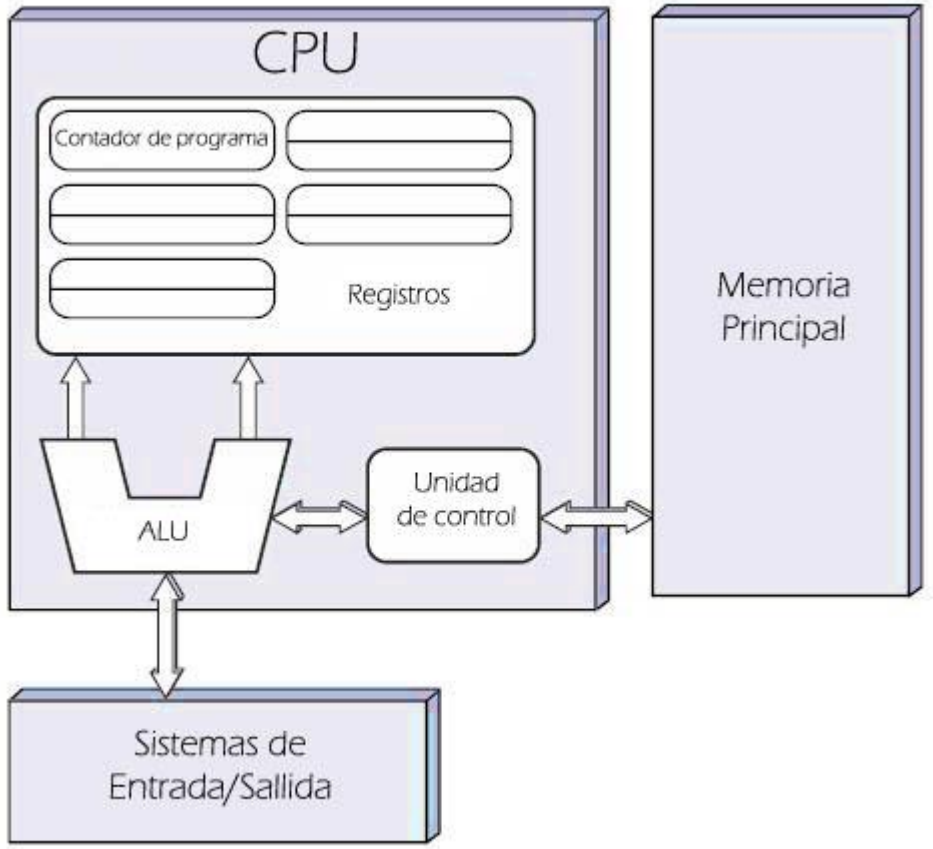
- **El bit** es la unidad mínima de información que se emplea en informática, este puede tener dos estados: uno o cero y comúnmente están asociados a que un dispositivo se encuentre apagado o encendido. Normalmente se utilizan para representar diferentes estados:
  - 0 y 1 | Apagado y encendido | Falso y verdadero| Abierto y cerrado

- El **byte** es una unidad de información en tecnología y telecomunicaciones compuesta por 8 bits.

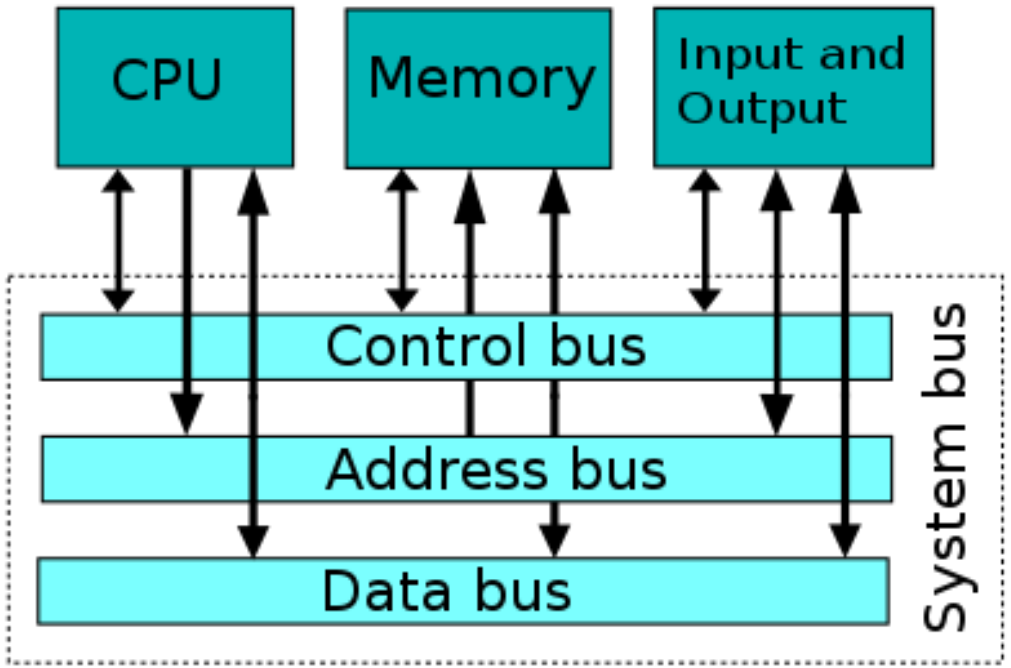
Medida	Equivalencia
1 byte	8 bits
1 Kilobyte	1024 Bytes
1 Megabyte	1024 Kilobytes
1 Gigabyt	1024 Megabytes
1 Terabyt	1024 Gigabytes

Arquitectura de Von Newman

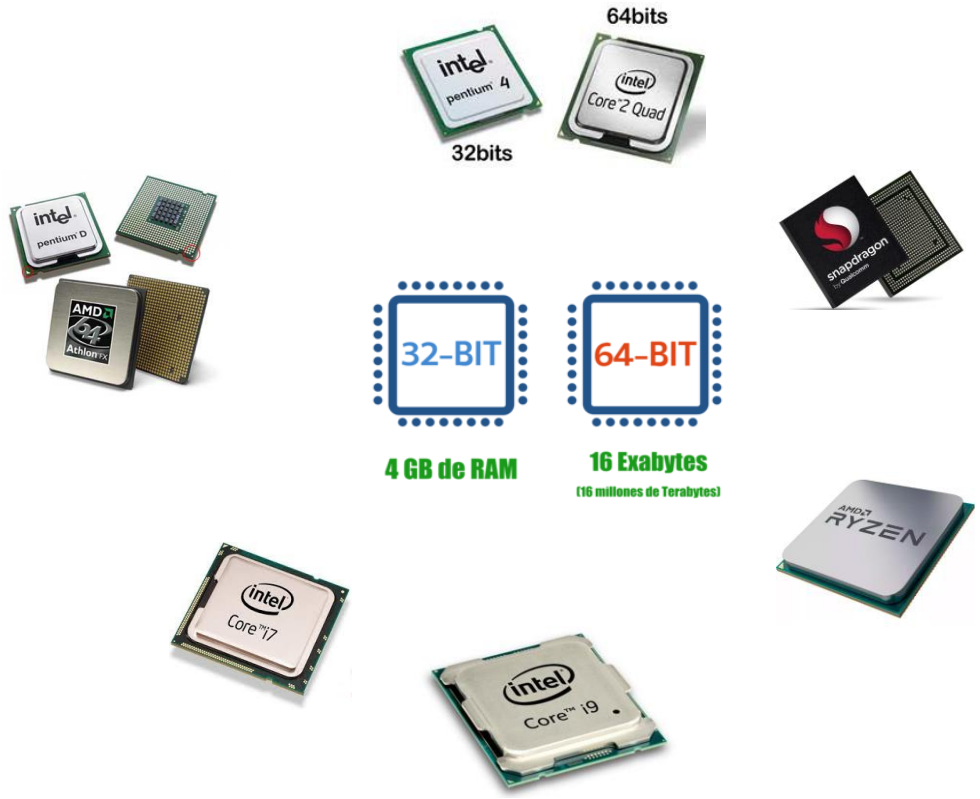


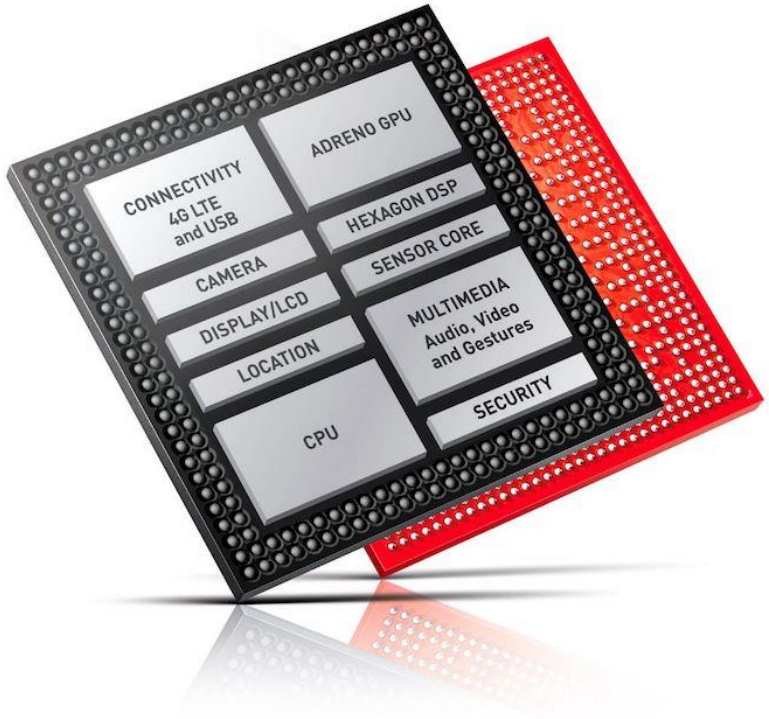


Arquitectura de Von Newman



Microprocesadores

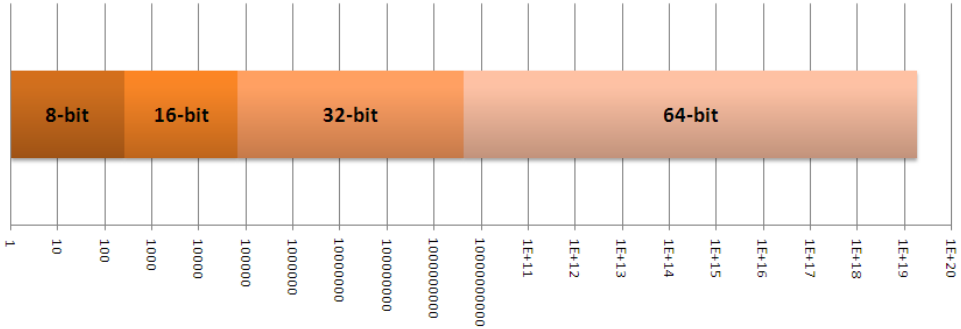




Capacidad de direccionar en RAM

4 GB de RAM

2^{32} direcciones o 4 gigabytes de RAM



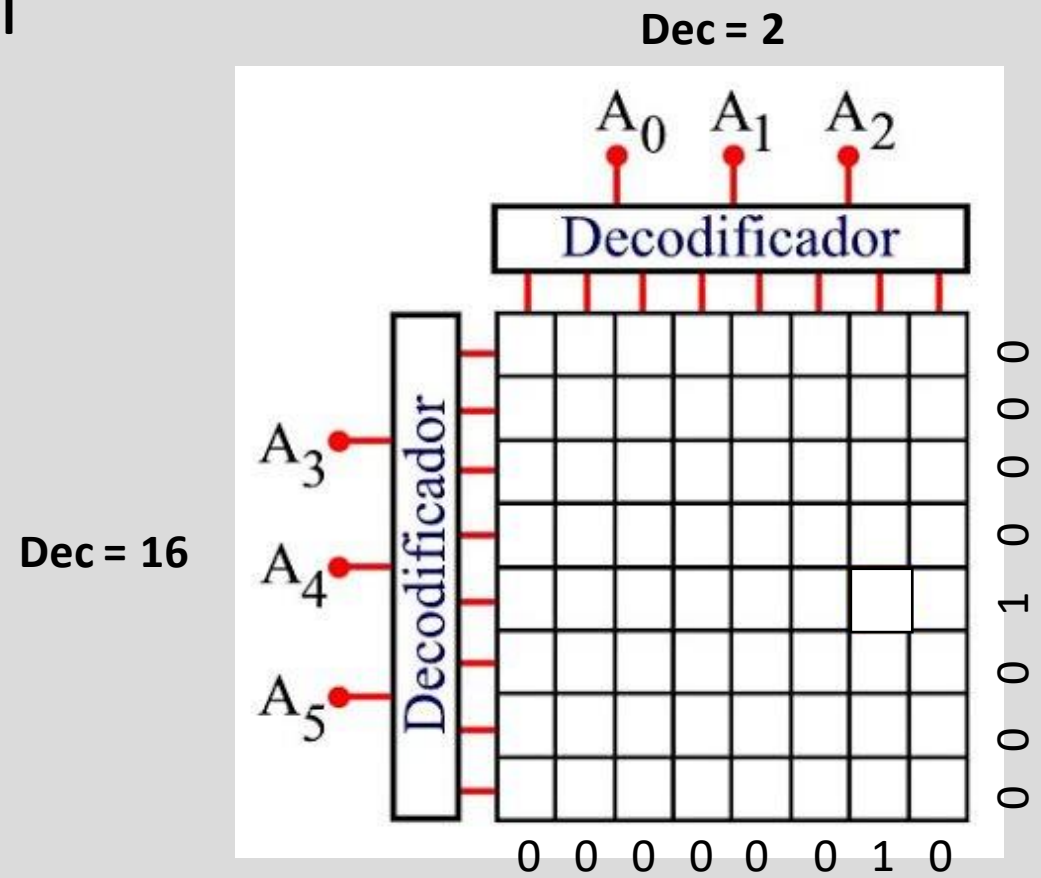
16 Exabytes

(16 millones de Terabytes)

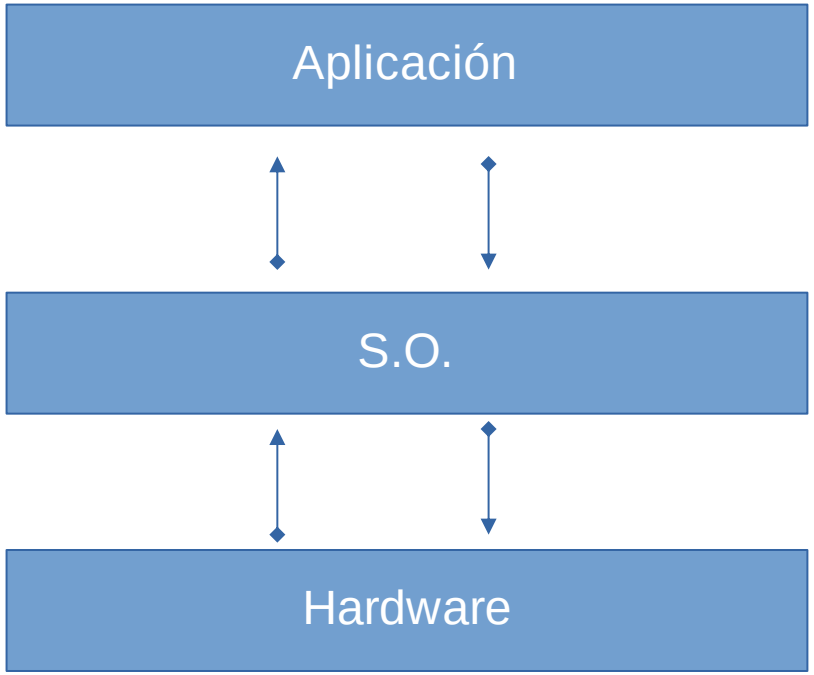
2^{64} direcciones o 16 Exabytes de RAM

Memoria dinámica

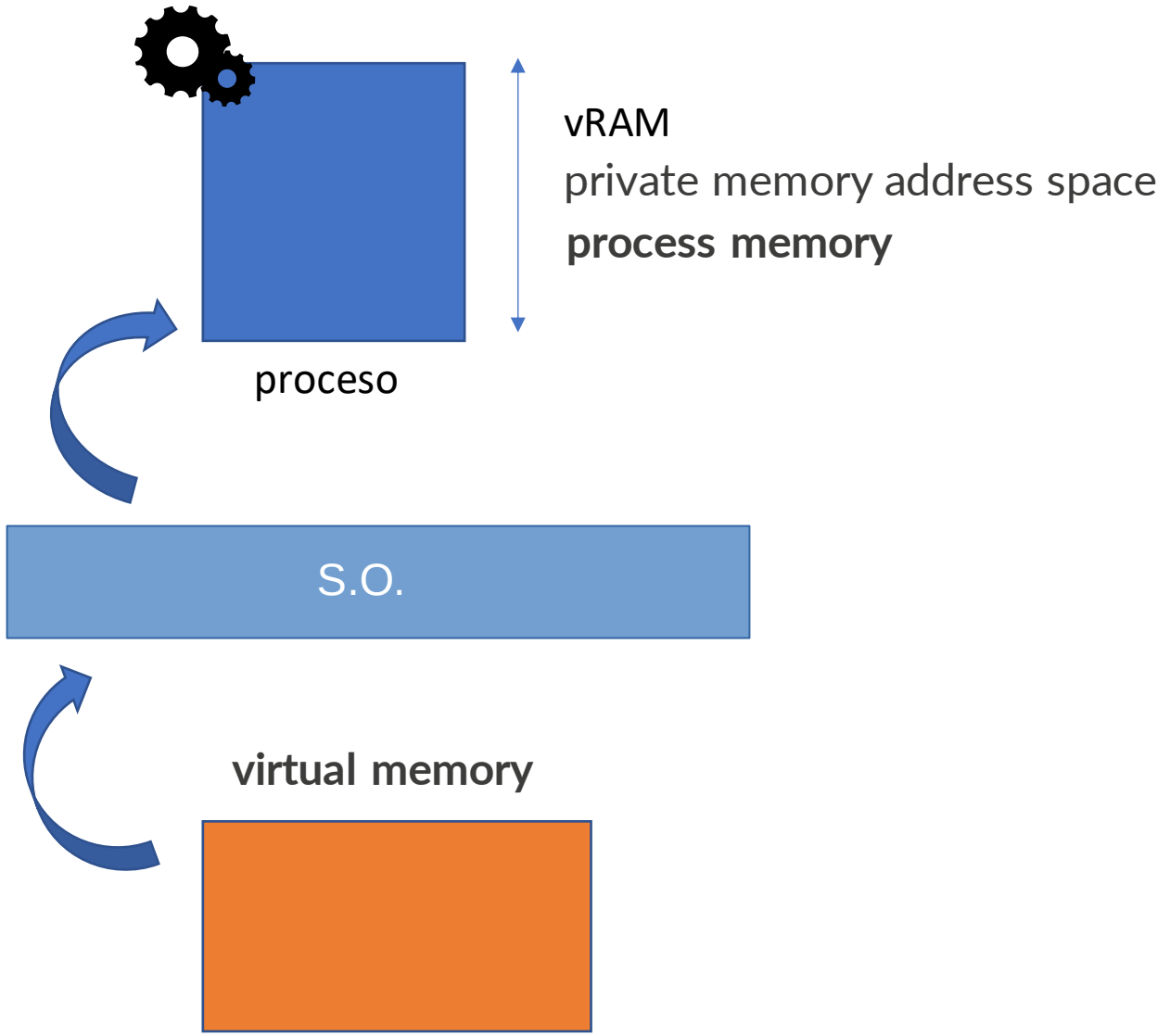
RAM



Aplicación y modelo de hardware



Aplicación y proceso



Cuándo programamos, necesitamos un medio de **grabar** o **retener** los datos. ¿Qué solemos usar en ese caso?

Variables

- Un programa necesita un medio de **grabar** o **retener** los datos que usa. Las variables y Constantes ofrecen varias maneras para representar y manipular los datos.

- Una variable es un **espacio** para guardar información

Variables

- La memoria RAM de la computadora puede ser vista como una serie de pequeñas casillas, cada una de las casillas esta numerada secuencialmente, este número que se le asigna representa su dirección de memoria y el nombre es la etiqueta de la caja.

Numero de Caja:
0000001

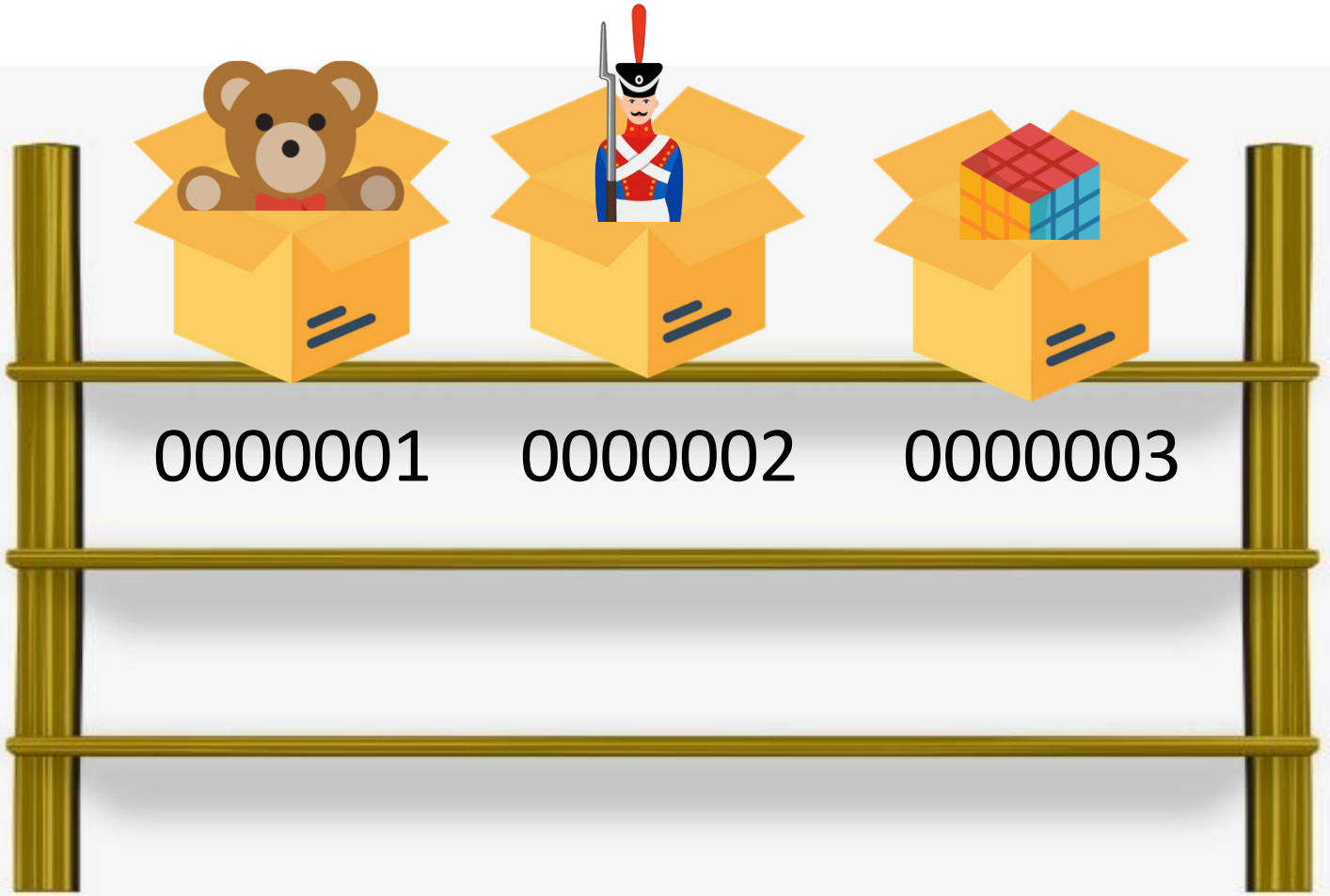
Contenido

Variables

Contenido:

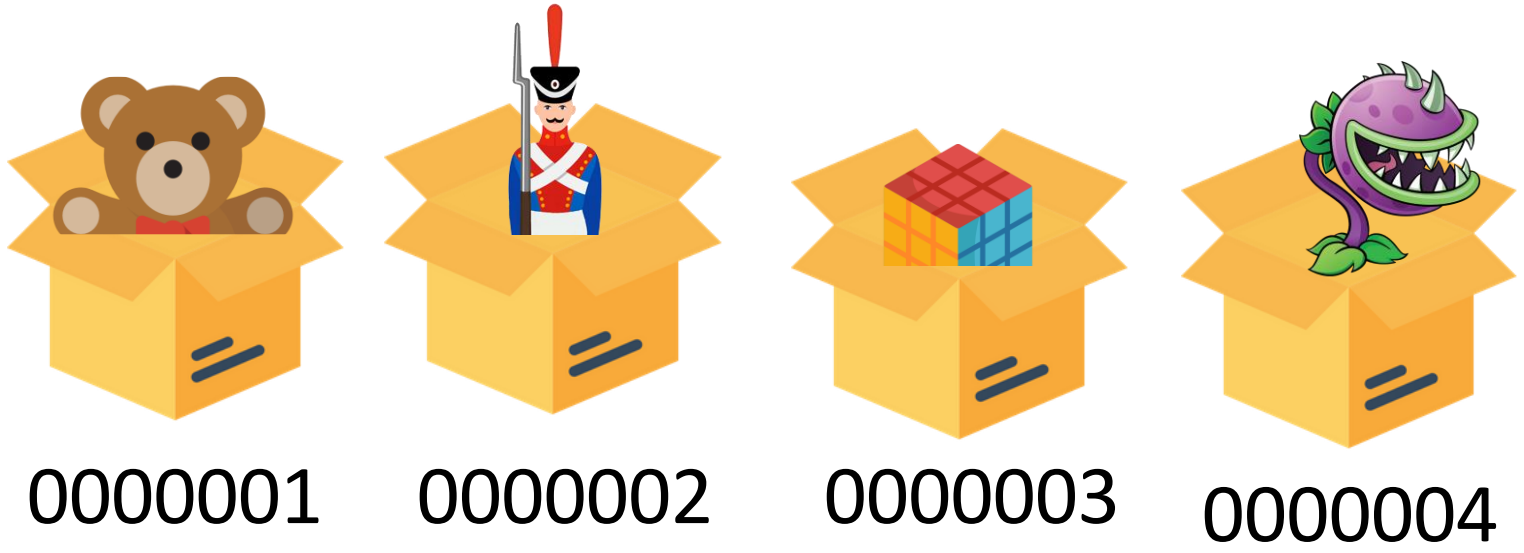
Numero de Caja:

Etiqueta



Variables

Ejercicio: Poner nombre a las variables



0000001 0000002 0000003 0000004

MEMORIA RAM

Dirección de memoria		0019FF48	0019FF50				
Etiqueta		Energía	E_Enemigo				
Contenido		100	100				

bytes

Tipos de datos

Tipo	Rango	Bytes	Printf ()
char	-128 ... 127 (ASCII)	1	%c
int	-32.768 ... 32.767	4	%d
long	-2.147.483.648 ... 2.147.483.647	8	%l
float	3.4*10^-38 ... 3.4*10^38	4	%f
double	1.7*10^-308 ... 1.7*10^308	8	%i
void	Valor nulo	nulo	-

https://www.tutorialspoint.com/c_standard_library/c_function_printf.htm

Operador sizeof

C proporciona un operador unario a tiempo de compilación llamado sizeof que se puede emplear para calcular el tamaño en bytes de cualquier Tipo de de dato

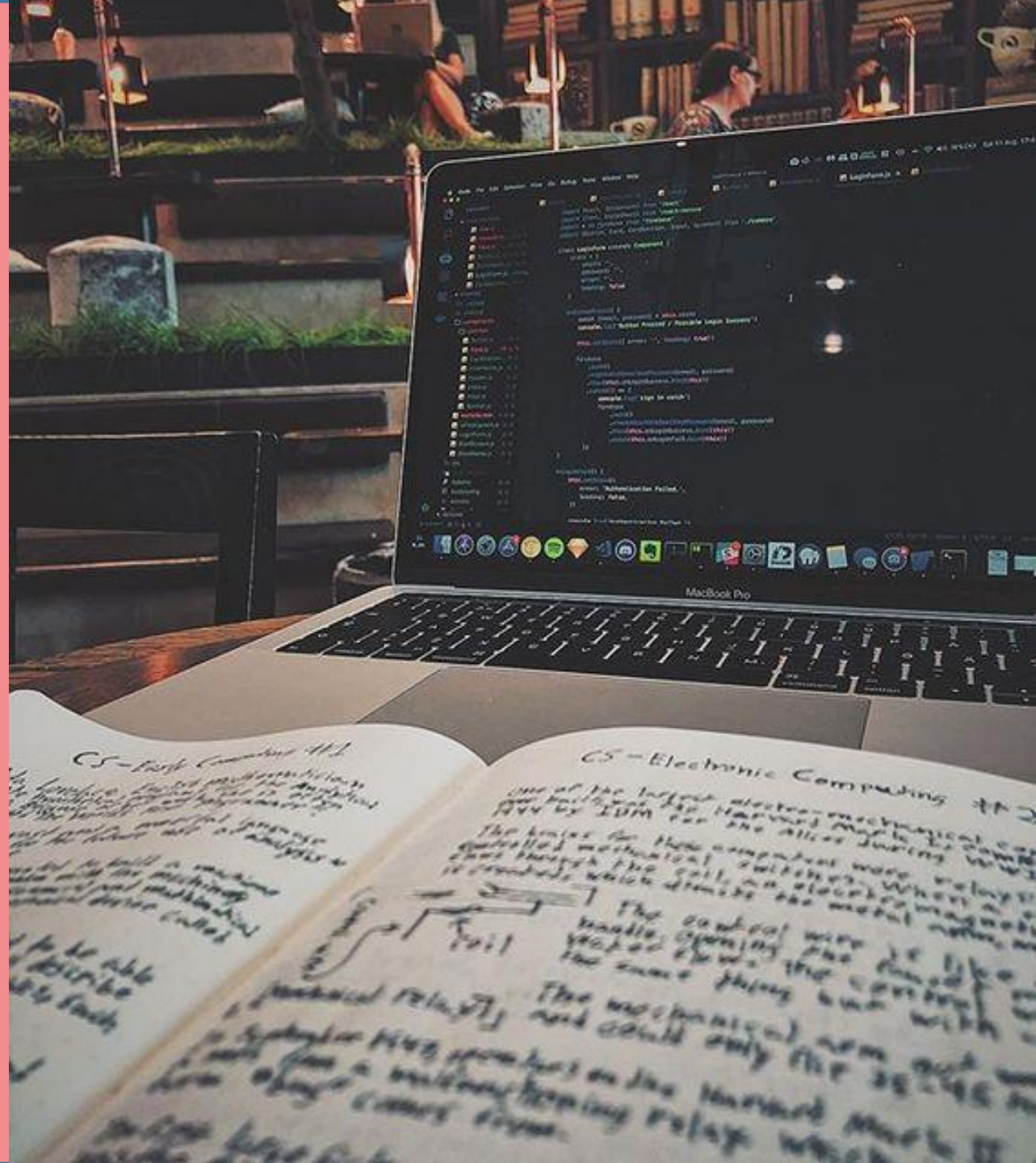
- sizeof (objeto)– Objeto puede ser variable, array, estructura

int  sizeof (*Tipo*)

Caso de estudio en código

Tema 2

Punteros



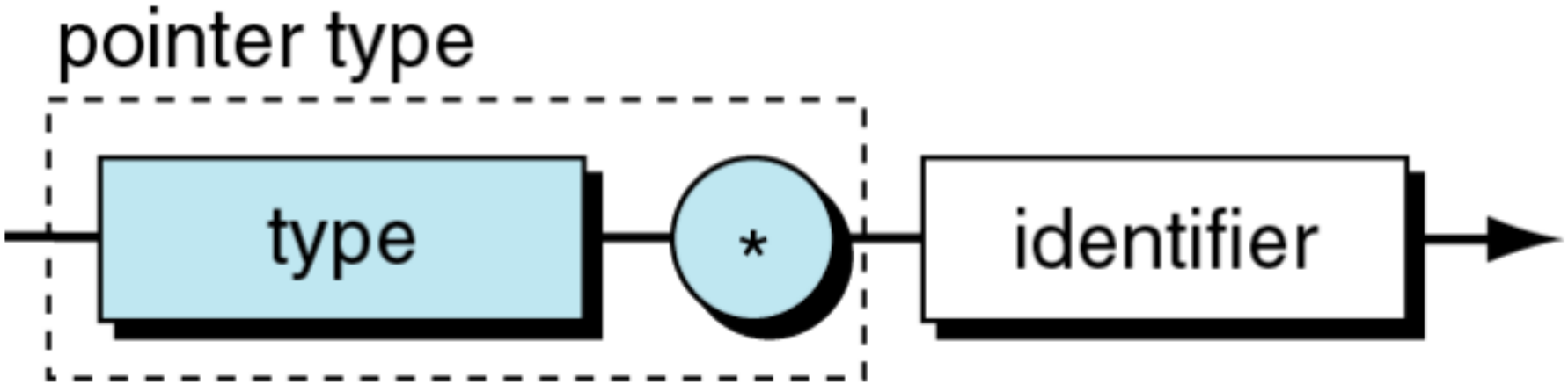
Punteros

- Un puntero es una variable que guarda la dirección de memoria de una variable

- Un puntero *referencia* a una ubicación en memoria, y a la obtención del valor almacenado en esa ubicación se la conoce como *desreferenciación* del puntero.

Punteros

Las variables puntero se declaran usando el operador indirección *, llamado también operador dereferencia.

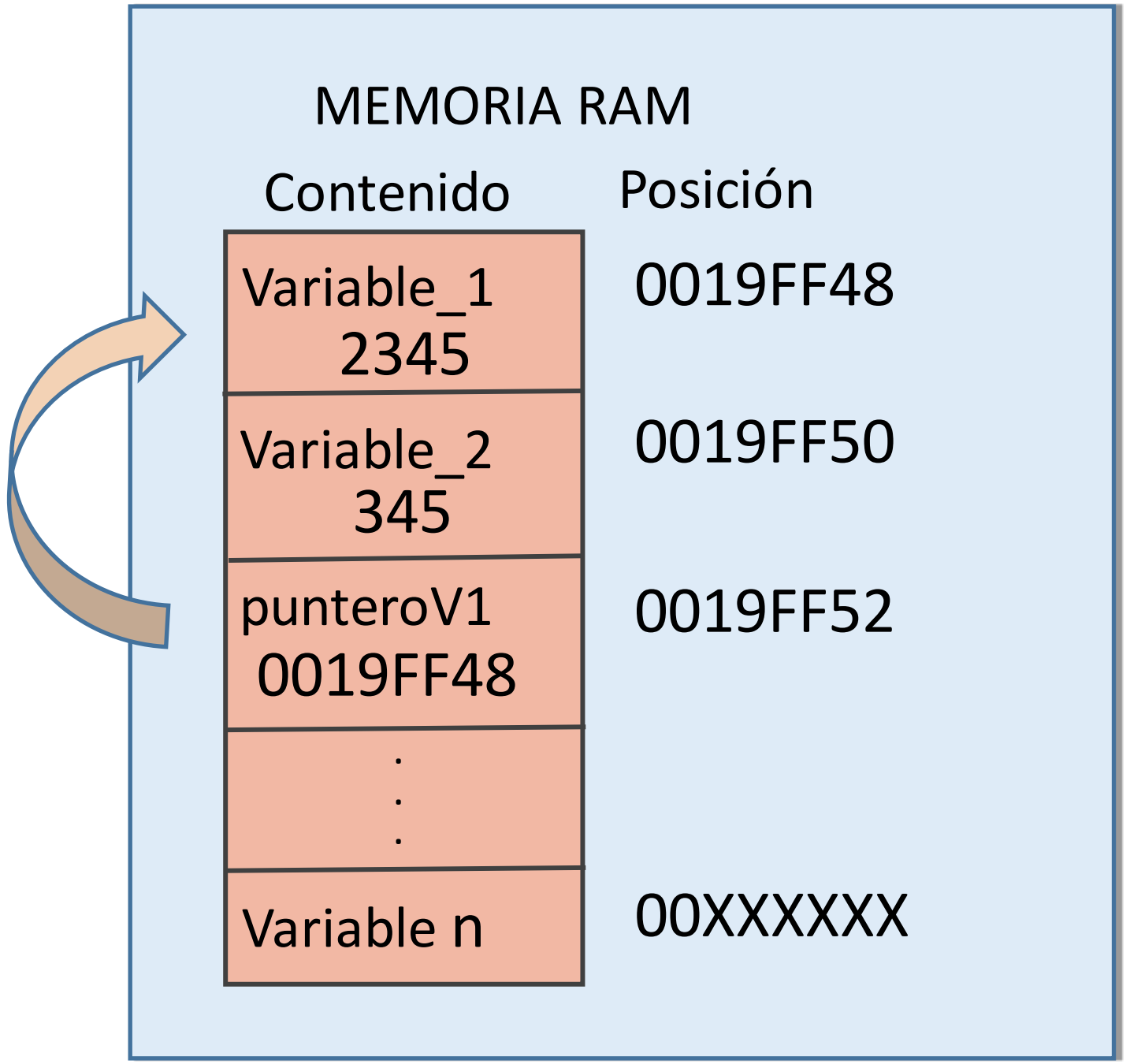


Punteros

- El principal uso de un puntero en lenguaje C está vinculada a la asignación y uso de Memoria dinámica.
  [Contenidos que se ven en la materia más adelante].

- Pasaje por referencia de variables a una función.

¿Que es un puntero?

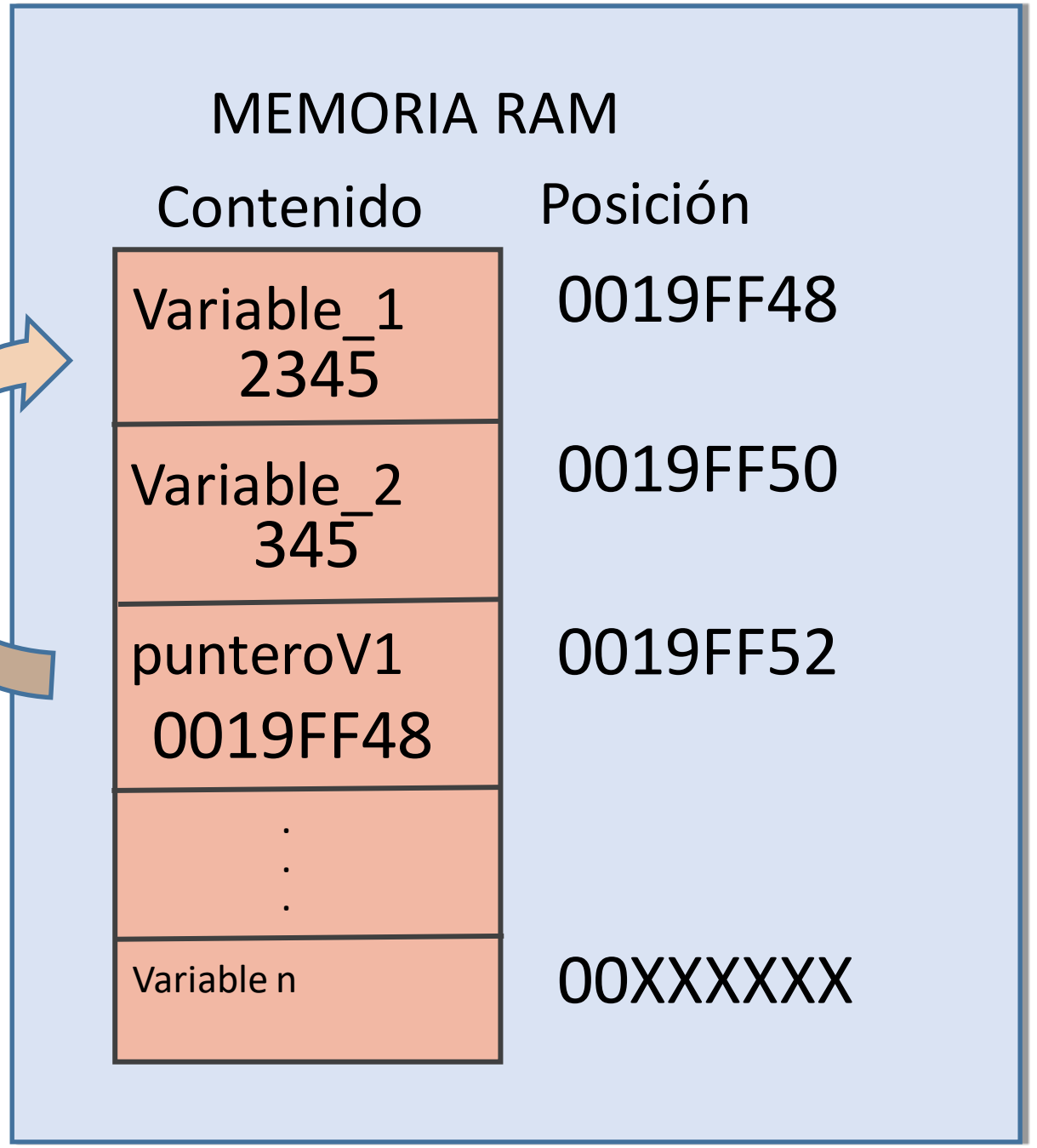


Declarando un puntero

```
int Variable_1 = 2345
```

```
int * punteroV1;
```

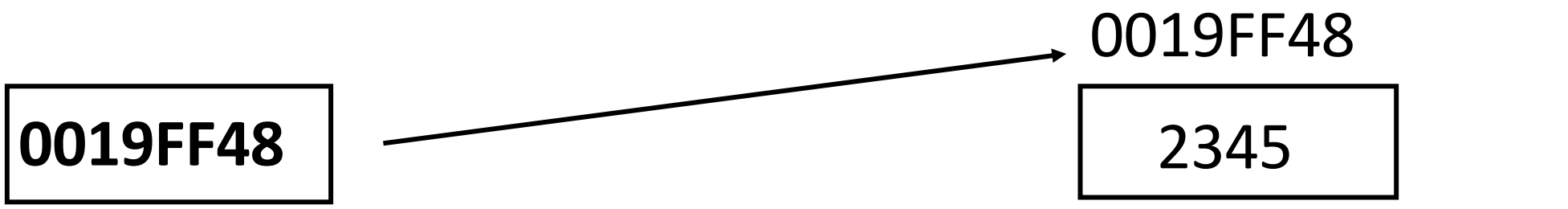
```
punteroV1 = &Variable_1;
```



¿Cómo usar un puntero?

</> Código

```
int variable1 = 2345;
int * punteroV1  = 0019FF48;
```



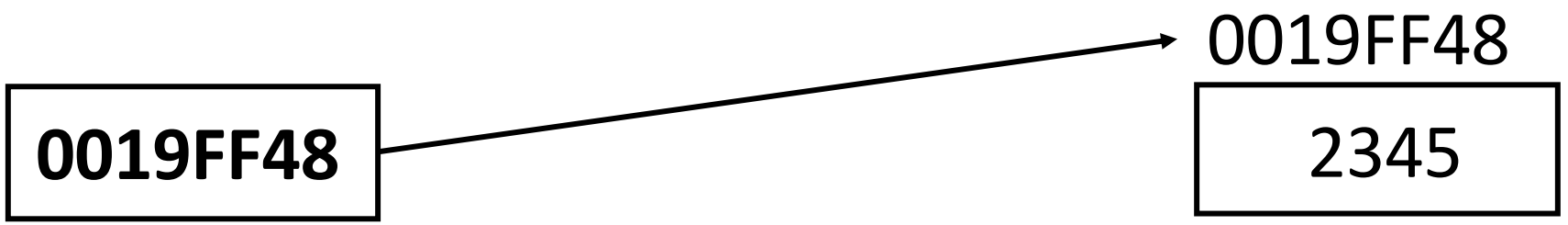

```
int * punteroV1 = 0xffffdbc3;
```

```
Int variable1 = 2345;
```

¿Cómo usar un puntero?

</> Código

```
int variable1  = 2345;
int * punteroV1  = &Variable1;
```



int * punteroV1 **= &Variable;**

Int variable1 **=** 2345**;**

Punteros – Operadores

Declaración de un puntero

<tipo> *<identificador>

Operadores

El operador & ← devuelve una **dirección** del **mismo tipo del operando**

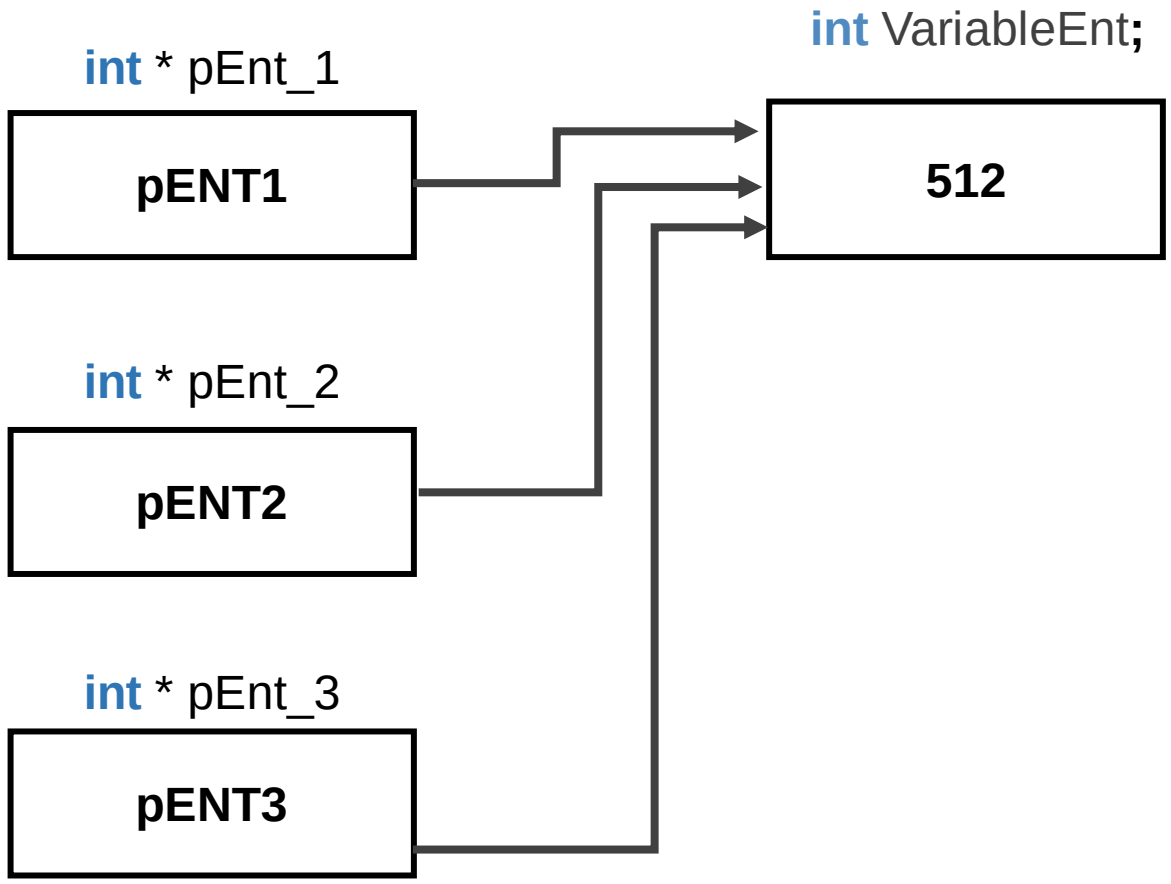
El operador * ← devuelve el **contenido** de lo apuntado por un puntero.

Parámetro en printf ()

El operador %p ← Permite mostrar la **dirección** de **memoria apuntada**.

Caso de estudio en código

Ejercicio práctico



ACTIVIDAD

- Declarar una variable y asigne un valor
- Declarar tres punteros que apunten a la misma variable antes declarada
- Demostrar que la dirección apuntada por cada puntero es la misma con más de una notación.
- Mostrar por pantalla la posición de memoria de cada puntero y el valor contenido.

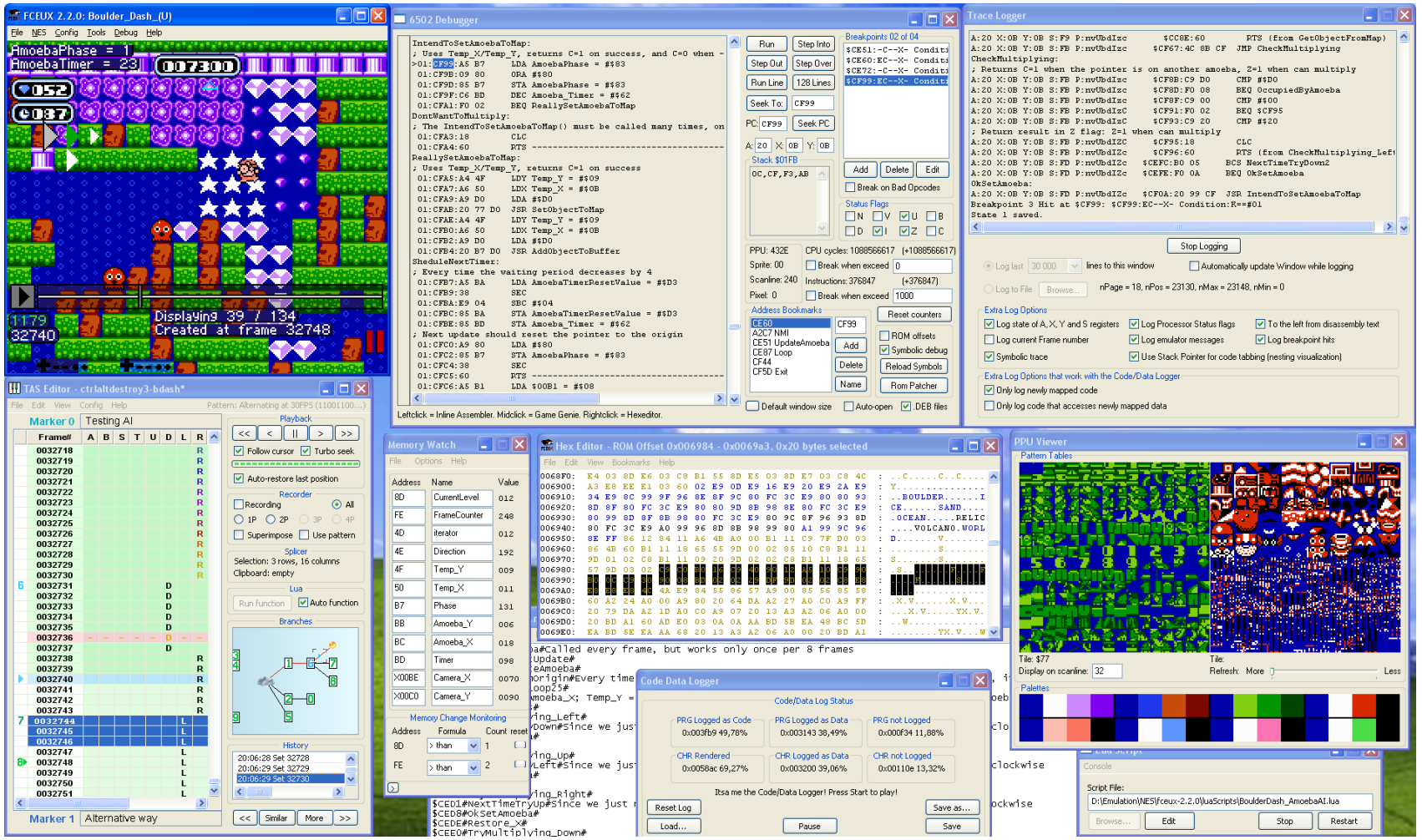
Anecdótico…


GAME
GENIE
VIDEO GAME ENHANCER
GENESIS
galoob


A B C D E F G H J K
L M N P R S T V W X Y Z
0 1 2 3 4 5 6 7 8 9
60.8

Anecdótico…

http://www.fceux.com/web/home.html



Repositorios interesantes en lenguaje c

https://github.com/topics/c

- Repositorio de git
- El repositorio del lenguaje Ruby
- El Kernel de Linux

Entre otros..

Breve historia del lenguaje c

Escrito por **Dennis Ritchie**

Algunas características que nos interesa

- Es un lenguaje muy flexible que permite programar con múltiples paradigmas.
- Acceso a memoria de bajo nivel mediante el uso de **punteros**.
- Un conjunto reducido de palabras clave.
- Código portable por medio de las **biblioteca estándar de C**