Arquitectura de Ordenadores

Representación de Datos en Memoria

Abelardo Pardo
abel@it.uc3m.es

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Definición de Memoria

- La memoria es un **dispositivo electrónico** en el que se almacenan datos.
- La memoria permite operaciones de **escritura** en las que los datos se almacenan en un lugar dado.
- La memoria también permite operaciones de **lectura** en las que dado un lugar, se obtiene el contenido previamente almacenado en dicho lugar.
- **Otra posible definición:** La memoria es un circuito sobre el que se permiten dos operaciones:

1. **Data Read(Address a)**: Dada una posición o dirección de memoria se obtiene un dato.
2. **void Write(Addres a, Data d)**: Se almacena un dato dado en una posición o dirección de memoria (no se devuelve resultado).

Estructura Interna de la Memoria

- La memoria consta de **celdas capaces de almacenar un bit**.
- Cada celda **necesita voltaje** para mantener la información almacenada.
- A este tipo de memoria se le conoce como **memoria RAM (Random Access Memory)**
- Existen **dos tipos** de memoria RAM

1. **Estática (SRAM):** Almacena la información en las celdas indefinidamente mientras se mantenga el voltaje (utiliza un biestable).
2. **Dinámica (DRAM):** La información almacenada se borra tras un tiempo entre 10 y 100 milisegundos (utiliza un condensador). Para evitar perder información se debe reescribir **cada bit de información** antes de perderlo. Es **10 veces** más lenta, **6 veces** más compacta y **100** veces más barata.

Propiedades de la Memoria

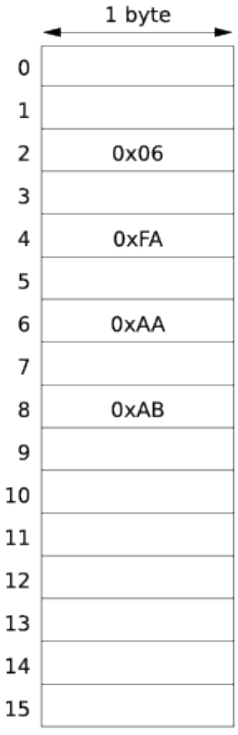
- Tabla en la que se almacenan los datos en **formato binario**.
- **Definición:** La **unidad mínima de direccionamiento** se define como la cantidad de información que se almacena en una posición de la memoria a la que se refiere una dirección. Es decir, los datos almacenados entre **dir** y **dir+1**.
- Generalmente, esta unidad mínima de direccionamiento es **un byte** (8 bits).
- En adelante se asumirá 1 byte **a no ser que se indique lo contrario**.
- Cada una de las posiciones tiene una **dirección única**.
- Cada **unidad mínima de direccionamiento** tiene una **dirección única**.
- La **primera unidad** de información tiene la dirección 0.
- Una memoria que almacena **1024 bytes** tiene direcciones desde la **0** hasta la **1023**

Tamaño de Memoria

- El tamaño de memoria se mide por la **cantidad de información** que es capaz de almacenar.
- Un kilobyte son **1024 bytes** (2^{10} bytes)
- Un megabyte son **1024 kilobytes** (2^{20} bytes)
- Un gigabyte son **1024 megabytes** (2^{30} bytes)
- Un terabyte son **1024 gigabytes** (2^{40} bytes)
- **Ejemplo:** Un ordenador con **512 megabytes de memoria** puede almacenar $512 * 2^{20}$ bytes de información.
- ¿Por qué la memoria se suele medir siempre en tamaños **potencias de 2**?

Operaciones sobre Memoria

- Se ejecutan las siguientes operaciones sobre memoria:
  - Write(2, 0x06)
  - Write(4, 0xFA)
  - Write(6, 0xC6)
  - Write(8, 0xAB)
  - Write(6, 0xAA)
- El contenido **inicial de la memoria es indefinido**.
- ¿Cuál es el resultado de la operación **Read(6)**



Escritura(2, 0x06)
Escritura(4, 0xFA)
Escritura(6, 0xC6)
Escritura(8, 0xAB)
Escritura(6, 0xAA)
Lectura(6) = 0xAA

Parámetros de las Operaciones de Memoria

- Toda información debe **estar codificada en Binario**.
- Tanto los datos como la dirección deben estar **codificados en binario**.
- En la operación **Write(2, 0x06)**:
  - ¿cuántos bits se precisan para codificar **el dato**?
  - ¿cuántos bits se precisan para codificar **la dirección**? Depende del tamaño
- Al conjunto de direcciones posibles para acceder a la memoria se le denomina **Espacio de Direcciones**
- ¿Cuántos elementos tiene el **espacio de direcciones** de una memoria de **1 Megabyte**?
- El número de bits utilizado para codificar el espacio de direcciones **depende** del número de unidades mínimas de direccionamiento y del tamaño de la memoria.

Tamaño de Memoria y Tamaño de la Dirección de Memoria

- El tamaño de la memoria es **el número de unidades de información** que puede almacenar.
- Cada una de estas unidades de memoria tiene una **dirección**.
- El número de direcciones de memoria determina el **tamaño** de la dirección de memoria.
- Estos dos conceptos están **relacionados** mediante la fórmula:

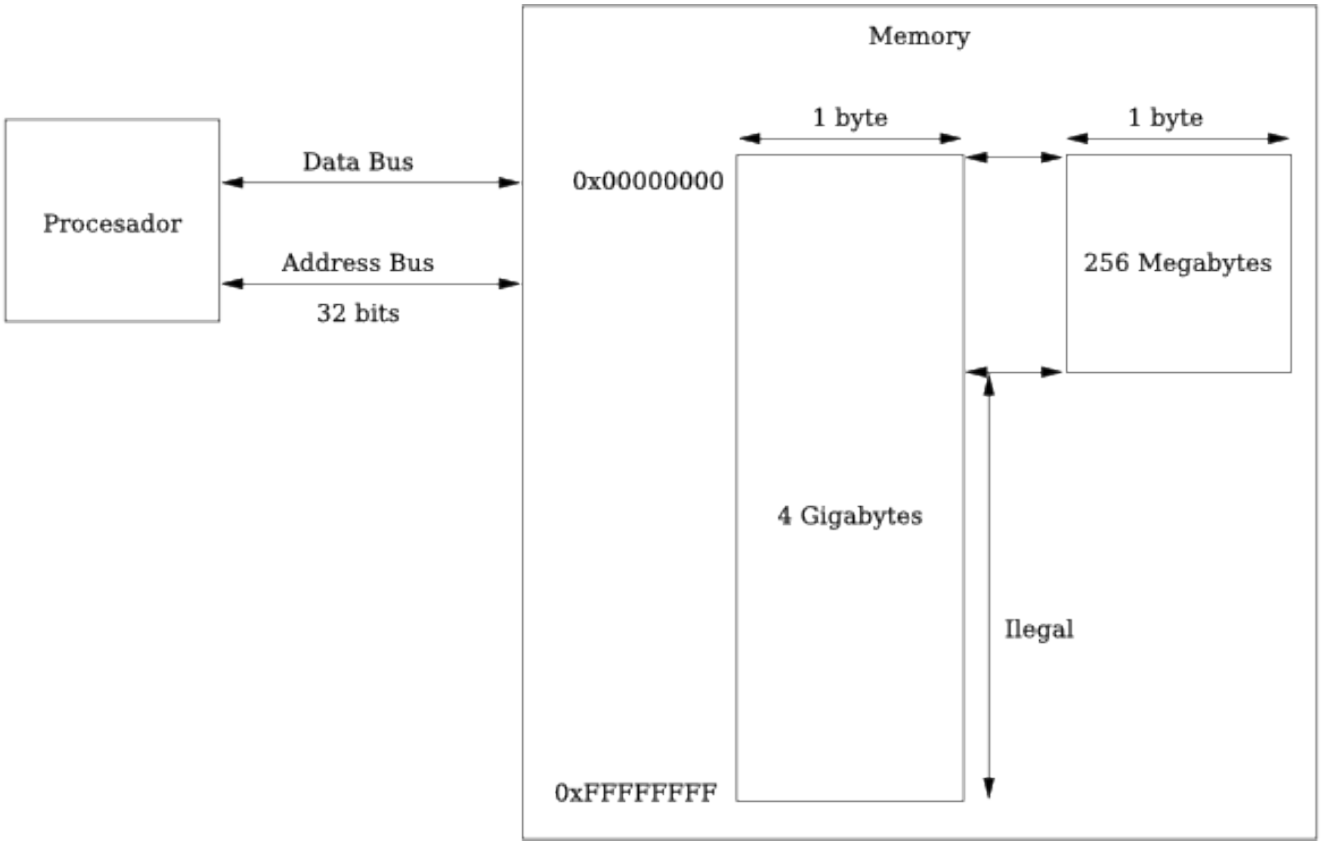
$$\text{tamaño de memoria} = 2^{\text{tamaño de dirección}}$$

- **Ejemplo:** Una memoria cuya dirección se representa con **20 bits** tiene un tamaño de **1 Megabyte**.
- **Ejemplo:** En una memoria de 64 kilobytes, una posible operación de escritura sería **Write(0xC8A4, 0x55)**.

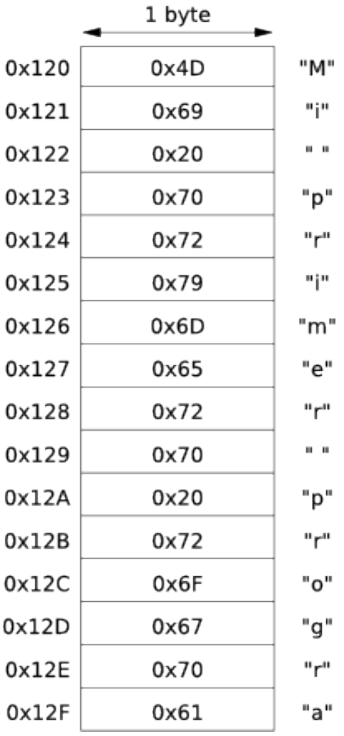
Memoria Posible y Memoria Real de un Ordenador

- La memoria es un módulo **externo al procesador**.
- El procesador asume que la memoria tiene un **determinado tamaño de dirección**.
- Es posible que la memoria **tenga un tamaño más reducido** que el que puede manipular el procesador.
- **Ejemplo:** El procesador Pentium tiene un tamaño de dirección de memoria de **32 bits**.
- ¿Cuál es el **tamaño de memoria** que corresponde a este tamaño de dirección?
- ¿Cuál es el tamaño de memoria que **suelen tener** los ordenadores?
- Existen direcciones de memoria **que nunca son utilizadas por el procesador**.

Memoria Real



Almacenamiento de Caracteres en ASCII



- Utilizando la codificación **ASCII** cada símbolo se representa como **1 byte**.
- Un string es una **secuencia de caracteres** y por tanto se almacena en memoria **en posiciones consecutivas**.
- **Ejemplo:** `start: push $msg`
- La **posición de memoria en donde se almacena un string** es la posición de su **primer elemento**.
- ¿Cómo se sabe la **longitud** de un string?

Tamaño de Datos en Java

- Cada lenguaje de programación **debe definir los tamaños de los tipos de datos primitivos**.
- Los tamaños de los tipos de datos básicos en **JAVA** son:

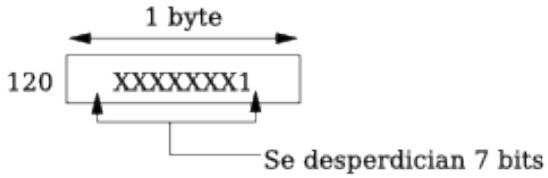
Tipo	Contiene	Tamaño	Rango
boolean	true, false	1 bit	
byte	Entero	8 bits	-128 a 127
char	Caracter Unicode	16 bits	0 a 65535
short	Entero	16 bits	-32768 a 32767
int	Entero	32 bits	-2147483648 a 2147483647
long	Entero	64 bits	-9223372036854775808 a 9223372036854775807
float	IEEE-754 Coma Flotante	32 bits	1.4E-45 a 3.4028235E+38
double	IEEE-754 Coma Flotante	64 bits	4.9E-324 a 1.7976931348623157E+308

- ¿Cómo se almacena **un booleano** en una memoria con bytes como **unidad mínima de direccionamiento**?
- ¿Cómo se almacena **un long**?

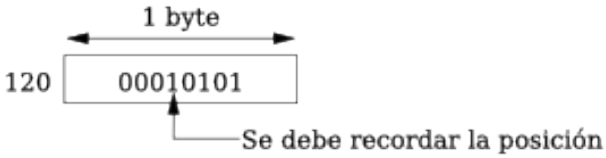
Almacenamiento de un Booleano

- Un booleano codifica **dos valores** y por tanto **sólo requiere un bit**.
- Para almacenar un booleano en memoria existen dos opciones:

1. Utilizar **1 byte** sin utilizar 7 de los 8 bits.

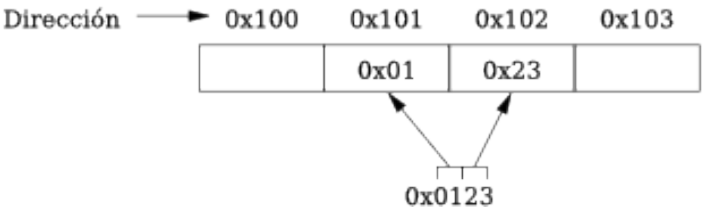


2. Utilizar **un único bit** dentro de 1 byte (caben 8 booleanos en un byte).

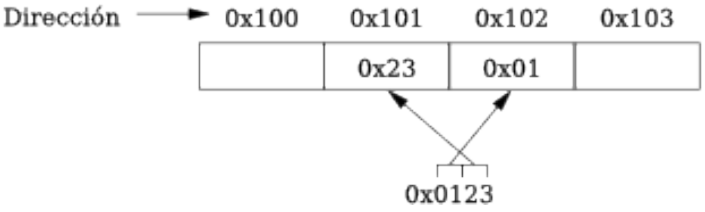


Almacenamiento de Números Enteros

- **Ejemplo:** ¿Cómo se almacena en memoria el **entero** de 16 bits **0x0123**?
- Se requieren **dos bytes consecutivos**. ¿Cuál de los dos bytes se **almacena primero**?
  - **Big Endian:** Se almacena comenzando por el de **más peso**.



  - **Little Endian:** Se almacena comenzando por el de **menos peso**.



- Esta diferencia es **irrelevante** excepto si los datos se intercambian entre **procesadores** con diferentes políticas.
- La **dirección del entero** es la **dirección de su primer byte** almacenado en memoria.
- El procesador **Pentium** almacena los enteros en formato **Little Endian**.

Almacenamiento de Diferentes Enteros

- Para el caso de enteros de **32** y **64** bits se sigue la misma política.
- **Ejemplo de almacenamiento de int:** Write(0x100, (int) 0x89A8C6A4)

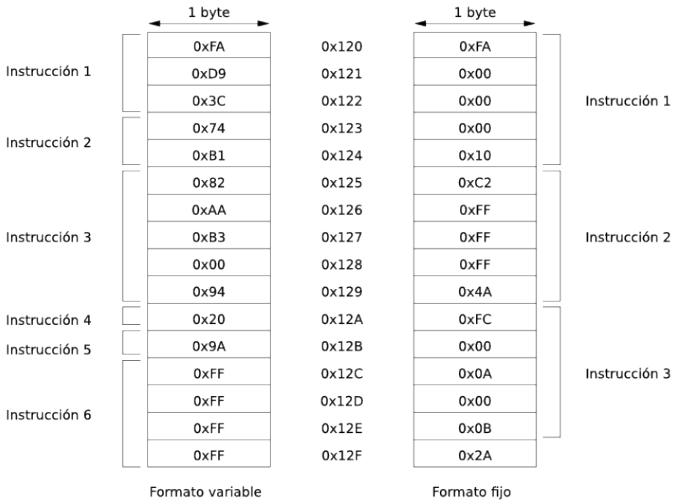
Little Endian		0xA4	0xC6	0xA8	0x89	
Dirección →	0xFF	0x100	0x101	0x102	0x103	0x104
Big Endian		0x89	0xA8	0xC6	0xA4	

- **Ejemplo de almacenamiento de long:** Write(0x100, (long) 0xFA45D3B333CA112F)

Little Endian		0x2F	0x11	0xCA	0x33	0xB3	0xD3	0x45	0xFA	
Dirección →	0xFF	0x100	0x101	0x102	0x103	0x104	0x105	0x106	0x107	0x108
Big Endian		0xFA	0x45	0xD3	0xB3	0x33	0xCA	0x11	0x2F	

Almacenamiento de Instrucciones

- Como **toda la información** que manipula el procesador está codificada en binario, es susceptible de ser almacenada en memoria.
- **Ejemplo:** El conjunto de símbolos que representan **operaciones entre 2 enteros de 8 bits**.



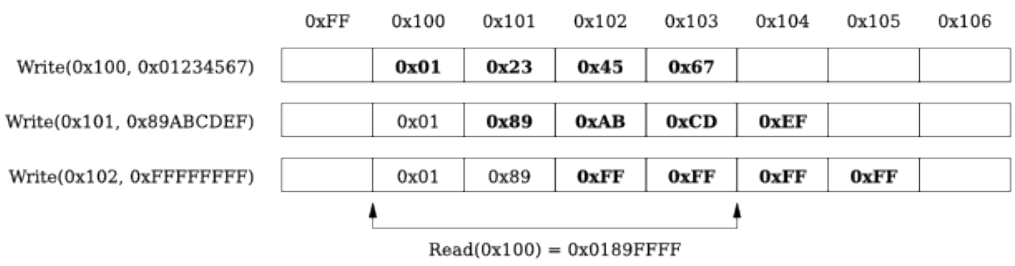
- Para instrucciones **no es preciso** distinguir entre los estilos **big endian** o **little endian** pues los bytes **no existen bytes más significativos que otros**.
- Los bytes se almacenan **en el orden en el que se escriben**.

Tamaño de los Datos en las Operaciones de Lectura/Escritura

- La mayoría de datos que se almacenan en memoria **requieren más de un byte**.
- La **unidad mínima de direccionamiento** es un byte.
- Para obtener **mejor rendimiento** en la transferencia de datos, **la unidad de transferencia** suele ser **mayor que un byte**.
- Si la **unidad de transferencia** es n bytes la operación de lectura devuelve n bytes consecutivos, y la de escritura modifica n bytes consecutivos.
- **Ejemplo:** La **Unidad de transferencia** es de 4 bytes.
  - **Read(0x100)**: Devuelve los cuatro bytes almacenados en las posiciones 0x100 a 0x103.
  - **Write(0x100, 0xAB132432)**: Almacena a partir de la posición 0x100 los cuatro bytes 0xAB132432.

Ejemplo de Operaciones en Memoria

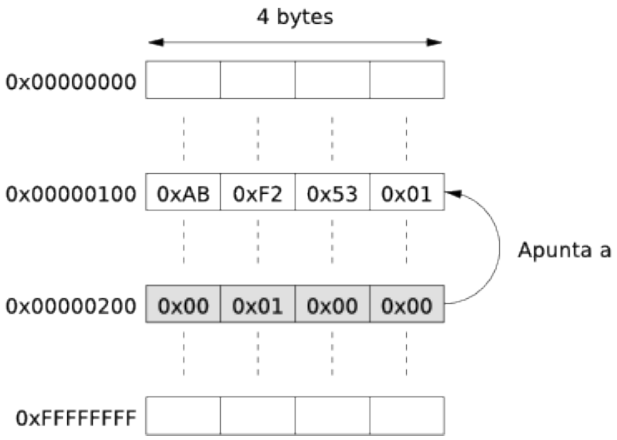
- Supongamos una memoria con unidad de transferencia **4 bytes**.
- Se realizan las siguientes **operaciones** (los bytes se almacenan en el orden en que aparecen).
  - Write(0x100, 0x01234567)
  - Write(0x101, 0x89ABCDEF)
  - Write(0x102, 0xFFFFFFFF)
  - Read(0x100)
- ¿Cuál es el **resultado** de la última operación?



Almacenamiento de Direcciones como Datos

- Una dirección de memoria se puede representar con tantos bits como indica el **tamaño de dirección de memoria**.
- Supongamos una memoria cuya dirección se representa con **32 bits**.
- Tras la operación **Write(0x00001000, 0xFAFBFCFD)** en la posición 0x00001000 se encuentran **cuatro bytes consecutivos** con los valores 0xFA, 0xFB, 0xFC y 0xFD.
- Pero **la dirección de memoria** es ella misma un dato susceptible de **ser almacenado en la propia memoria**.
- **Write(0x00002000, 0x00001000)**: En la posición 0x00002000 se encuentran **cuatro bytes consecutivos** cuyo valor el 0x00001000.
- Podemos concluir que **en la posición de memoria 0x00002000 se encuentra almacenada la dirección del número 0xFAFBFCFD**.

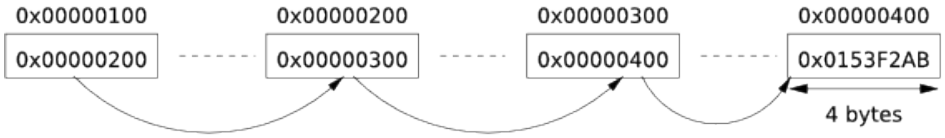
Indirección



- ¿Puedo obtener el dato 0xFAFBFCFD si **sólo conozco** la dirección 0x00002000?
- **Read(Read(0x00002000))**
- Un conjunto de bytes en memoria puede ser interpretado como **entero, coma flotante, dirección, etc.**
- La forma de interpretar un conjunto de bytes **no está almacenada en memoria**

Múltiples Indirecciones

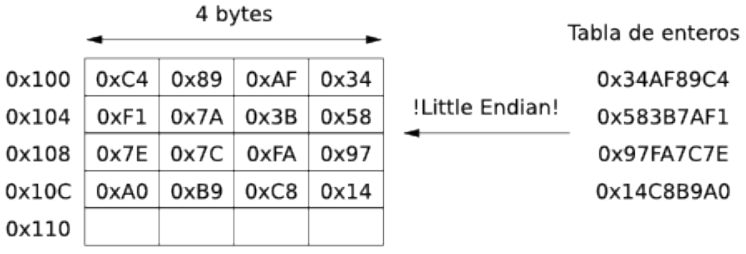
- El concepto de **indirección** se puede encadenar **un número arbitrario de veces**.



- En la memoria, por tanto pueden **coexistir**:
  - Números (enteros, naturales, coma flotante).
  - Letras.
  - Booleanos.
  - Direcciones.
  - Códigos que representan símbolos de un conjunto previamente definido.

Almacenamiento de un Array

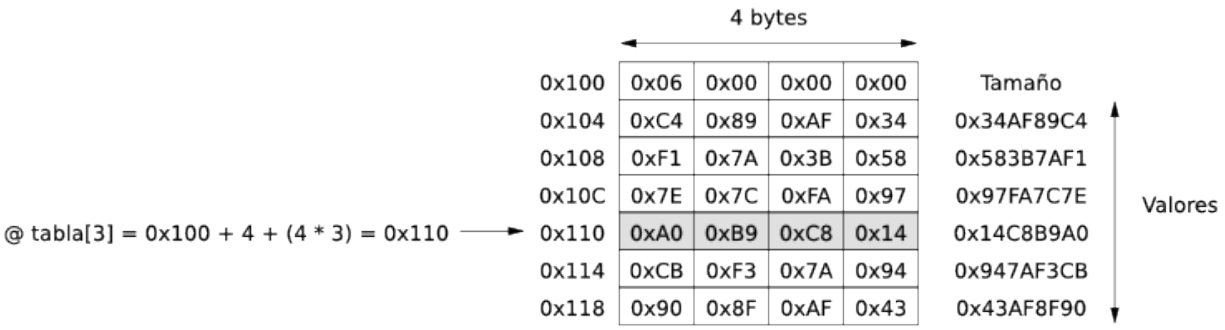
- ¿Cómo se almacena un **array de enteros de 32 bits**?
- Lo más intuitivo es almacenar los elementos **en posiciones consecutivas de memoria**.
- **Ejemplo:** Un array con 4 **enteros** de 32 bits almacenado **a partir de la posición 0x100** ocupa las posiciones desde la 0x100 hasta la 0x10F.
- Si la memoria es la del Pentium **(little endian)**



- ¿Cómo se sabe el **tamaño de un array**?

Almacenamiento de Arrays de Java

- El tamaño de un array es **tan importante** que en Java se almacena como **primer elemento** de la tabla.
- **Todo acceso** a un elemento del array se realiza a través de un índice que se comprueba primero si está en el rango permitido.
- **Ejemplo:** `int[] narray = new int[6];`



- ¿En qué posición de memoria está almacenado `narray[4]`?
- ¿Qué valor asocia el ordenador al símbolo `narray`?