

Universidad de Los Andes  
Facultad de Ingeniería  
Escuela de Sistemas

# Modelos de Programación Paralela

**Prof. Gilberto Díaz**  
**gilberto@ula.ve**

*Departamento de Computación, Escuela de Sistemas, Facultad de Ingeniería  
Universidad de Los Andes, Mérida 5101 Venezuela*

Un modelo de programación paralela o paradigma es un conjunto de tecnologías de software que permiten expresar algoritmos paralelos para implantar aplicaciones en la arquitectura adecuada.

Un modelo de programación paralela incluye distintas áreas:

- Aplicaciones
- Lenguajes de programación
- Compiladores
- Bibliotecas
- Sistemas de comunicación
- Dispositivos de I/O paralelos

Hoy en día es muy difícil realizar un programa paralelo de forma automática.

Por esto, el usuario debe escoger el modelo apropiado, o una mezcla de ellos, para construir su programa

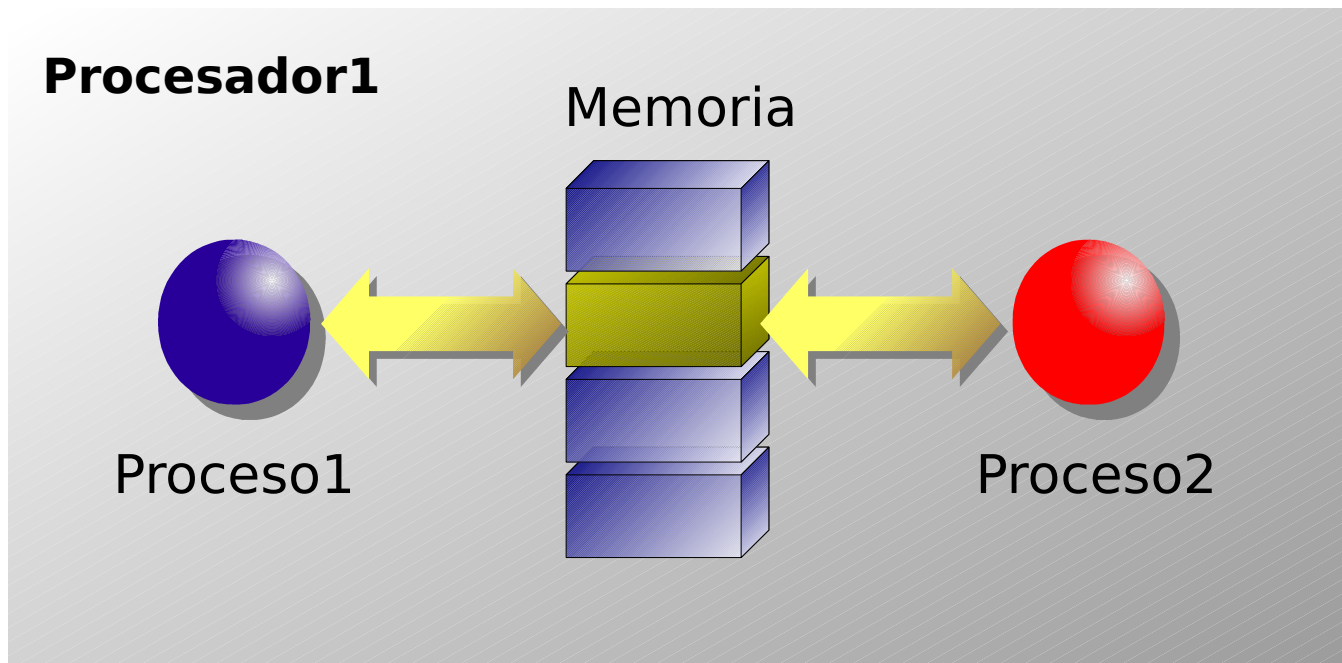
Un modelo de programación paralela es implantado de distintas formas:

- Como bibliotecas invocadas desde programas tradicionales
- Como extensiones de lenguajes de programación
- Como modelos de ejecución completamente nuevos

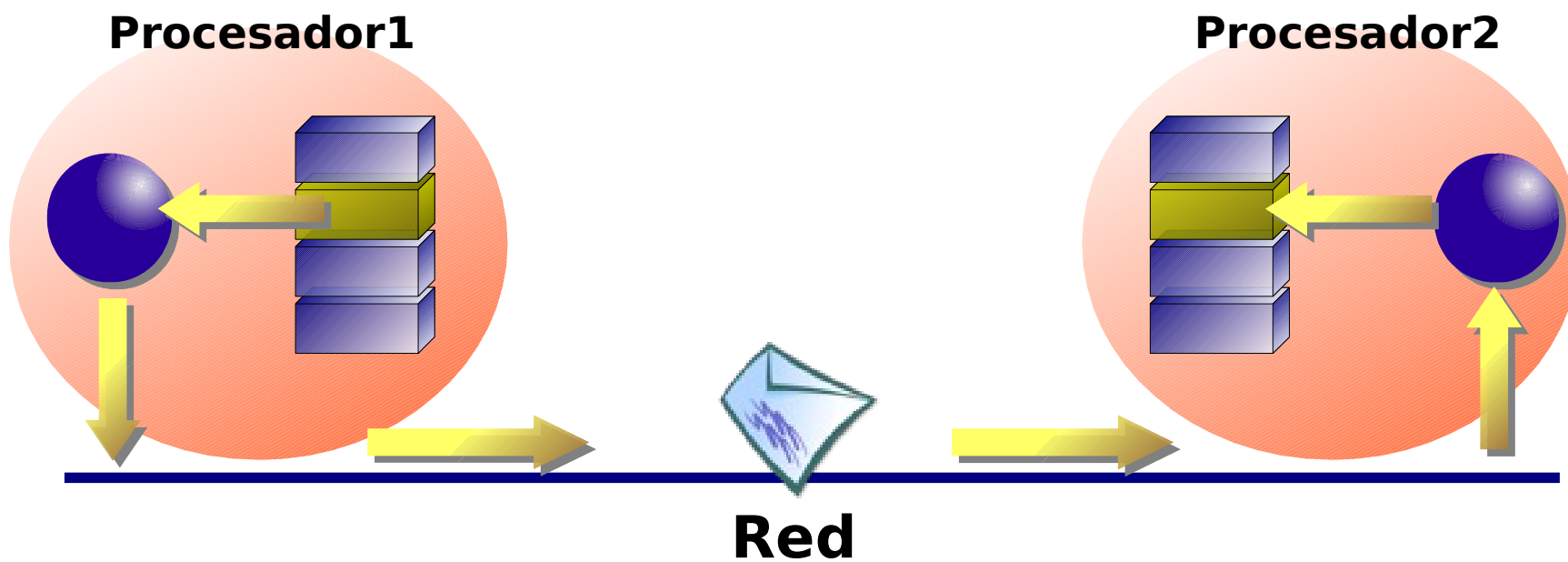
Una primera categorización de estos modelos se realiza de acuerdo al manejo de la memoria:

- Memoria compartida (shared memory)
- Memoria distribuida (distributed memory)
- Memoria compartida distribuida (distributed shared memory)

## Memoria Compartida

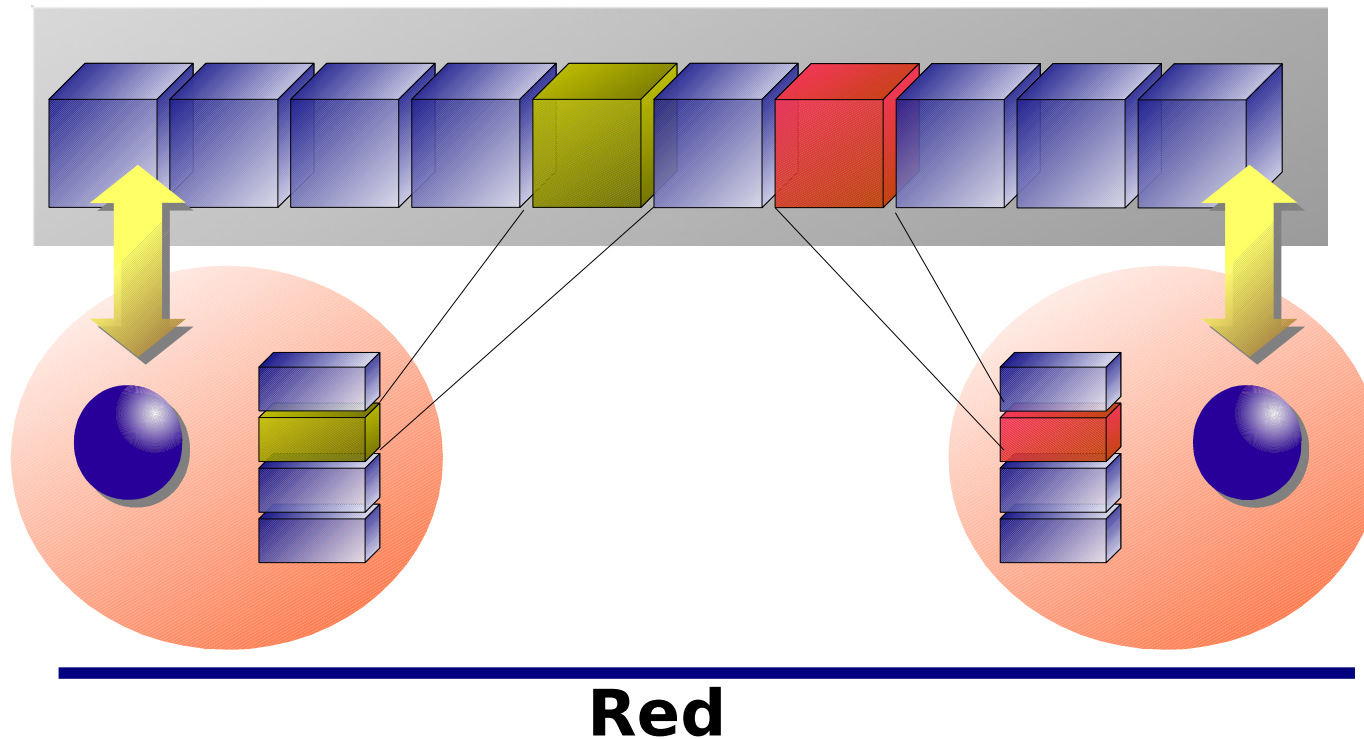


## Pase de Mensajes



## Memoria Compartida Distribuida

(LINDA, munin, etc)



Un modelo de programación paralela o paradigma es juzgado por factores como:

- **Expresividad**
- **Simplicidad**

Actualmente, se considera la productividad en programación como el factor más significativo

Algunos ejemplos de modelos de programación:

- Bibliotecas
  - POSIX threads
  - MPI
  - PVM
  - TTB (templates C++ from Intel)
- Lenguajes
  - ADA
  - HPF
  - LINDA
  - OpenCL

El paralelismo de un programa lo podemos obtener de distintas fuentes:

- Paralelismo de Bits
- Paralelismo a nivel de instrucciones
- Paralelismo a nivel de datos
- Paralelismo funcional

## Paralelismo de Bits

Se basa en el tamaño de la palabra que es capaz de manejar el procesador:

- 8 bits
- 16 bits
- 32 bits
- 64 bits .....

Mientras más grande el tamaño de la palabra menos instrucciones ejecuta el procesador para realizar una operación determinada

### Paralelismo a nivel de instrucciones

En el siguiente conjunto de instrucciones

```
x = y * z;  
a = b + c;  
w = x + a;
```

la tercera instrucción depende de los resultados de la primera y segunda.

Sin embargo, las dos primeras no tienen dependencias y podrían ser ejecutadas simultáneamente.

## Paralelismo a nivel de instrucciones

Uno de los objetivos de los compiladores es encontrar este tipo de instrucciones para agilizar la ejecución del programa.

```
x = y * z;  
a = b + c;  
w = x + a;
```

## Paralelismo a nivel de instrucciones

Mecanismos de la arquitectura son utilizados entonces para ejecutar este tipo de paralelismo:

- Pipelining
- Superscalar
- Ejecución desordenada
- Ejecución especulativa
- Renombramiento de registros
- Predicción de precedencia de memoria
- Predicción de ramificaciones del flujo

## Paralelismo a nivel de instrucciones

- Pipelining

Es una técnica utilizada en el diseño de computadores para incrementar sus prestaciones

Los procesadores se basan en una señal de reloj y lo más natural es realizar una tarea por ciclo

## Paralelismo a nivel de instrucciones

- Pipelining

Este mecanismo consiste en crear un cauce en los circuitos de tal manera que se pueda ejecutar una operacion completa por ciclo

## Paralelismo a nivel de instrucciones

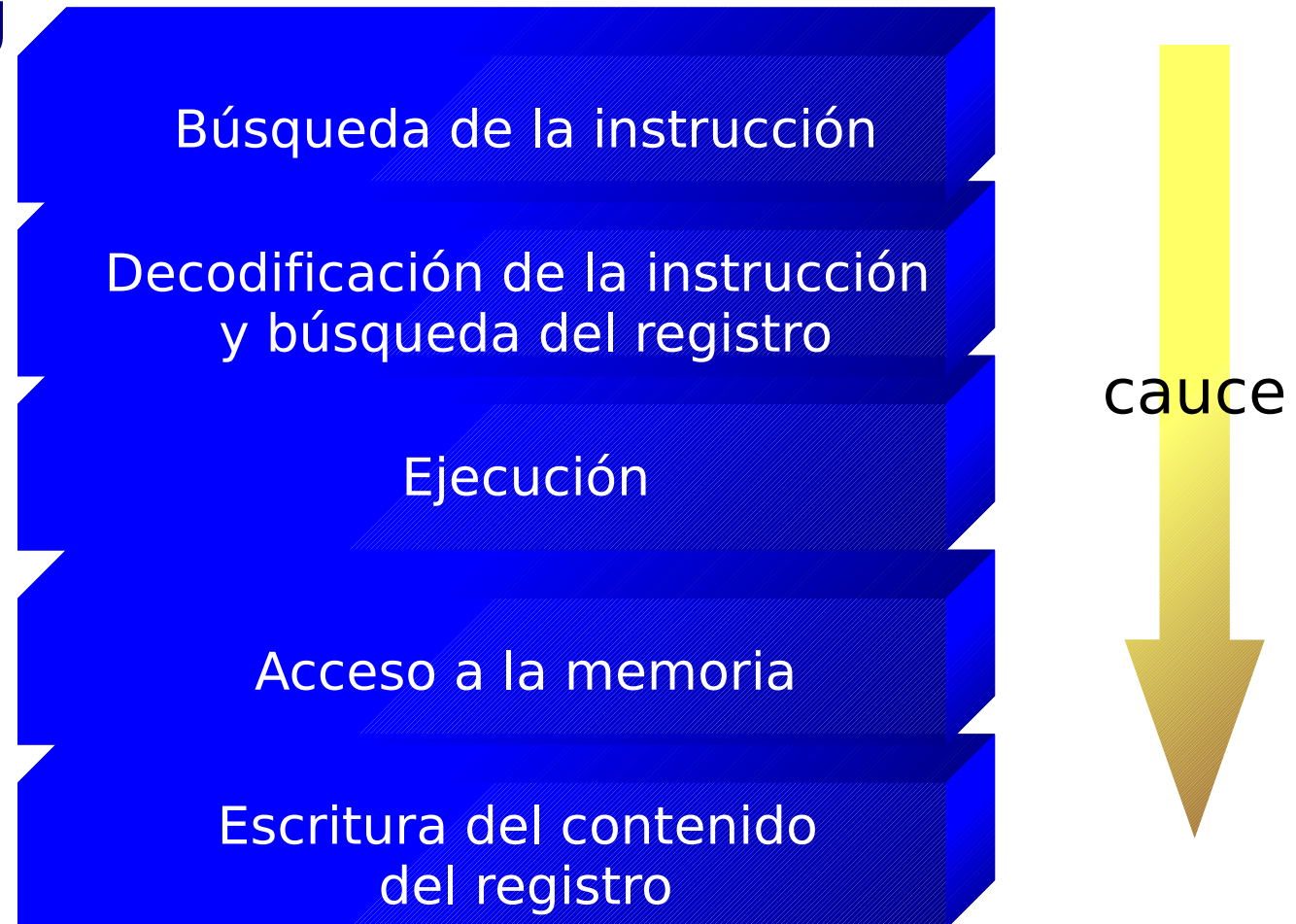
- Pipelining

Una operación envuelve las siguientes tareas:

- Búsqueda de la instrucción
- Decodificación de la instrucción y búsqueda del registro
- Ejecución
- Acceso a la memoria
- Escritura del contenido del registro

## Paralelismo a nivel de instrucciones

- Pipelining



## Paralelismo a nivel de instrucciones

- **Pipelining**

Ventajas:

- El ciclo de reloj se reduce y se incrementa el ancho de banda

Desventajas:

- No todas las operaciones son aptas para un cauce
- El ancho de banda no se puede predecir

## Paralelismo a nivel de instrucciones

- Pipelining

Los Pentium 4 tienen 20 capas

Los Pentium D tienen hasta 31 capas

## Paralelismo a nivel de instrucciones

- Superscalar

Para incrementar las capacidades de un procesador muchas veces se le agrega más de una unidad lógica aritmética

De esta manera es posible ejecutar más de una operación por ciclo de reloj

## Paralelismo a nivel de instrucciones

- Ejecución desordenada de instrucciones

La ejecución natural de instrucciones es:

- Búsqueda de la instrucción
- Si los operandos están disponibles en los registros del procesador, ésta es despachada a una unidad funcional. Si no, el procesador debe esperar hasta que los datos se carguen desde la memoria

## Paralelismo a nivel de instrucciones

- Ejecución desordenada de instrucciones
  - Una vez que están disponibles entonces se ejecuta la operación y el resultado se guarda.

## Paralelismo a nivel de instrucciones

- Ejecución desordenada de instrucciones

En la ejecución desordenada el proceso es diferente

- Cada instrucción se coloca en una cola
- Cuando los operandos están disponibles los resultados también se colocan en una cola.
- Cuando todas las instrucciones anteriores hayan escrito los resultados a memoria, entonces el resultado es almacenado

## Paralelismo a nivel de instrucciones

- Ejecución especulativa
- Predicción de ramificaciones del flujo

Las instrucciones en un programa pueden ser divididas en:

- Instrucciones que deben ejecutarse y son obligatorias
- Instrucciones que no son necesarias y pueden no ejecutarse pues son irrelevantes
- Instrucciones que no se puede probar que pertenezcan a los dos grupos anteriores

## Paralelismo a nivel de instrucciones

- Ejecución especulativa
- Predicción de ramificaciones del flujo

En un conjunto de instrucciones donde se involucre una condición, se puede ejecutar la rama que más probablemente sea seleccionada.

Este conjunto se ejecuta concurrentemente y cuando se llegue al punto de la evaluación de la condición y se debe ejecutar realmente las instrucciones se tiene una ganancia

## Paralelismo a nivel de instrucciones

- Ejecución especulativa
- Predicción de ramificaciones del flujo

Si no, no se pierde nada y se continua con la ejecución normal del programa.

## Paralelismo a nivel de instrucciones

- Renombramiento de registros

Consideremos el siguiente conjunto de instrucciones:

RA = MEM[100]

RA = RA + 2

MEM[101] = RA

RA = MEM[138]

RA = RA \* 2

MEM[139] = RA

## Paralelismo a nivel de instrucciones

- Renombramiento de registros

Si se renombra el registro utilizado por las tres últimas instrucciones, estas se pueden ejecutar concurrentemente

RA = MEM[100]

RA = RA + 2

MEM[101] = RA

RB = MEM[138]

RB = RB \* 2

MEM[139] = RB

## Paralelismo a nivel de instrucciones

- Predicción de dependencia de memoria

Es una técnica empleada por procesadores que ejecutan instrucciones desordenadamente, para realizar operaciones de acceso de memoria (loads and stores) con la finalidad de predecir dependencias entre las operaciones, en el tiempo de ejecución.

## Paralelismo a nivel de instrucciones

- Predicción de dependencia de memoria

Con la información de dependencias predecidas, el procesador puede decidir ejecutar ejecutar especulativamente tales operaciones de memoria.

## Paralelismo a nivel de datos

Este tipo de paralelismo se enfoca en la distribución de los datos entre varios procesadores.

Se conoce también como **paralelismo a nivel de lazos** (loop-level parallelism)

## Paralelismo a nivel de datos

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

## Paralelismo a nivel de datos

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix}$$

## Paralelismo a nivel de datos

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

## Paralelismo a nivel de datos

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

## Paralelismo a nivel de datos

$$C_{00} = a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20}$$

$$C_{01} = a_{00}b_{01} + a_{01}b_{11} + a_{02}b_{21}$$

$$C_{02} = a_{00}b_{02} + a_{01}b_{12} + a_{02}b_{22}$$

.....

$$C_{22} = a_{20}b_{02} + a_{21}b_{12} + a_{22}b_{22}$$

## Paralelismo a nivel de datos

$$C_{00} = a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20}$$



$$C_{01} = a_{00}b_{01} + a_{01}b_{11} + a_{02}b_{21}$$



$$C_{02} = a_{00}b_{02} + a_{01}b_{12} + a_{02}b_{22}$$



.....

$$C_{22} = a_{20}b_{02} + a_{21}b_{12} + a_{22}b_{22}$$



## Paralelismo a nivel funcional

En este caso un programa paralelo que ejecuta cálculos distintos sobre el mismo conjunto de datos o sobre datos diferentes.

El paralelismo funcional generalmente no escala con el tamaño del problema.

## Paralelismo a nivel funcional

Obtener distintos resultados a partir de un mismo conjunto de datos, por ejemplo:

Para un matriz hallar

- El determinante
- La traspuesta
- La inversa