

# Diseño de Compiladores I

Introducción

# Temas

---

- ▶ **Introducción a los compiladores**
- ▶ **Historia**
- ▶ **Conceptos relacionados con compiladores**
- ▶ **Estructura de un compilador**



# Introducción

---

*Los compiladores son programas de computadora que traducen un lenguaje a otro.*



# Introducción

---

- ▶ Un compilador toma como entrada un programa escrito en su **lenguaje fuente** y produce un programa equivalente escrito en su **lenguaje objetivo**.



# Salidas

---

- ▶ **Otro lenguaje**
  - ▶ Ada → C++ → Assembler → Binario
- ▶ **Assembler**
  - ▶ Facilita debugging
  - ▶ Depende de la máquina para la que se escribió.
- ▶ **Código binario / objeto**
  - ▶ Debugging complicado.
  - ▶ Permite compilaciones separadas,
  - ▶ Requiere linker.
- ▶ **Ejecutable (absoluto)**
  - ▶ No permite compilaciones separadas.
- ▶ **Otra CPU / Otro SO**

Cuanto más alto sea el nivel de la salida, más portable será.

---



# Un poco de historia...

---

## ***Fines de los 40's***

- ▶ Computadora con programa almacenado  
( John von Neumann)

→ Programas: secuencias de códigos

**C7 05 0000 0002**

***Lenguaje de máquina***



(instrucción para mover el número 2 a la ubicación 0000 )



# Un poco de historia...

---

## **Lenguaje Ensamblador (Assembler):**

Formas simbólicas para representar localidades de memoria e instrucciones

**MOV X , 2**

(instrucción para mover el número 2 a la ubicación X, donde la localidad de memoria simbólica X es 0000 )



## Un poco de historia...

---

→ Necesidad de escribir las operaciones de un programa

- ▶ de una manera concisa,
- ▶ parecida a la notación matemática o lenguaje natural,
- ▶ independiente de cualquier máquina en particular.

$$X = 2$$

→ Necesidad de traducción a un código ejecutable.





# Un poco de historia...

---

## **1954 a 1957**

- ▶ Un equipo en IBM dirigido por John Backus desarrolla FORTRAN y su compilador.
- ▶ Chomsky estudia la estructura del lenguaje natural, arribando a la **Jerarquía de Chomsky**:
  - ▶ Gramáticas tipo 0, 1, 2 y 3
  - ▶ Las gramáticas de tipo 2 o **gramáticas independientes de contexto** se utilizan hoy para representar las estructuras de los lenguajes de programación.



# Un poco de historia...

---

## ***Décadas del 60 y 70***

- ▶ Problema del ***análisis sintáctico***:
  - ▶ Determinación de algoritmos eficientes para el reconocimiento de lenguajes independientes de contexto.
- ▶ ***Autómatas finitos y expresiones regulares*** (gramáticas tipo 3 de Chomsky):
  - ▶ Proveen métodos simbólicos para expresar la estructura de las palabras o tokens de un lenguaje de programación.
- ▶ ***Métodos para la generación de código***:
  - ▶ Se desarrollan ***Técnicas de mejoramiento de código***.
    - ▶ Más complejo que los anteriores.
    - ▶ Comenzó con los primeros compiladores y continúa.



# Un poco de historia...

---

## 1975

- ▶ Steve Johnson crea Yacc (Yet another compiler compiler) para Unix.

Es un **generador de analizadores sintácticos**.

- ▶ Se desarrollan **generadores de analizadores léxicos**. El más conocido es Lex, desarrollado por Mike Lesk para Unix.

## **Fines de los 70 y principios de los 80:**

- ▶ Se intenta automatizar otras partes de un compilador, con poco éxito.



# Un poco de historia...

---

## **Avances más recientes**

- ▶ Incorporación de los compiladores a **ambientes de desarrollo interactivo (IDE)**, incluyendo:
  - ▶ Editores
  - ▶ Linkers
  - ▶ Debuggers
  - ▶ Administradores de proyectos.
- ▶ Poca estandarización



---

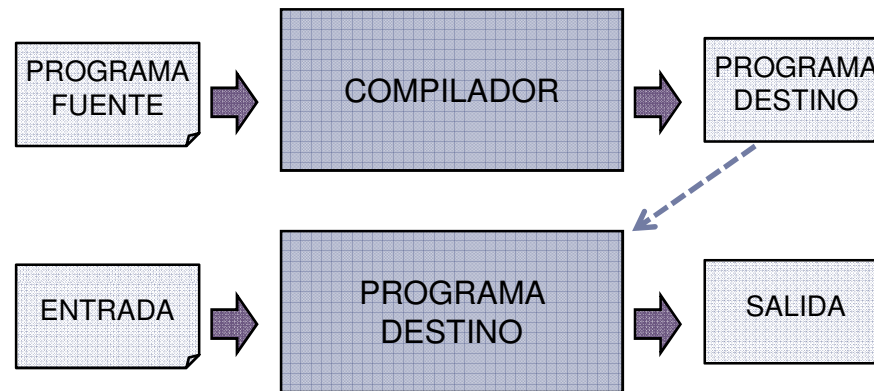
Volviendo al proceso de traducción...



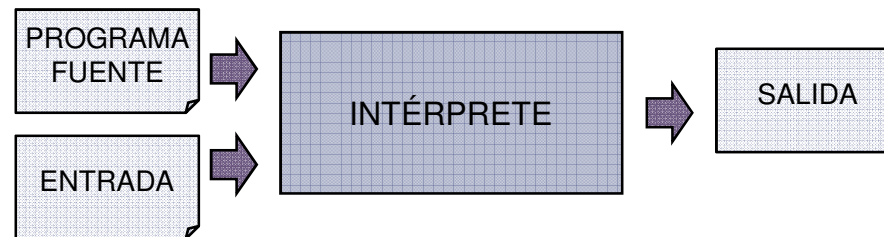
# Compilación e interpretación

---

- ▶ Un **compilador** genera un código objetivo que se ejecuta después que se completa la traducción.



- ▶ Un **intérprete** es un traductor de lenguaje que ejecuta el programa fuente inmediatamente.



# Compilación e interpretación

---

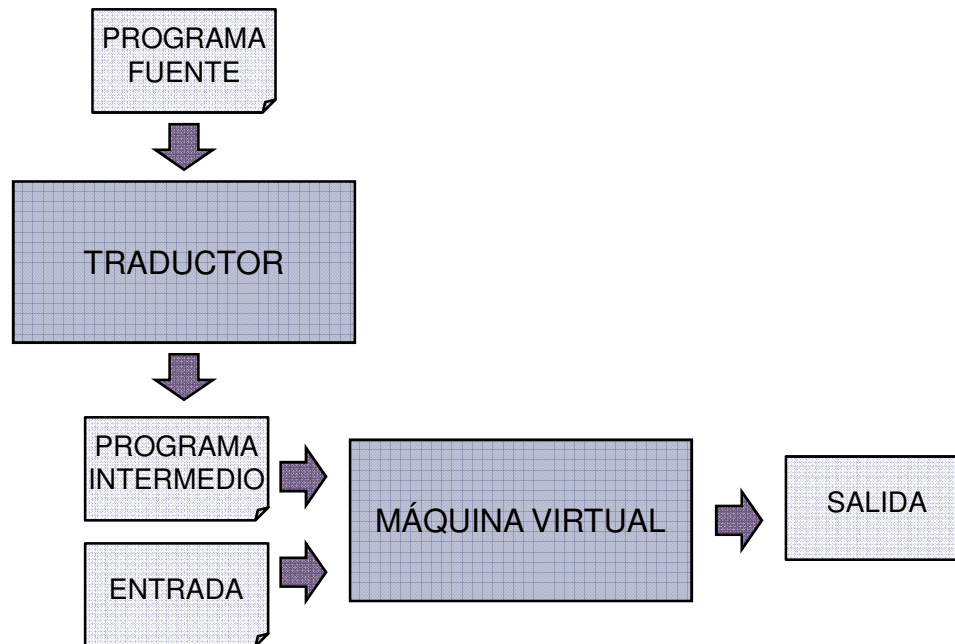
- ▶ En principio, cualquier lenguaje de programación se puede interpretar o compilar.
  - ▶ *Compilador*
    - ▶ Mayor velocidad en la salida
  - ▶ *Intérprete*
    - ▶ Mejores diagnósticos de error
- ▶ Ejemplos de lenguajes compilados: Ada, C, C++, Pascal, etc.
- ▶ Ejemplos de lenguajes interpretados: Basic, Prolog, LISP, etc.



# Compilación e interpretación

---

## ▶ *Traductores híbridos:*



## ▶ Ejemplo; Java

---





# En la historia...

---

- ▶ El concepto de *máquina objeto* apareció en los 70's.
  - ▶ Se genera código para una máquina imaginaria.
  - ▶ En la máquina real se interpreta.

## Ejemplos:

- ▶ p-Machine
- ▶ p-Code
- ▶ Existieron intentos de *máquinas lenguaje*.
  - ▶ La máquina entiende el lenguaje de alto nivel.

## Ejemplos:

- ▶ Máquina LISP
  - ▶ Máquina FORTH
- 



# Máquina objeto → Máquina virtual

---

- ▶ La máquina objeto quedó en desuso hasta la aparición de Java.
- ▶ Un compilador genera código (bytecode) para la máquina virtual.
  - ▶ El código se interpreta
  - ▶ El código se compila con un compilador *just-in-time*.
- ▶ Ejemplos:
  - ▶ Java
  - ▶ Perl
  - ▶ PHP
  - ▶ Python

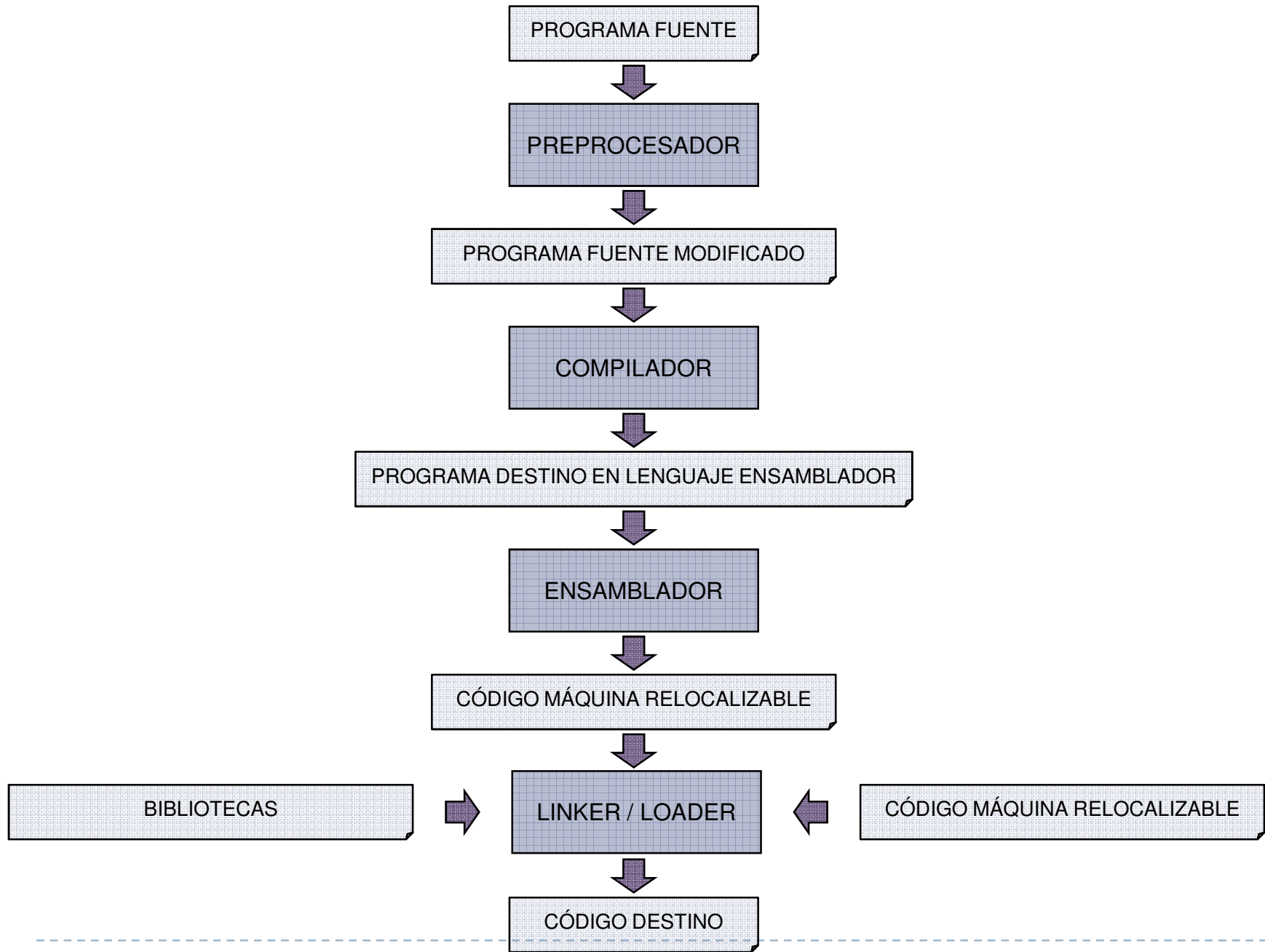


---

¿Es el compilador el único programa  
utilizado en la traducción?

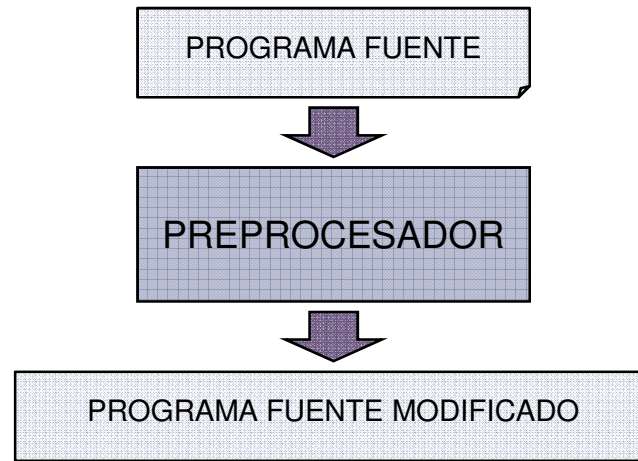
---





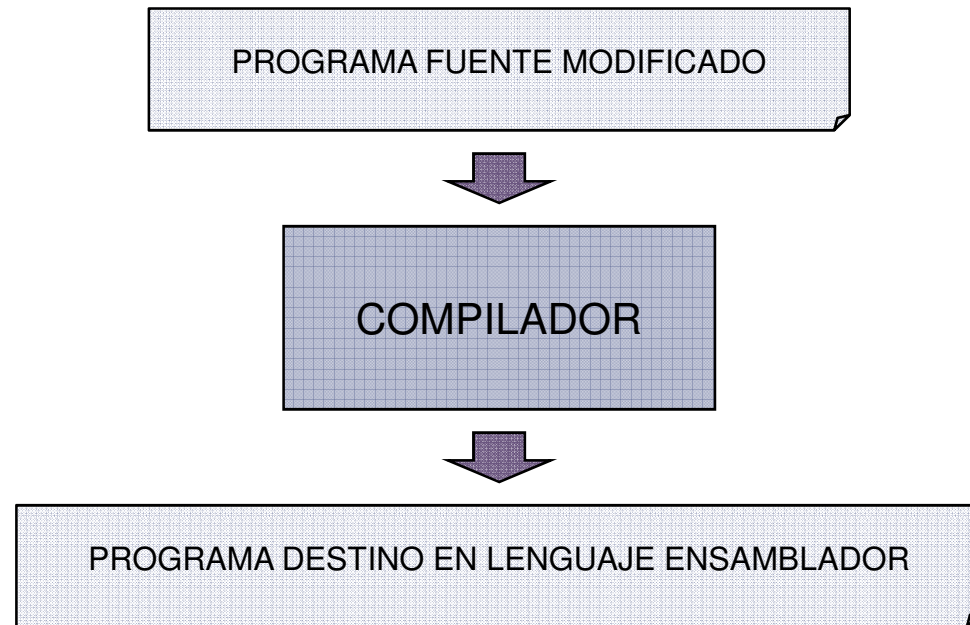
# Preprocesadores

---



- ▶ Recolecta módulos de código ubicados en archivos diferentes
- ▶ Expande macros





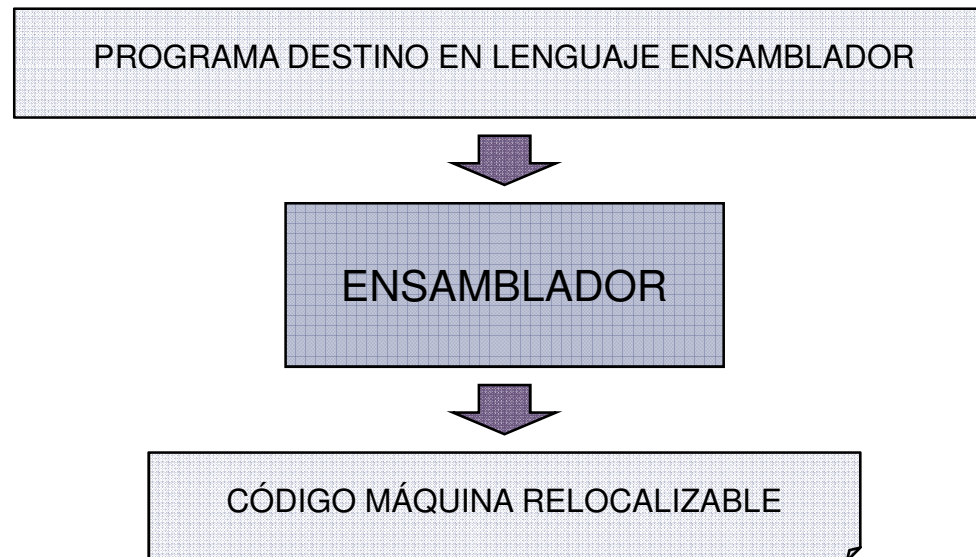
- ▶ Un compilador que genera lenguaje ensamblador como su salida, requiere un ensamblador para terminar la traducción.



# Ensambladores

---

- ▶ Un ensamblador es un traductor para el lenguaje ensamblador (Assembler) de una computadora en particular.



# Compilaciones separadas

---

PF1 → Salida 1  
PF2 → Salida 2  
Bibliotecas



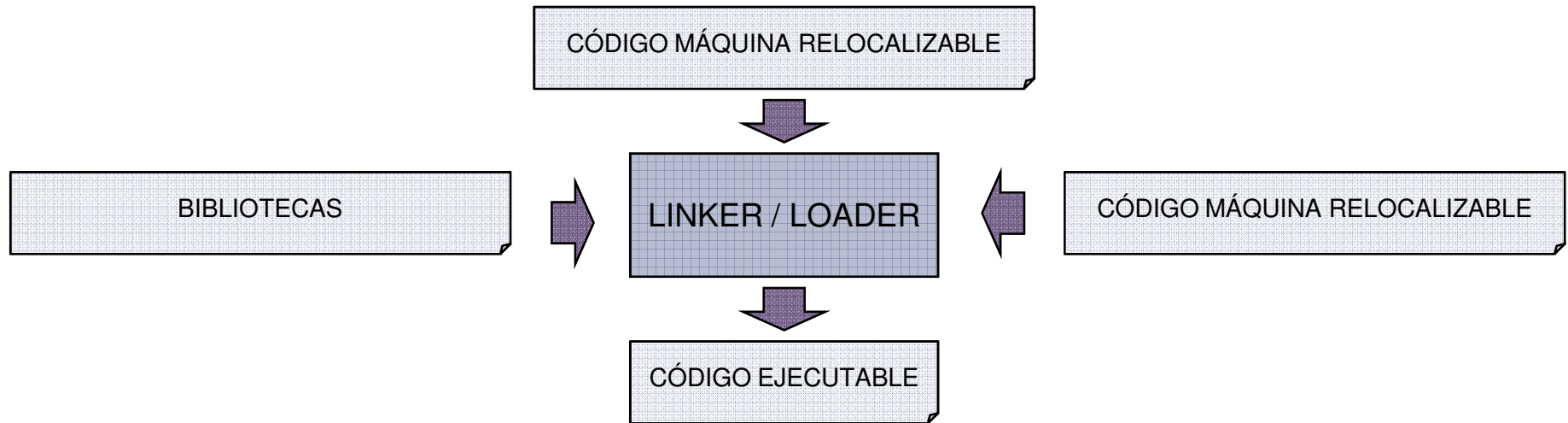
Generación del Ejecutable





# Vinculación

---



- ▶ *Linker*, ligador o enlazador
- ▶ *Loader* o cargador



# Linker / Loader

---

- ▶ El **linker** recopila:
  - ▶ código compilado o ensamblado por separado en diferentes archivos objeto
  - ▶ código de funciones de librerías estándar
- ▶ Se conecta el programa objeto con:
  - ▶ recursos suministrados por el sistema operativo de la computadora:
    - ▶ asignadores de memoria
    - ▶ dispositivos de entrada y salida.
- ▶ El proceso de vinculación está fuertemente ligado al sistema operativo y al procesador.



---

# Compiladores con características especiales

---



# Cross-Compilers

---

- ▶ Se ejecutan en una máquina, pero producen una salida para otra.
  - ▶ Ejemplo: Compilador de C++ que corre en SUN, pero su salida se ejecuta en PENTIUM.
- ▶ Usos:
  - ▶ Aplicaciones móviles.
  - ▶ Aplicaciones empotradas,.
    - ▶ Ejemplos:
      - Teléfonos públicos
      - Contadores de billetes
      - Etc.

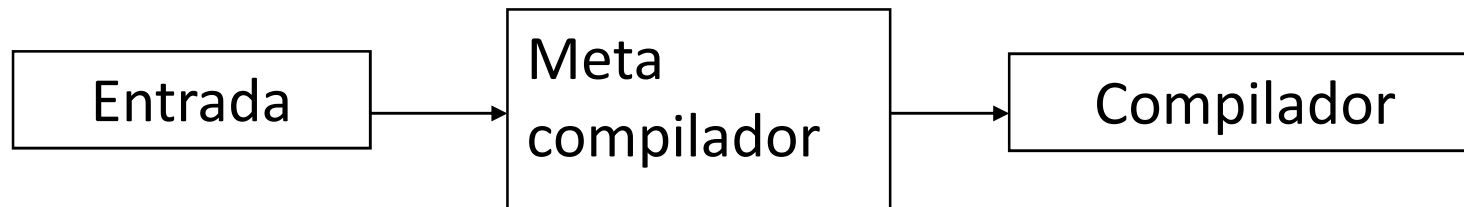
(la aplicación corre en dispositivos no aptos para desarrollo)



# Metacompiladores

---

Son compiladores de compiladores



- ▶ Para generar automáticamente un compilador, la entrada debería formalizar:
  - ▶ Sintaxis del lenguaje
  - ▶ Semántica del lenguaje
  - ▶ Arquitectura de la CPU
  - ▶ Descripción del SO
- ▶ Sólo se ha logrado formalizar la sintaxis mediante BNF.
  - En los 70's se crea **Yacc**, cuya salida es lenguaje C.
  - Hacia 1999 empiezan a aparecer versiones de Yacc para otros lenguajes, y hoy en día, existen versiones para la mayoría de los lenguajes.



# Compiladores incrementales

---

- ▶ Se utilizaron a principios de los 90's. En estos compiladores, se aprovecha para ir avanzando en la compilación, a medida que se edita.
- ▶ No prosperaron.
- ▶ Los editores guiados por sintaxis tienen su origen en este tipo de compiladores.



# Autocompiladores

---

- ▶ Son compiladores escritos en el mismo lenguaje que compilan.
  - ▶ Ejemplo: Un compilador de C++ escrito en C++.
- ▶ El compilador de Ada de Linux, es un autocompilador. Está escrito en Ada y la salida es C++, y el compilador de C++ está escrito en C++.
- ▶ En general, todos los compiladores de C y C++ están escritos en C.



# Diagramas T

Notación para representar implementaciones

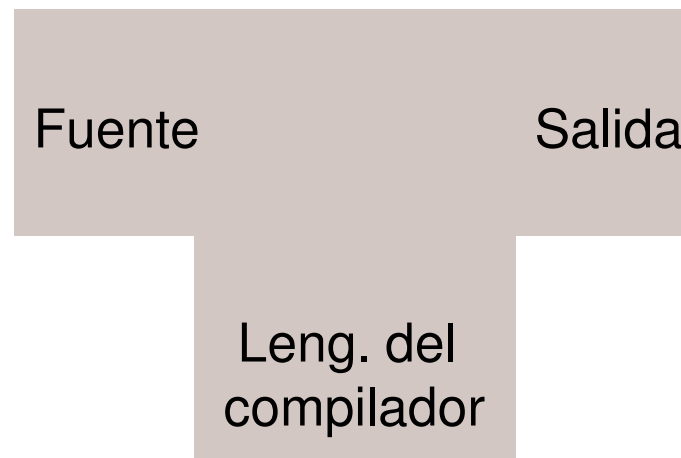


# Diagramas T

Notación para representar implementaciones

---

- ▶ En un compilador, hay 3 lenguajes involucrados:



Fuente del compilador

---

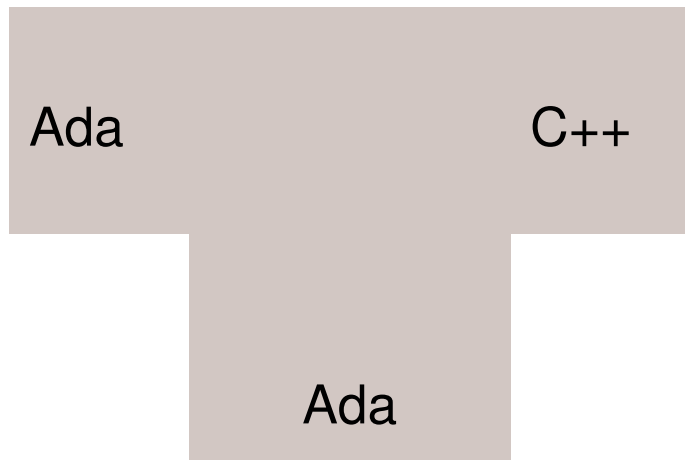


# Diagramas T

Notación para representar implementaciones

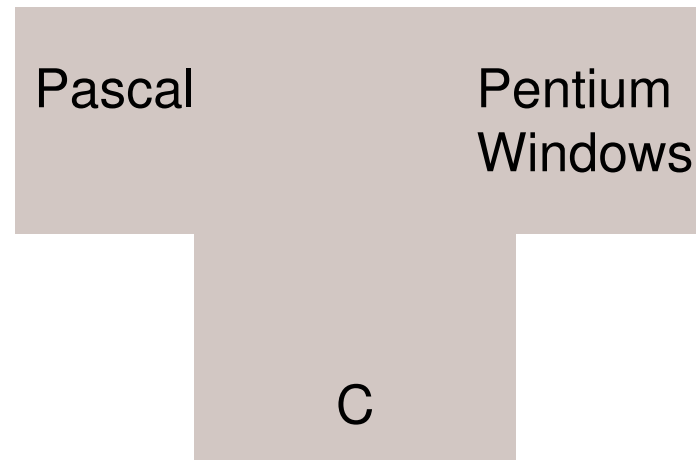
---

Ejemplo 1



Autocompilador de Ada

Ejemplo 2



Compilador de Pascal  
escrito en C

---

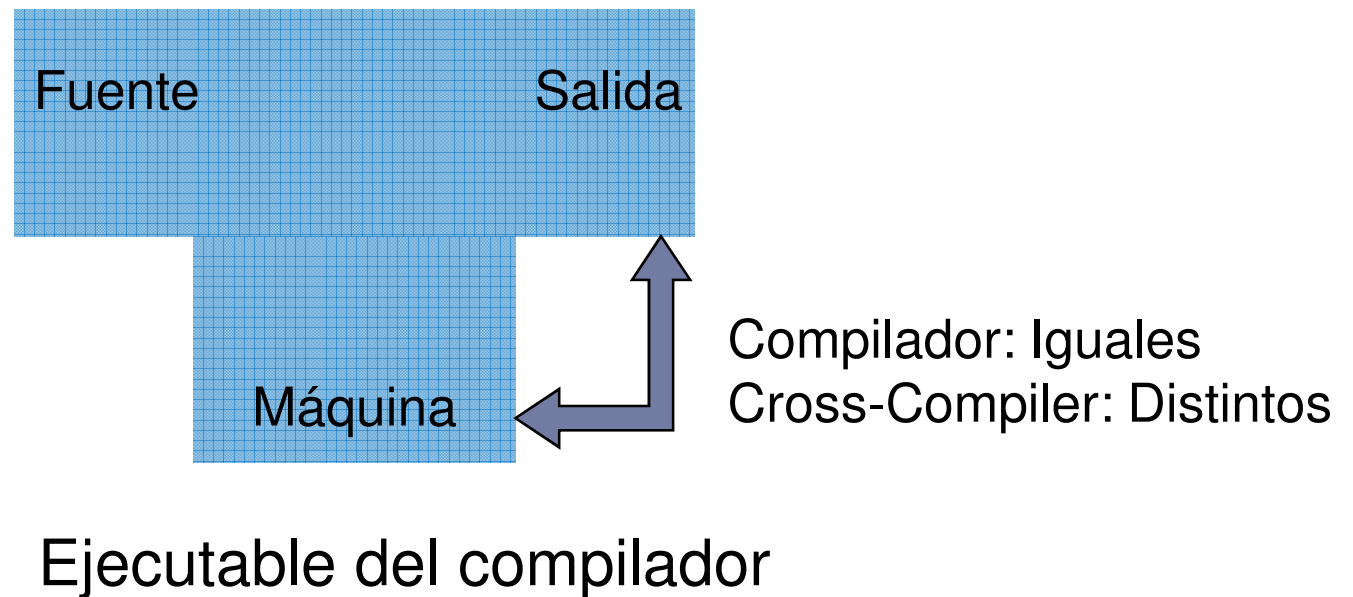


# Diagramas T

Notación para representar implementaciones

---

- ▶ Un compilador tiene un ejecutable:

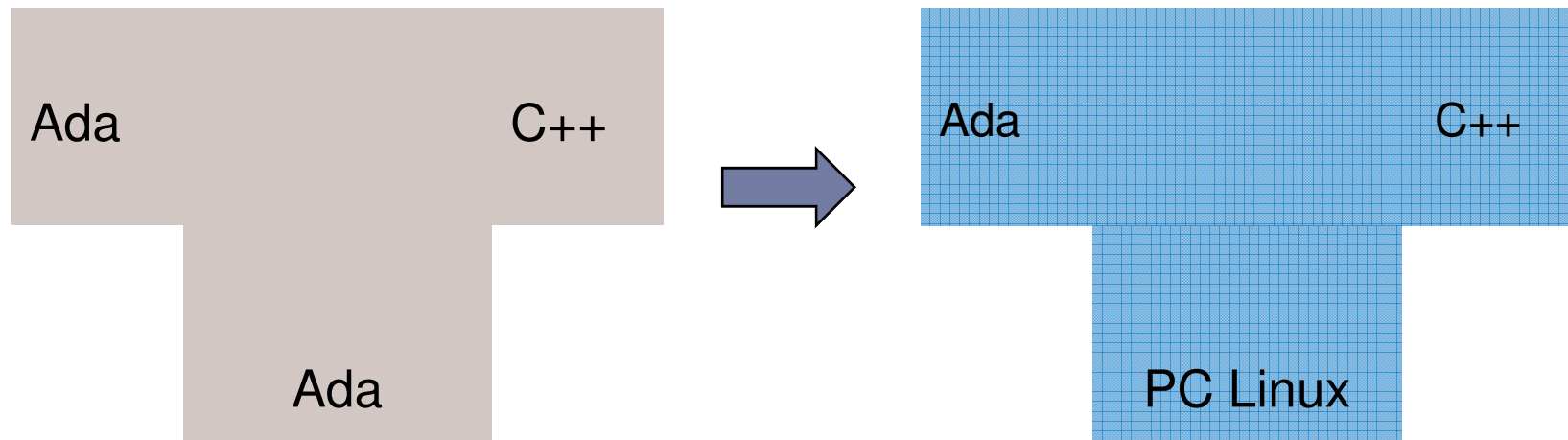


# Diagramas T

Notación para representar implementaciones

---

## Ejemplo I: Autocompilador de Ada



Fuente del compilador

Ejecutable del compilador

- ▶ Es necesario compilar el fuente para obtener el ejecutable
- ▶ Puedo necesitar uno o más compiladores

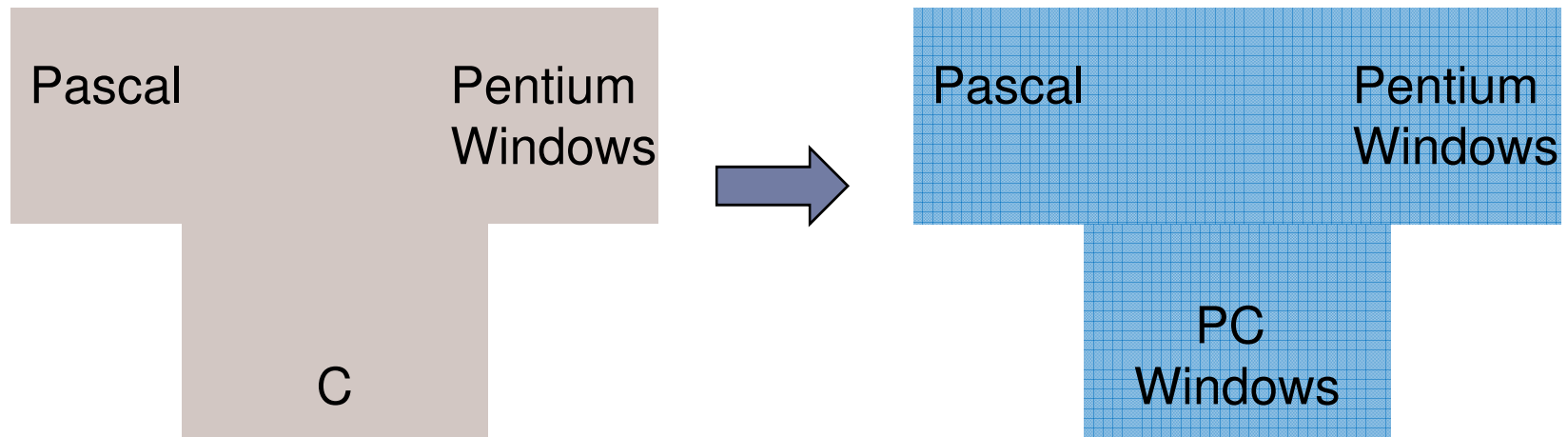


# Diagramas T

Notación para representar implementaciones

---

## Ejemplo 2: Compilador de Pascal escrito en C



Fuente del compilador

Ejecutable del compilador

- ▶ Es necesario compilar el fuente para obtener el ejecutable

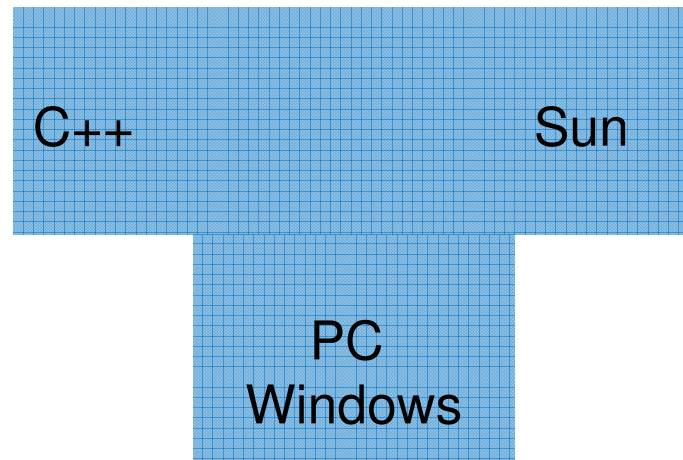


# Diagramas T

Notación para representar implementaciones

---

▶ **Ejemplo 3: Cross-Compiler**



Ejecutable del compilador

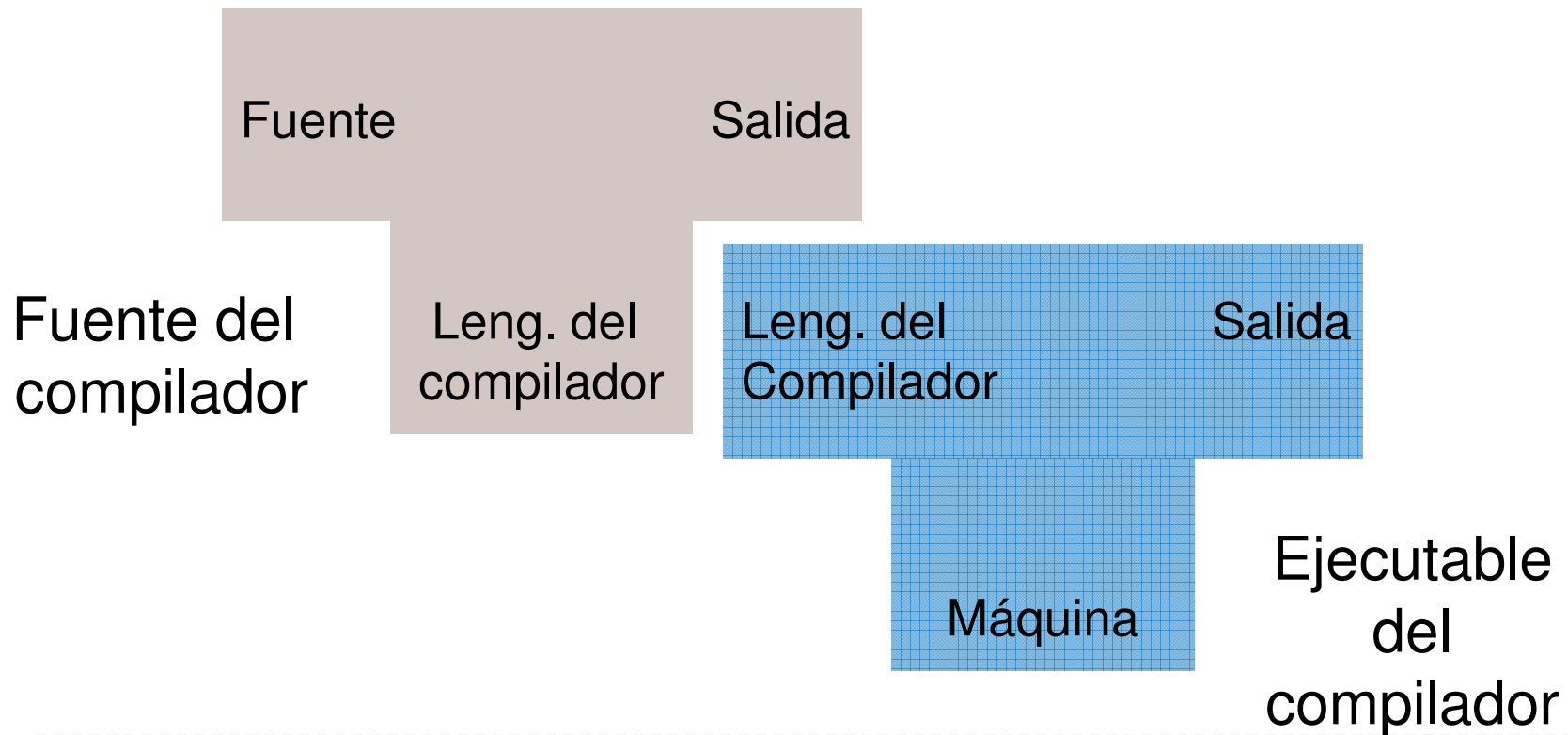


# Diagramas T

Notación para representar implementaciones

---

- ▶ ¿Cómo se compila un compilador?

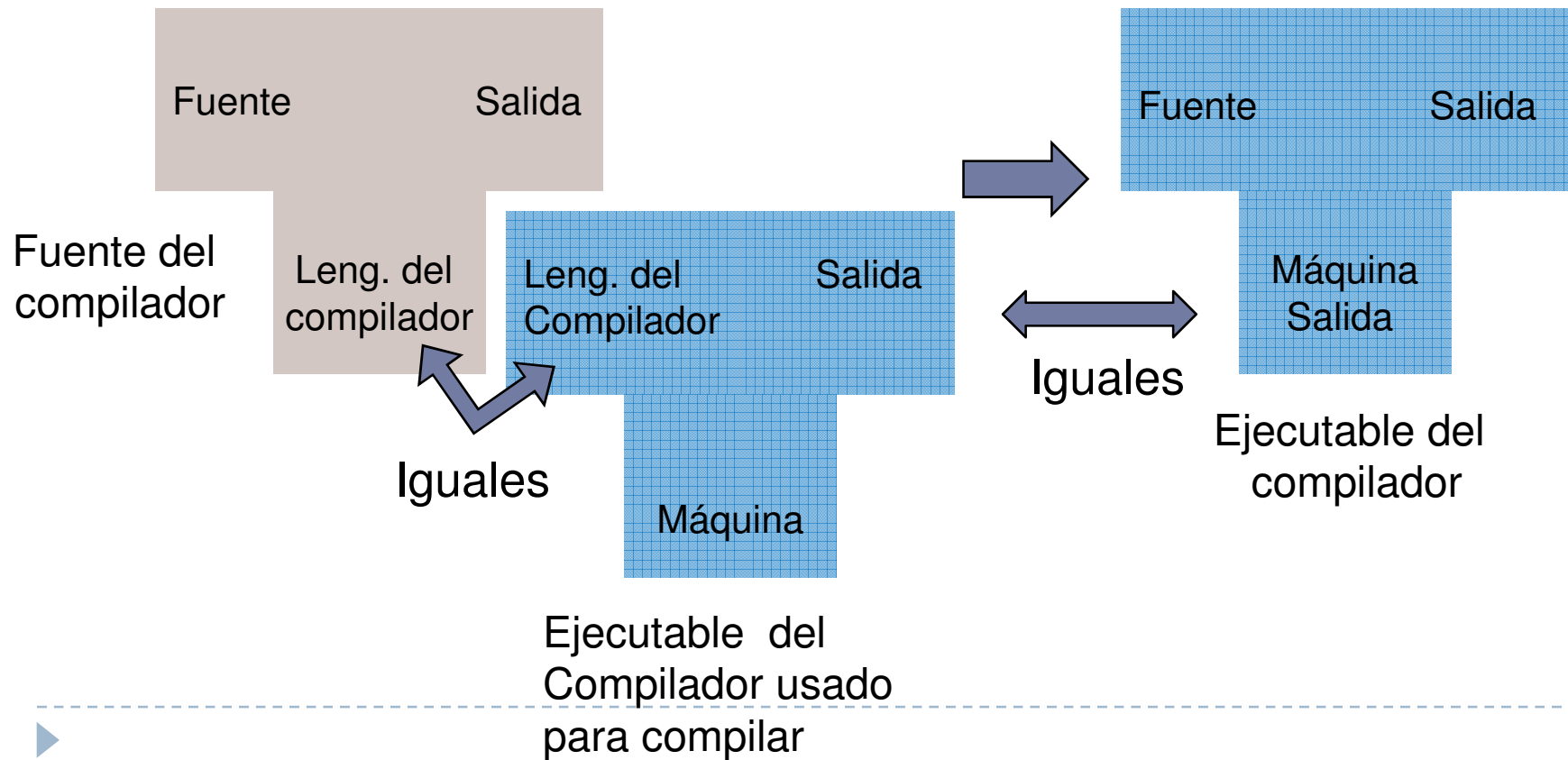


# Diagramas T

Notación para representar implementaciones

---

## ► ¿Cómo se compila un compilador?



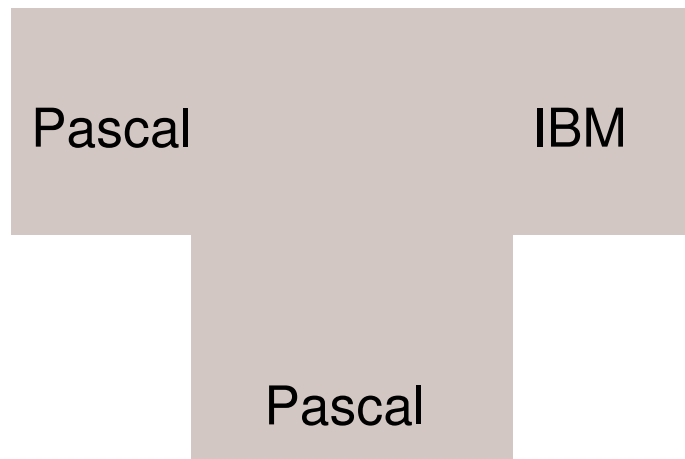


# Diagramas T

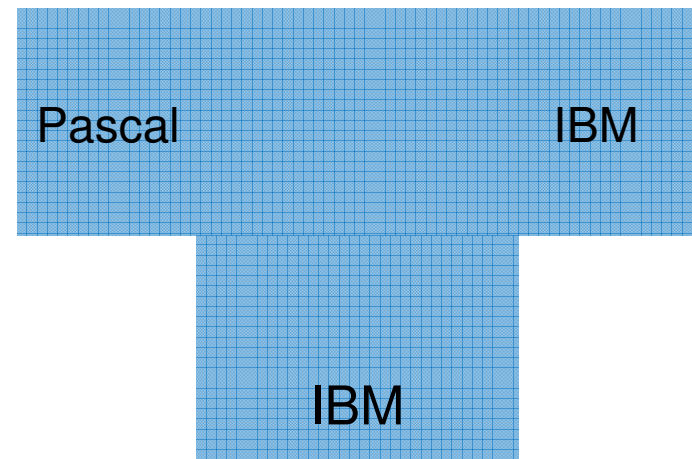
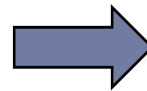
Notación para representar implementaciones

---

## Autocompilador de Pascal



Fuente del compilador



Ejecutable del compilador

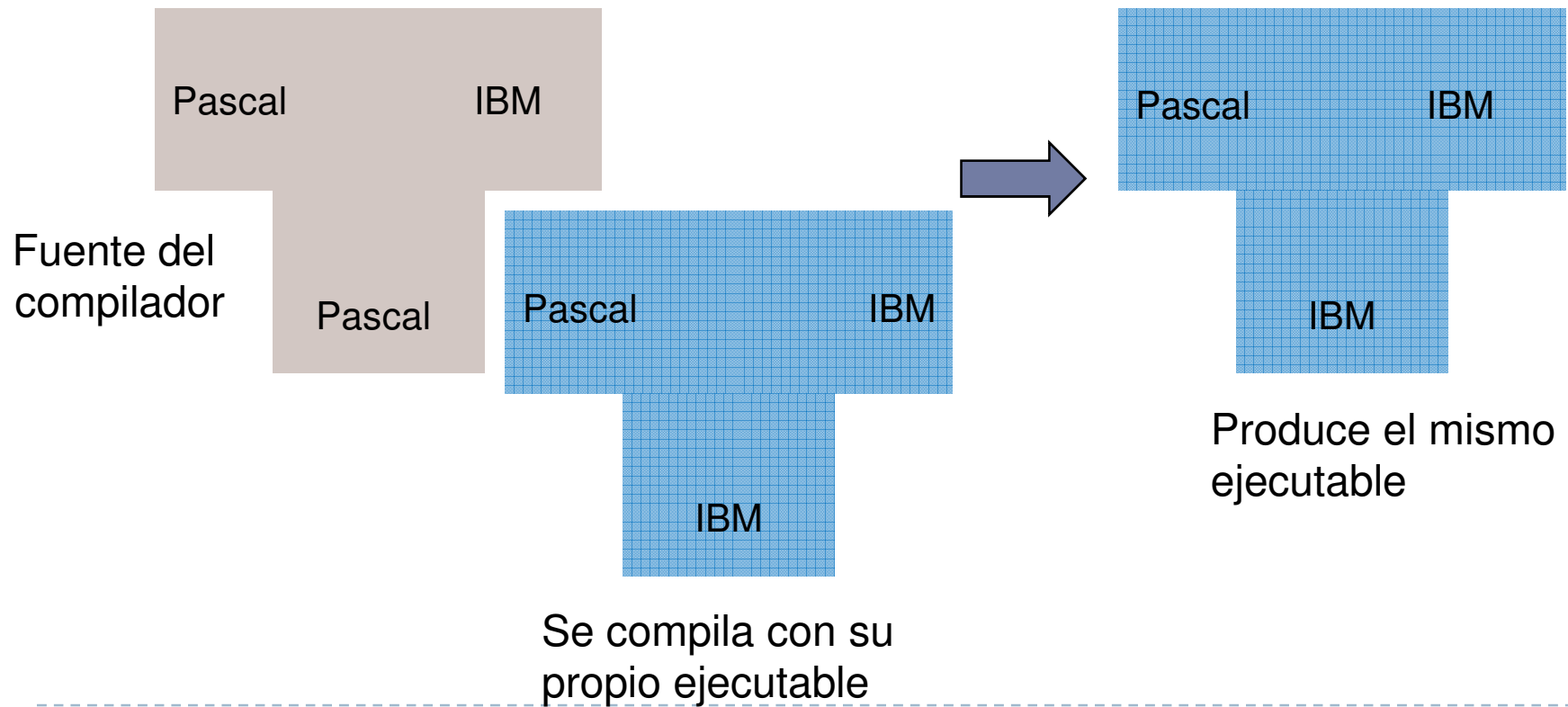


# Diagramas T

Notación para representar implementaciones

---

## ► Compilación de un autocompilador

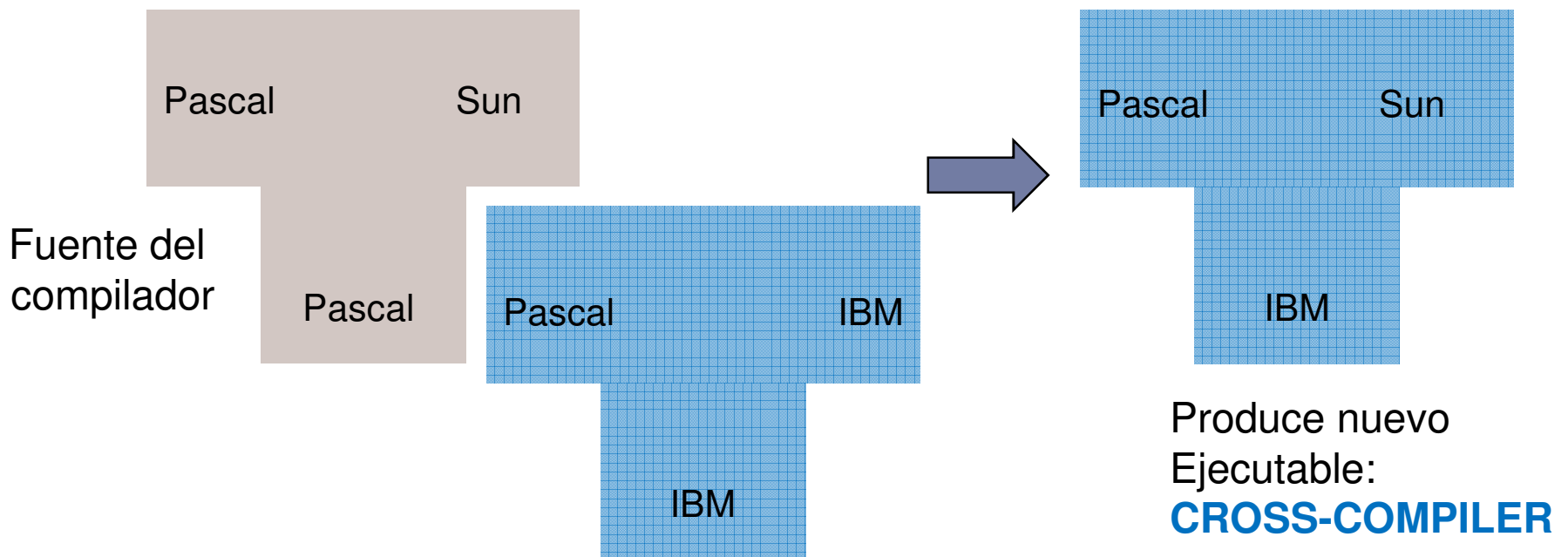


# Diagramas T

## Notación para representar implementaciones

---

- ▶ Migración a otra CPU (Sun) - Paso I:
  - ▶ Se modifica el fuente para que produzca salida Sun
  - ▶ Se compila con el único ejecutable existente

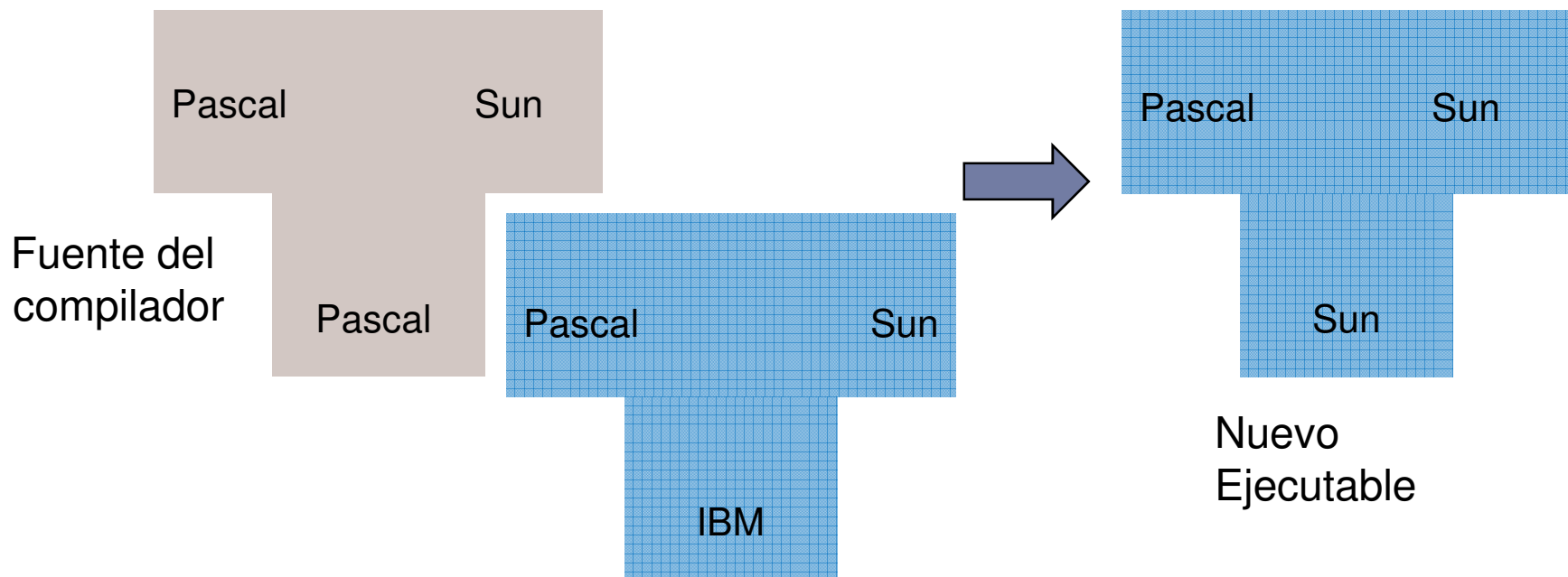


# Diagramas T

Notación para representar implementaciones

---

- ▶ Migración a otra CPU (Sun): Paso 2
  - ▶ Se compila con el CROSS-COMPILER

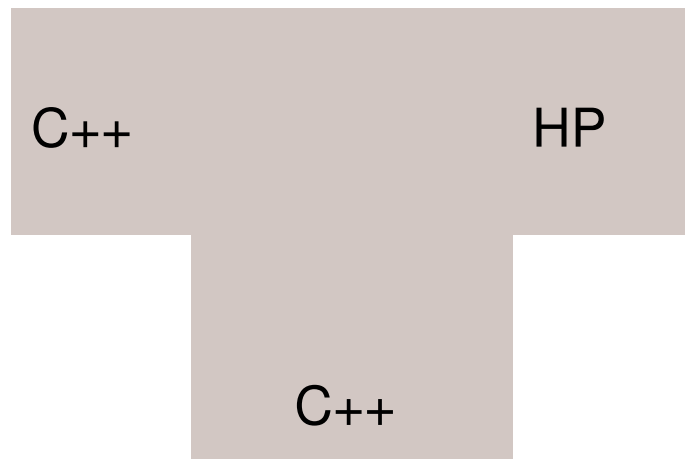


# Diagramas T

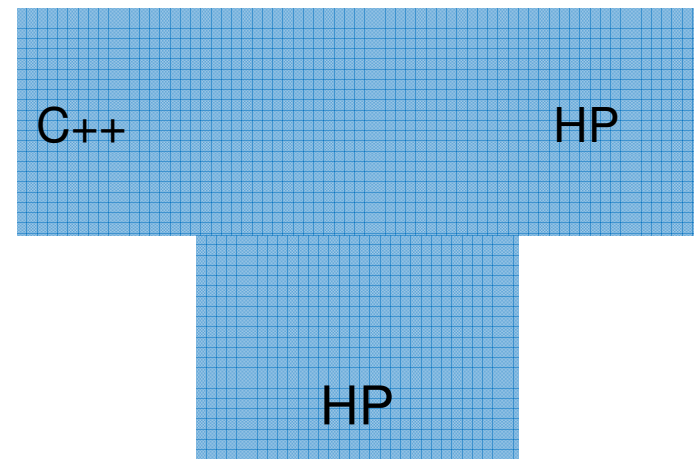
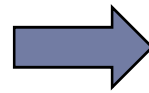
Notación para representar implementaciones

---

## Autocompilador de c++



Fuente del compilador



Ejecutable del compilador



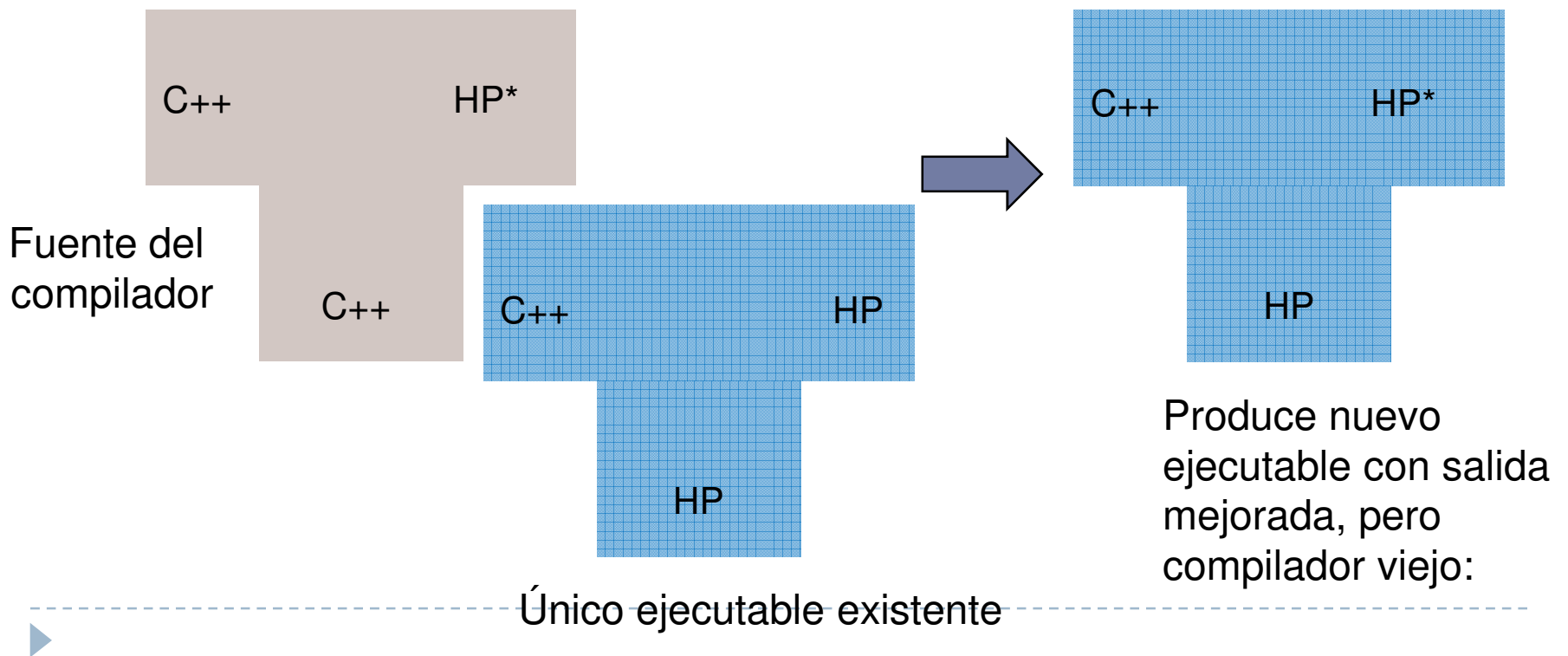
# Diagramas T

## Notación para representar implementaciones

---

- ▶ **Mejorar un autocompilador - Paso I:**

- ▶ Se modifica el fuente para que produzca salida mejor (HP\*)
- ▶ Se compila con el único ejecutable existente

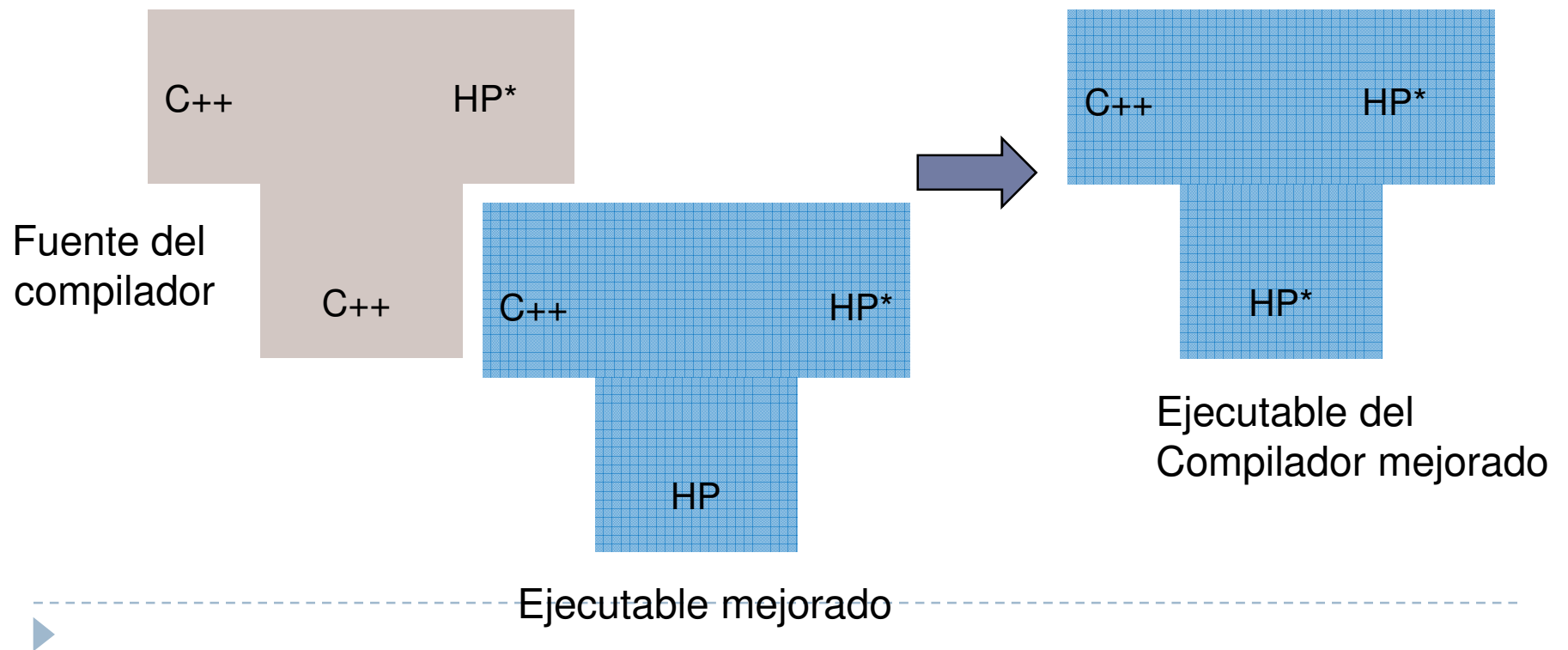


# Diagramas T

## Notación para representar implementaciones

---

- ▶ **Mejorar un autocompilador - Paso 2:**
  - ▶ Se compila con el ejecutable mejorado



# Ejercicios

---

- ▶ Considerar que se dispone de un compilador de PASCAL a ASSEMBLER de SUN escrito en “C”.
  - ▶ **¿Qué se necesita para generar, en una PC con procesador PENTIUM y Sistema Operativo WINDOWS, ejecutables para SUN?**  
**(Representar usando diagramas T)**





# Ejercicios

---

- ▶ Se tiene un cross-compiler que se ejecuta en una PC Pentium con Sistema Operativo Windows y genera código para la CPU SUN bajo el sistema operativo SOLARIS.
- ▶ **¿Este compilador puede ser simultáneamente un autocompilador?**
  - ▶ **En caso afirmativo indicar cómo se compila y bajo qué plataforma.**
  - ▶ **En caso negativo indique por qué no es posible.**

**(Representar usando diagramas T)**

