

Introducción a la computación

Electrónica IV

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

<https://facetvirtual.facet.unt.edu.ar/course/view.php?id=165>

Introducción: Temas a tratar

- ▶ Componentes de una computadora
 - ▶ Procesador, Memoria, E/S y Buses.
 - ▶ Ejecución de un programa
- ▶ Conceptos de Software
 - ▶ Lenguaje de Maquina y Lenguaje Superior
 - ▶ Ensamblador, interprete y compilador

Breve historia de la computación

- ▶ Las primeras computadoras se diseñaron para automatizar cálculos matemáticos repetitivos
- ▶ Una secuencia de instrucciones realizaba operaciones sobre datos variables almacenados en una memoria

Arquitectura Harvard



- ▶ Las primeras computadoras aplican este concepto literalmente
- ▶ Tenían una memoria para las instrucciones y otra memoria separada para los datos
- ▶ La modificación del programa se realizaba en forma externa a la computadora

Arquitectura Von Newman

- ▶ En 1945 John Von Newman propone la unificación de las memorias de datos y de instrucciones
- ▶ También divide el procesador en una unidad Aritmético-Lógica y un sistema de control
- ▶ Ahora un programa puede ser manipulado como datos por otro programa

Arquitectura de una computadora



- ▶ La memoria almacena instrucciones y datos
- ▶ El procesador ejecuta las instrucciones para transformar los datos
- ▶ La entrada/salida permite la comunicación con el mundo exterior

Interfases de Entrada/Salida

- ▶ Los dispositivos externos se controlan leyendo y/o escribiendo varios registros
- ▶ Para el programa estos registros se comportan como direcciones de memoria normales
 - ▶ Los registros de control permiten configurar el modo de funcionamiento del dispositivo
 - ▶ Los registros de estado permiten conocer las novedades sobre el funcionamiento del dispositivo
 - ▶ Los registros de datos permiten intercambiar información con el dispositivo

Entradas y Salidas Digitales

- ▶ Solo pueden adoptar dos valores lógicos (**0 y 1**) que se corresponden con dos valores de tensión (V_{ss} y V_{dd})
- ▶ **Entradas Digitales:** El valor de tensión en un terminal del dispositivo se traduce en el estado de un bit en una dirección de memoria determinada
- ▶ **Salidas Digitales:** El estado de un bit de una dirección de memoria determinada se traduce en el valor de tensión de un terminal del dispositivo

Entradas y Salidas Analógicas

- ▶ **Entradas Analógicas:** El valor de tensión en un terminal del dispositivo se traduce proporcionalmente en el valor de numérico almacenado en una dirección de memoria determinada
- ▶ **Salidas Analógicas:** El valor numérico almacenado en una dirección de memoria determinada se traduce proporcionalmente en el valor de tensión de un terminal del dispositivo
- ▶ Los conversores utilizados determinan la cantidad de bit con los cuales se representan los valores numéricos de entrada o de salida

Sistemas embebidos

- ▶ Se usan para una aplicación específica
 - ▶ Incluyen una computadora y la electrónica necesaria para resolver el problema.
 - ▶ La solución esta embebida en el sistema.
- ▶ Ejecutan siempre el mismo programa
 - ▶ No es accesible al usuario del dispositivo.
 - ▶ Resuelve un rango definido de problemas.

Sistemas de propósito general

- ▶ Las identificamos como computadoras
 - Procesadores como Pentium, Athlon, etc.
- ▶ Ejecución de diversos programas
 - Herramientas como Word, Excel, Chrome
- ▶ Mucha potencia de cálculo
 - Consumo de potencia mayor que en un μC

Instrucciones y datos

- ▶ Ejemplo de datos
 - ▶ Posición A contiene el dato X
 - ▶ Posición B contiene el dato Y
- ▶ Ejemplo de instrucciones
 - ▶ Tome X de la posición A
 - ▶ Sume el valor Y de la posición B
 - ▶ Guarde el resultado en la posición C
- ▶ Las instrucciones del programa se pueden escribir en diferentes lenguajes

Lenguaje de máquina

- ▶ Es el lenguaje nativo del procesador, esta conformado por las Instrucciones del ISA
 - ▶ Tal como son cargadas en memoria y leídas por el CPU
 - ▶ Expresados como cadenas binarias de '0' y '1'
- ▶ Las primeras CPUs se programaban así
 - ▶ Propenso a errores: basta con cambiar un 1 por un 0
 - ▶ Ilegible e Indocumentable
 - ▶ Requiere mucho tiempo de las personas
 - ▶ Al principio lo caro eran las computadoras
- ▶ Escribir en Hexadecimal es un primer avance

Lenguaje ensamblador

- ▶ Conformado por las instrucciones del ISA
 - ▶ Pero escritas en código mnemotécnico
 - ▶ Correspondencia uno a uno con lenguaje de máquina
 - ▶ Es necesario traducirlas antes de ejecutarlas
- ▶ El traductor se llama ensamblador
 - ▶ Tarea mecánica realizada por un programa traductor
 - ▶ Incorpora también símbolos para variables y rótulos
 - ▶ Acepta comentarios y directivas al ensamblador para modificar el resultado de la traducción

Lenguaje ensamblador

- ▶ La sentencia **add a,b,c** calcula la operación $a = b + c$
- ▶ En esta sentencia a, b y c son los operandos y pueden ser variables y/o constantes
- ▶ Los operandos pueden estar almacenados en registros o en memoria

Programa en ensamblador

```
.text                # Define la sección de programa
.globl main          # Define el punto de entrada del código

main:
    la  s0, dos       # Se apunta un registro al bloque de datos
    lb  t0, 0(s0)      # Se carga un registro con una variable
    lb  t1, 1(s0)      # Se carga otro registro con otra variable
    add t2, t0, t1     # Se calcula la suma de los dos registros
    sb  t2, 2(s0)      # Se almacena el resultado en una variable
lazo: j    lazo        # Se repite un autolazo indefinidamente

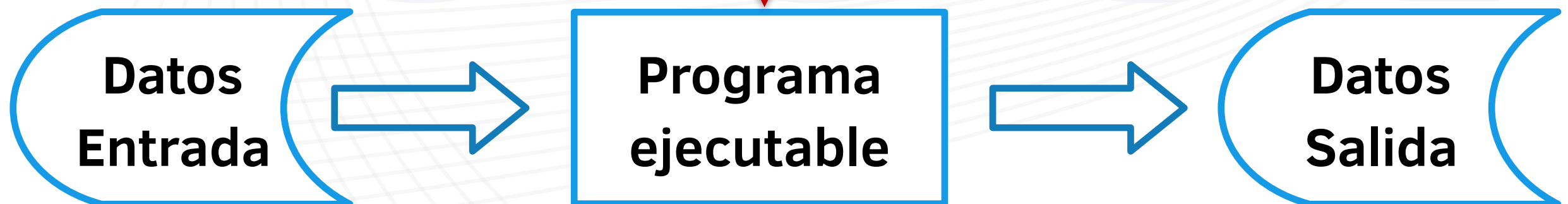
.data                # Define la sección de los datos
dos:  .byte 2         # Inicializa una variable con el valor dos
tres: .byte 3         # Inicializa una variable con el valor tres
res:  .space 1        # Reservar espacio para una variable
```

Ensamblado y ejecución

Traducción del código fuente a código objeto



Ejecución del código objeto en el procesador



Ventajas y desventajas

▶ Desventajas

- ▶ Distintas máquinas tienen diferentes lenguajes
- ▶ Está orientado a la máquina y no al usuario
- ▶ Hay que ser un experto
- ▶ Difícil desarrollar mercados...

▶ Ventajas

- ▶ Manejo exacto de tiempo y memoria
- ▶ Para aplicaciones críticas en tiempo real

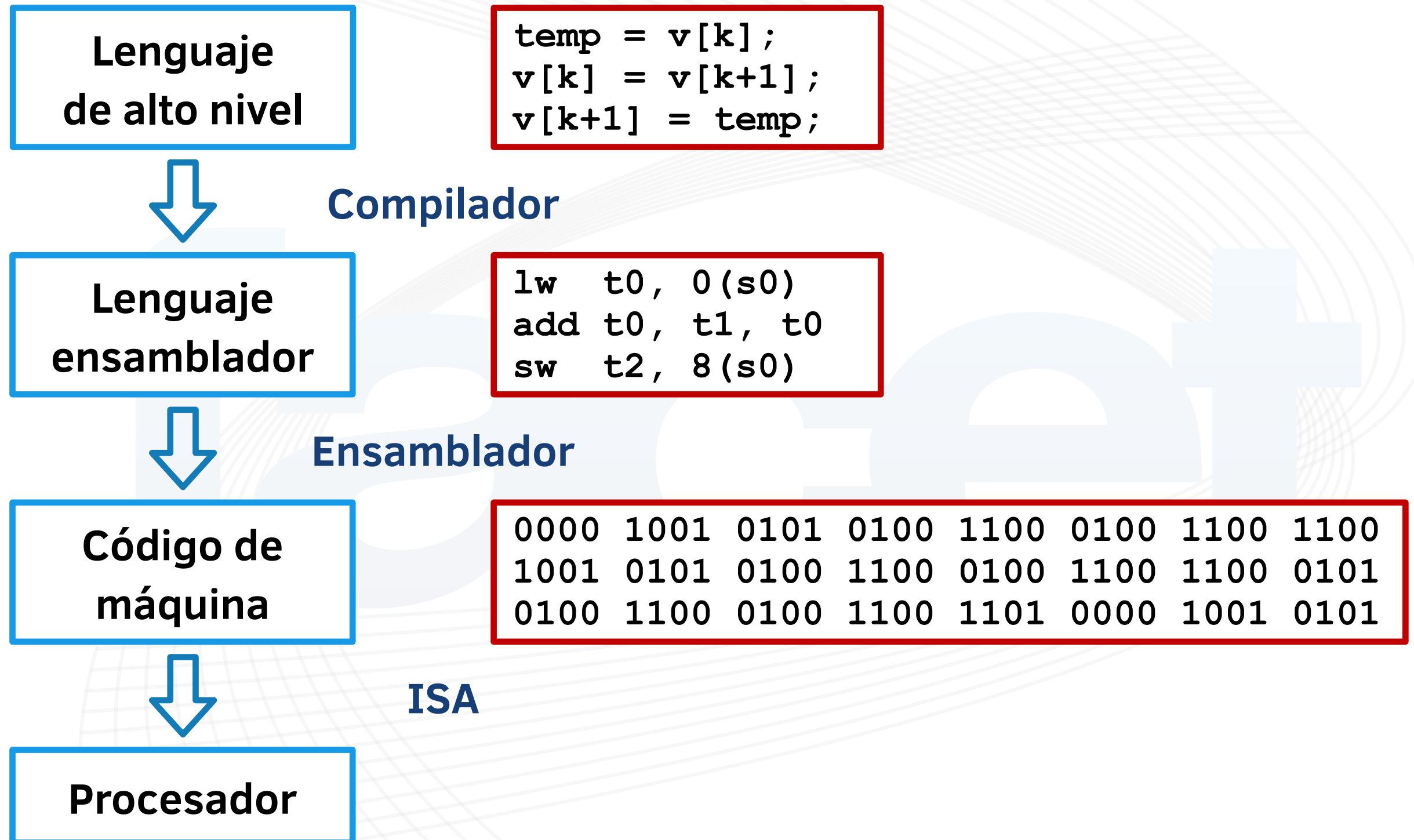
Lenguaje superior

- ▶ Son lenguajes orientados a resolver problemas
 - ▶ Están pensados para los usuarios
 - ▶ Cada sentencia se traduce en varias instrucciones del lenguaje de máquina
 - ▶ Son independientes de la máquina para la cual desarrolla el programa
- ▶ Dos tipos de traductores diferentes

Lenguajes compilados

- ▶ Se traduce todo el programa a la vez
- ▶ Se almacena una versión ejecutable del programa completo
- ▶ La traducción no es única y puede optimizarse
- ▶ Un cambio en el programa fuente implica una nueva compilación
- ▶ Resulta más difícil depurar errores, sobre todo si el ejecutable está optimizado

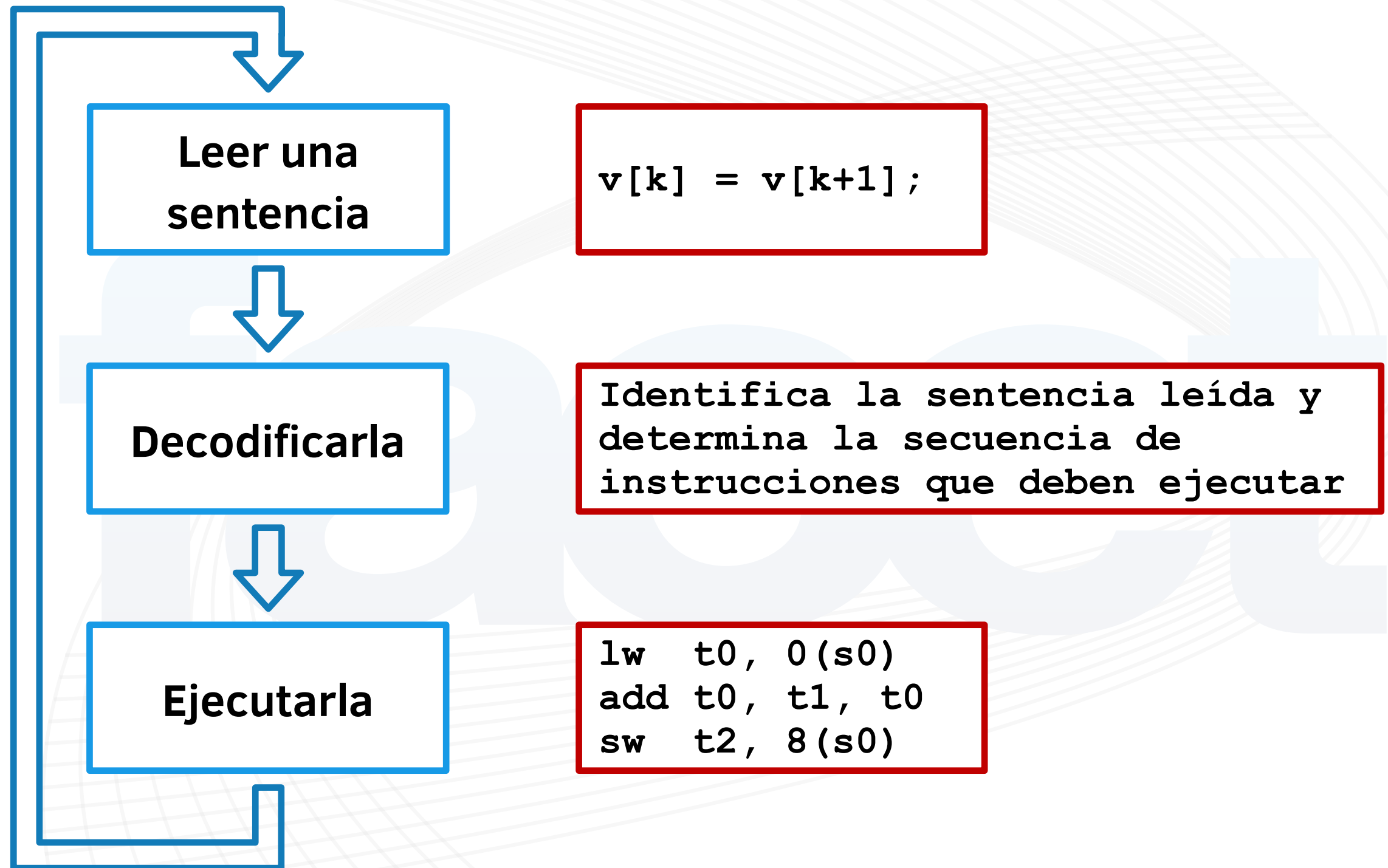
Funcionamiento del compilador



Lenguajes interpretados

- ▶ La traducción se realiza en tiempo de ejecución, tomando las sentencias de a una
- ▶ No se almacena una versión ejecutable del programa
- ▶ Es muy difícil optimizar porque no se conoce el programa completo
- ▶ Se pueden realizar cambios en el programa fuente mientras se ejecuta
- ▶ Es mucho más sencillo depurar errores

Funcionamiento del interprete



Sistemas Operativos

- ▶ Gestionan en forma eficiente los recursos de una computadora
 - Permiten compartir recursos
 - Definen mecanismos de seguridad
- ▶ Facilitan el uso de las mismas
 - Facilitan a los programadores interactuar con la computadora y sus periféricos
 - Facilitan al usuario la ejecución de los diferentes programas

Niveles de abstracción

