



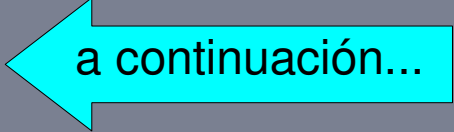
Unidad 2 – Clase 5

Sistemas Operativos



Lenguajes de programación

- Niveles de lenguajes (alto y bajo)

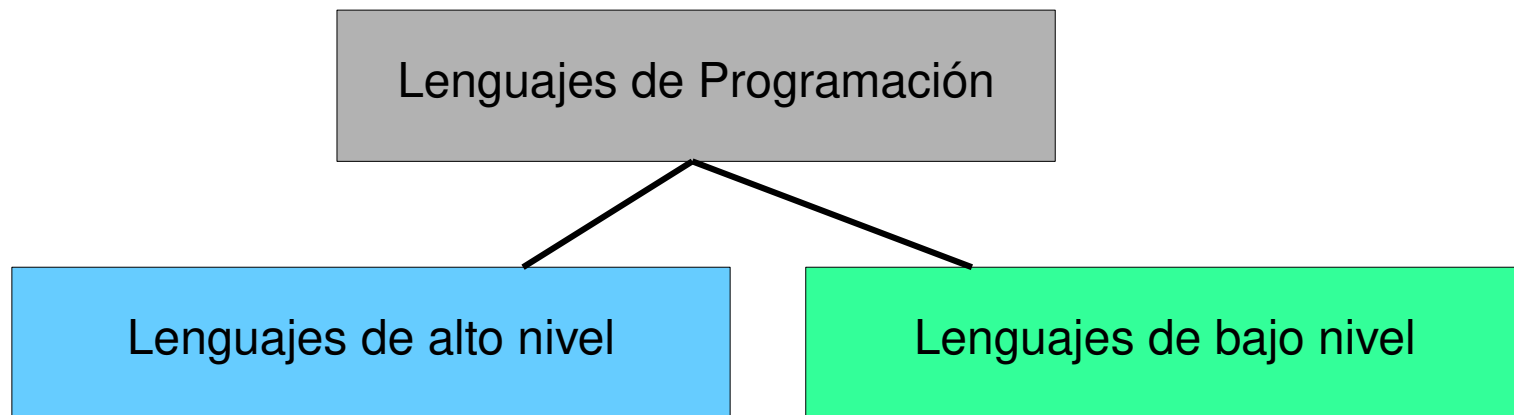


a continuación...

- Traductores

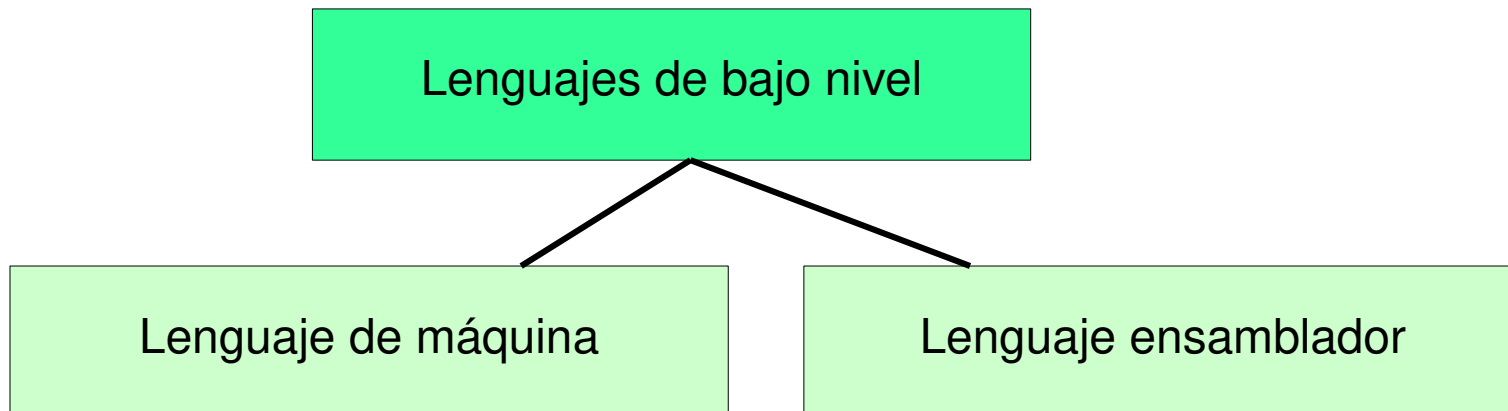
Lenguajes de programación

- Los lenguajes de programación especifican reglas para escribir programas.
- Hay principalmente dos tipos de lenguajes de programación:



Lenguajes de bajo nivel

- Se denominan de “bajo nivel” por la reducida abstracción existente entre el lenguaje y el hardware.
- Sus instrucciones ejercen un control directo sobre el hardware. Las instrucciones son dependientes de la máquina.
- Clasificación:



Lenguaje de Máquina

- Es el lenguaje que entiende la CPU, es decir, conformado por instrucciones del ISA.
- Un programa escrito en este lenguaje es ejecutado directamente por la CPU.
- Escribir un programa en este lenguaje y depurarlo (corregir sus errores) son tareas muy difíciles porque todo está en binario (las instrucciones y los datos) y es fácil equivocarse.

Lenguaje Ensamblador

- Es una representación simbólica del lenguaje de máquina.
- Normalmente, una instrucción de lenguaje ensamblador es implementada por una única instrucción en lenguaje de máquina.
- El lenguaje ensamblador utiliza:
 - Mnemónicos en lugar de códigos binarios de operaciones. Ejemplo: `ADD` para indicar operación de suma.
 - Identificación de registros en decimal: `R1`, `R2`, ...
 - Etiquetas en lugar de direcciones de memoria.
Ejemplo: `LOAD R1, cantidad`
 - Los datos numéricos se pueden escribir en decimal u otras bases. El texto se puede escribir utilizando caracteres.

Lenguaje Ensamblador

Lenguaje de máquina

```
01000111
11100100
01111111
10100110
11011101
00100000
00000001
00000011
```

Lenguaje ensamblador

Etiqueta	Mnemónico	Argumento
----------	-----------	-----------

	LD	CANT
SIGUE:	JZ	FIN
	ST	OUT
	SUB	UNO
	JMP	SIGUE
FIN:	HLT	
UNO:	1	
CANT:	3	

Lenguaje Ensamblador

Ensamblador x86

```
.globl _start
.text                # seccion de codigo
_start:
    movl    $len, %edx        # carga parametros longitud
    movl    $msg, %ecx        # y direccion del mensaje
    movl    $1, %ebx          # parametro 1: stdout
    movl    $4, %eax          # servicio 4: write
    int     $0x80             # syscall

    movl    $0, %ebx          # retorna 0
    movl    $1, %eax          # servicio 1: retorno de llamada
    int     $0x80             # syscall

.data                # seccion de datos
msg:
    .ascii  "Hola, mundo!\n"
len = . - msg           # longitud del mensaje
```



Lenguaje Ensamblador

Ensamblador ARM

```
.global main
main:
    @ Guarda la direccion de retorno lr
    @ mas 8 bytes para alineacion
    push    {ip, lr}
    @ Carga la direccion de la cadena y llama syscall
    ldr     r0, =hola
    bl      printf
    @ Retorna 0
    mov     r0, #0
    @ Desapila el registro ip y guarda
    @ el siguiente valor desapilado en el pc
    pop     {ip, pc}
hola:
    .asciz "Hola, mundo!\n"
```



Lenguaje Ensamblador

Ensamblador PowerPC

```
.data                # seccion de variables
msg:
    .string "Hola, mundo!\n"
    len = . - msg    # longitud de cadena
.text                # seccion de codigo
    .global _start
_start:
    li 0,4           # syscall sys_write
    li 3,1           # 1er arg: desc archivo (stdout)
                        # 2do arg: puntero a mensaje
    lis 4,msg@ha      # carga 16b mas altos de &msg
    addi 4,4, msg@l    # carga 16b mas bajos de &msg
    li 5,len          # 3er arg: longitud de mensaje
    sc               # llamada al kernel

#
    li 0,1           # syscall sys_exit
    li 3,1           # 1er arg: exit code
    sc               # llamada al kernel
```



Utilidad de lenguajes de bajo nivel

- Un lenguaje de **bajo nivel** da al programador un **control total** sobre lo que puede hacer con el procesador. Por este motivo, son elegidos para el desarrollo de software especial que requiere ejecución con bajo consumo de recursos, elevado rendimiento computacional y bajo consumo energético. Ej: algunas partes del kernel, algunas aplicaciones en sistemas embebidos, o de computación de alto rendimiento.
- Son portables solo entre máquinas con igual ISA y S.O.
- Escribir un programa para resolver un problema complejo en un lenguaje de bajo nivel suele ser muy costoso en tiempo y esfuerzo. Además, el programador deberá conocer las particularidades del procesador mediante la lectura de su extensa documentación.

Lenguajes de alto nivel

Tienen un alto nivel de abstracción, que aíslan al programador de entender el funcionamiento de un procesador (y sistema operativo) particular, para permitirle enfocarse en la resolución de un problema en un dominio particular. Es decir, están **orientados a la resolución de ciertos tipos de problema**.

- Permiten especificar operaciones para resolver problemas en forma más parecida al lenguaje humano, matemático, etc. Ejemplos: matlab, lenguajes de consultas de bases de datos, etc.

Lenguajes de alto nivel

Una instrucción de alto nivel podría requerir cientos o miles de instrucciones de bajo nivel para resolverse.

Un ejemplo muy simple (por razones de espacio):

- **Código de alto nivel**

DEUDA = DEUDA - PAGO

} 1 instrucción

- **Código ensamblador**

LD DEUDA

SUB PAGO

ST DEUDA

} 3 instrucciones

- **Código de máquina**

01001010

01101011

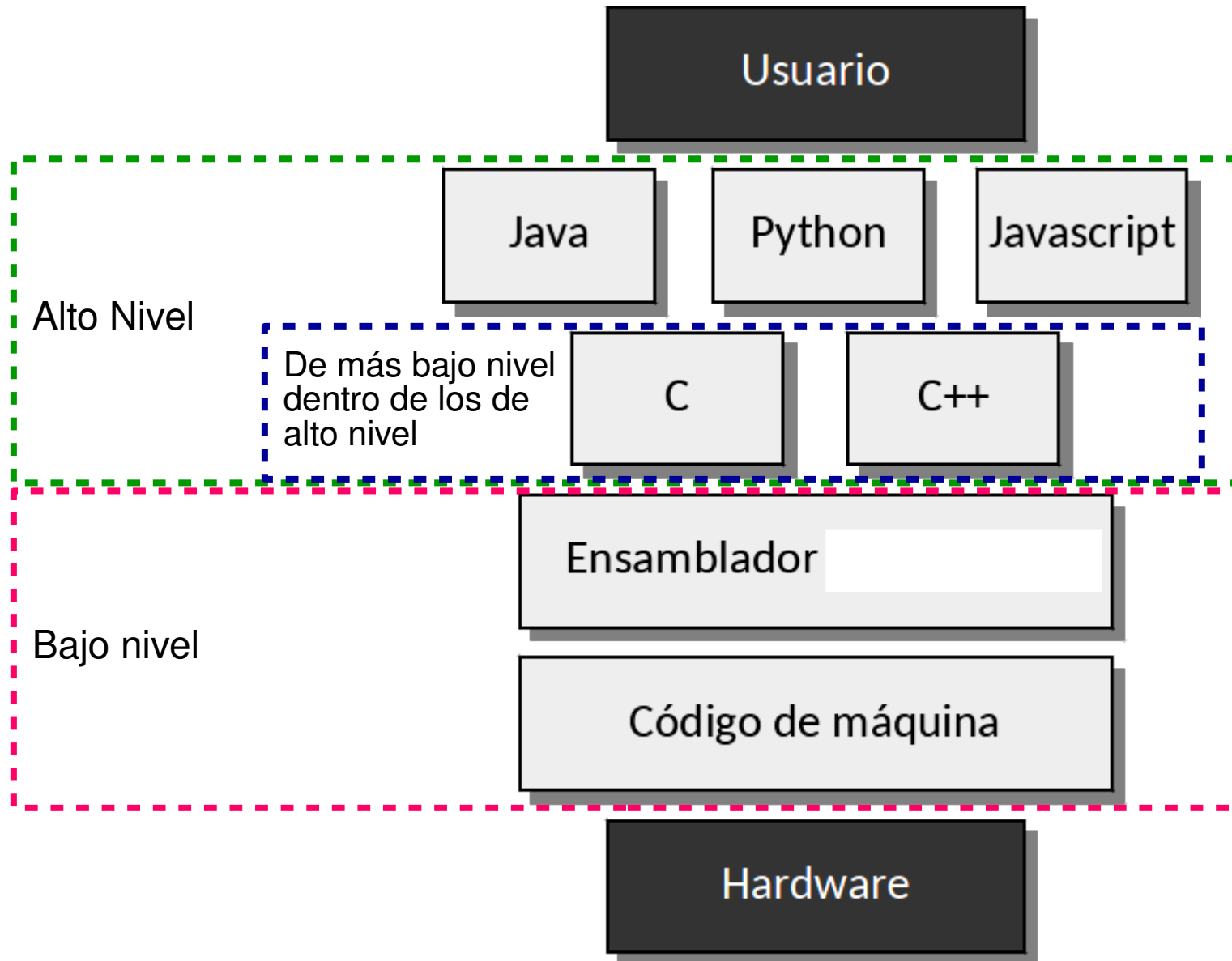
10101010

} 3 instrucciones

Utilidad del lenguajes de alto nivel

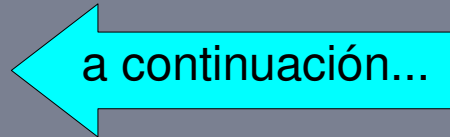
- El programador alcanza una alta productividad debido a que se le simplifica la programación de problemas de software complejos y extensos.
- Pueden ser más portables (ver detalles de compiladores e intérpretes).
- Su depuración (el proceso de corregir errores de programación) es normalmente más simple.
- La funcionalidad provista por el lenguaje podría impedir que el programador utilice cierta funcionalidad específica de un procesador, impidiendo aprovechar su máximo potencial. Debido a esto, algunos lenguajes como C, permiten incrustar código de lenguaje ensamblador en el código fuente.

Ejemplos:

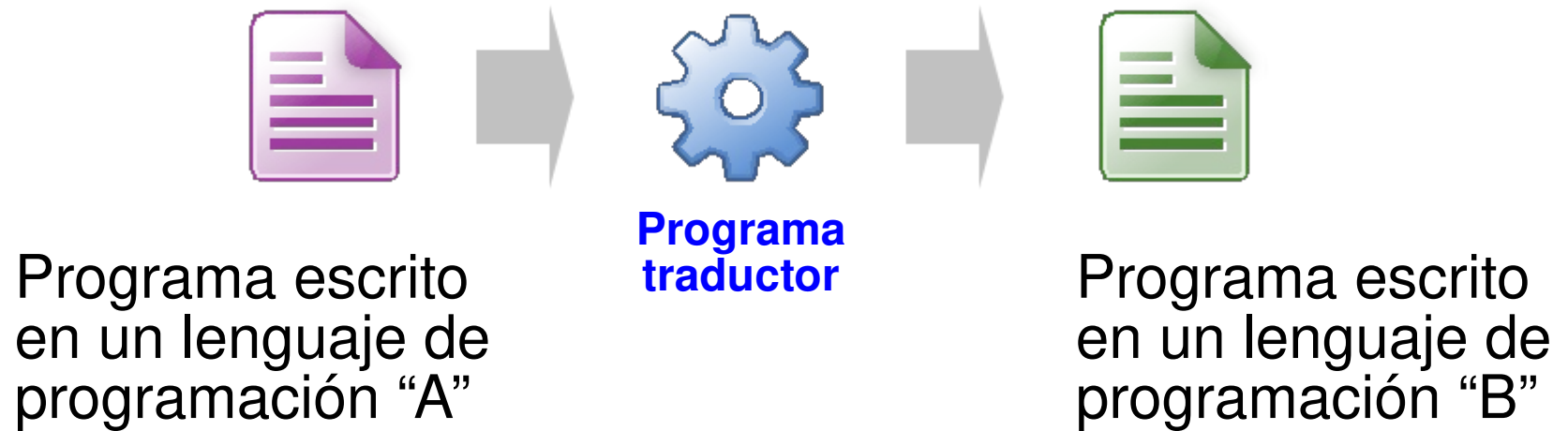


Lenguajes de programación

- Nivel de lenguajes (alto y bajo)
- Traductores



Traductores



Traductores de lenguajes de **bajo** nivel

El ensamblador es un traductor que convierte un programa escrito en lenguaje ensamblador a un programa escrito en lenguaje de máquina.

Programa
escrito en un
lenguaje
ensamblador
(**código
ensamblador**)

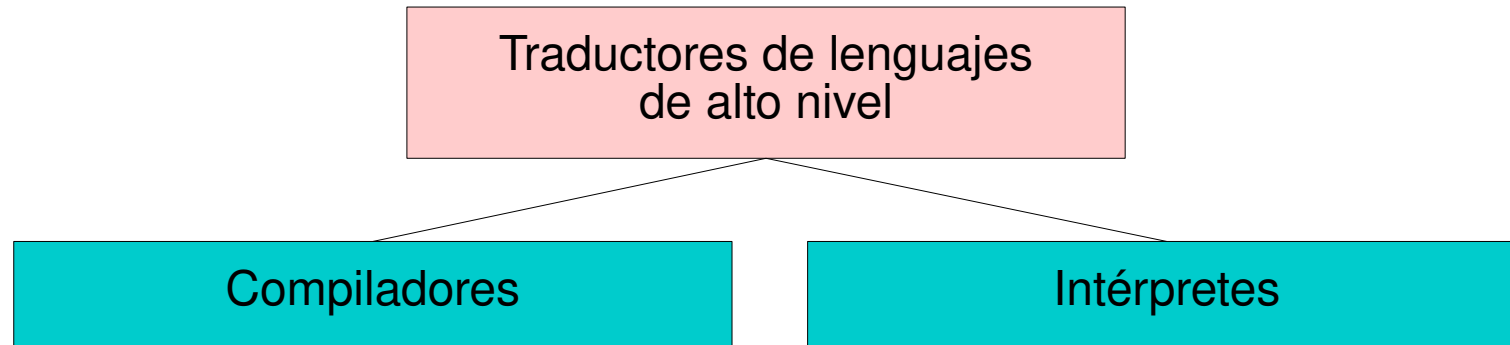


Ensamblador



Programa
escrito en un
lenguaje de
máquina
(**código de
máquina o
binario**)

Traductores de lenguajes de **alto nivel**



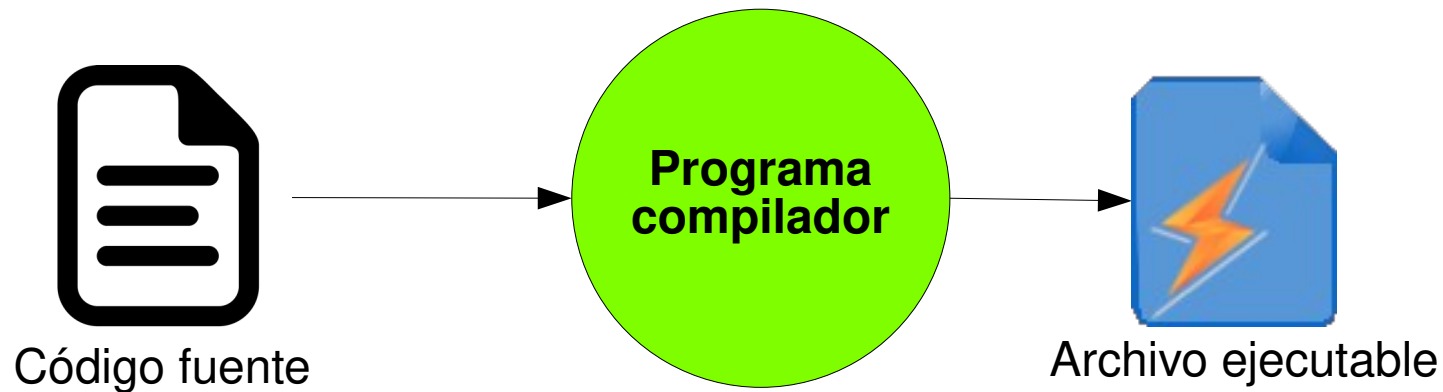
Los compiladores e intérpretes convierten código escrito en lenguaje de alto nivel a lenguaje de máquina.



- Posiblemente, un compilador/intérprete utilice un ensamblador para hacer parte de su trabajo

Compilación

Un **compilador** realiza la traducción (del código fuente) y almacena el resultado (código de máquina) en un archivo ejecutable. La traducción se hace una sola vez, y luego se ejecuta el programa tantas veces se quiera.



Etapas del proceso de Compilación

Traducción/Compilación: Se genera un código objeto, que es un código de máquina que corresponde a la traducción del código fuente. Si se producen errores, el programador debe leer los mensajes de error y corregir el archivo fuente.

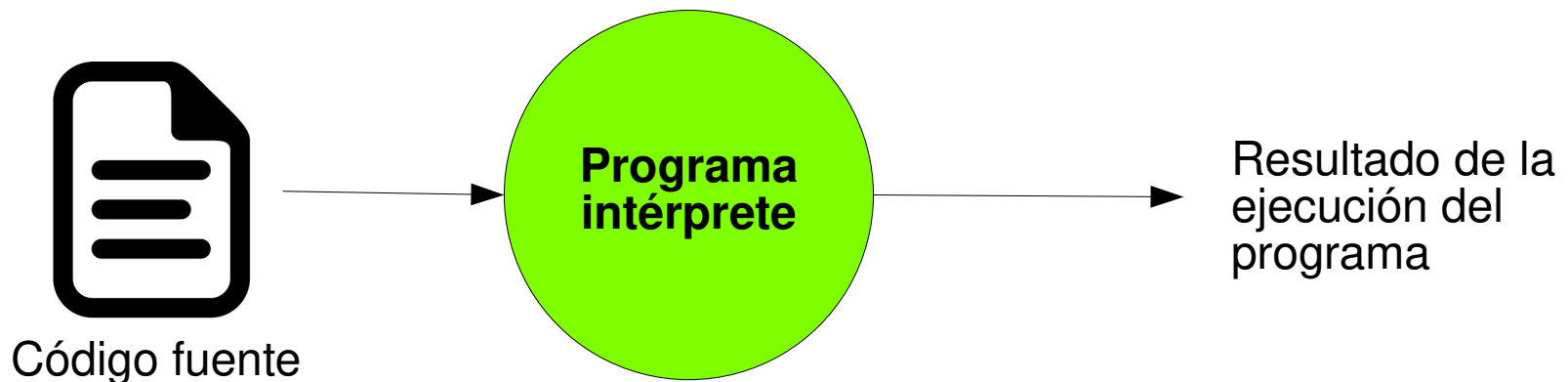


Vinculación (o enlazado): El programador, al escribir el código fuente, podría hacer uso de rutinas o funciones que vienen provistas por el lenguaje, y no necesita especificar cómo se realizan esas funciones. Al no aparecer en el programa fuente, esas funciones no aparecerán en el archivo objeto. La etapa de vinculación se ocupa de enlazar el archivo objeto con código de máquina faltante y logra así obtener el programa ejecutable.

Interpretación

Cuando un programa (escrito en un lenguaje interpretado) requiere ser ejecutado, el intérprete realiza la traducción de las instrucciones y las envía a ejecutar.

- El código de máquina no se almacena en un archivo sino que directamente se envía a ejecutar. Al finalizar la ejecución del intérprete, habrá finalizado la ejecución de nuestro programa.



Traductores de lenguajes de alto nivel

- Intérprete:

- Es portable: el programa puede ser ejecutado directamente en cualquier computadora para la cual existe el programa intérprete (posiblemente con diferente hardware y sistema operativo).
- Bajo rendimiento: la CPU no está dedicada exclusivamente a ejecutar el programa sino también a la traducción del código fuente.

- Compilador:

- No es portable: el programa solo puede ser ejecutado en computadoras con el mismo ISA y Sistema Operativo.
- Alto rendimiento: La ejecución del programa es rápida porque la traducción ya estaba hecha antes de ejecutarse el programa.

Bibliografía

- <http://es.wikipedia.org/>
- Capítulo 1 de “Fundamentos de Sistemas Operativos”.
Silberschatz, Galvin, Gagne.