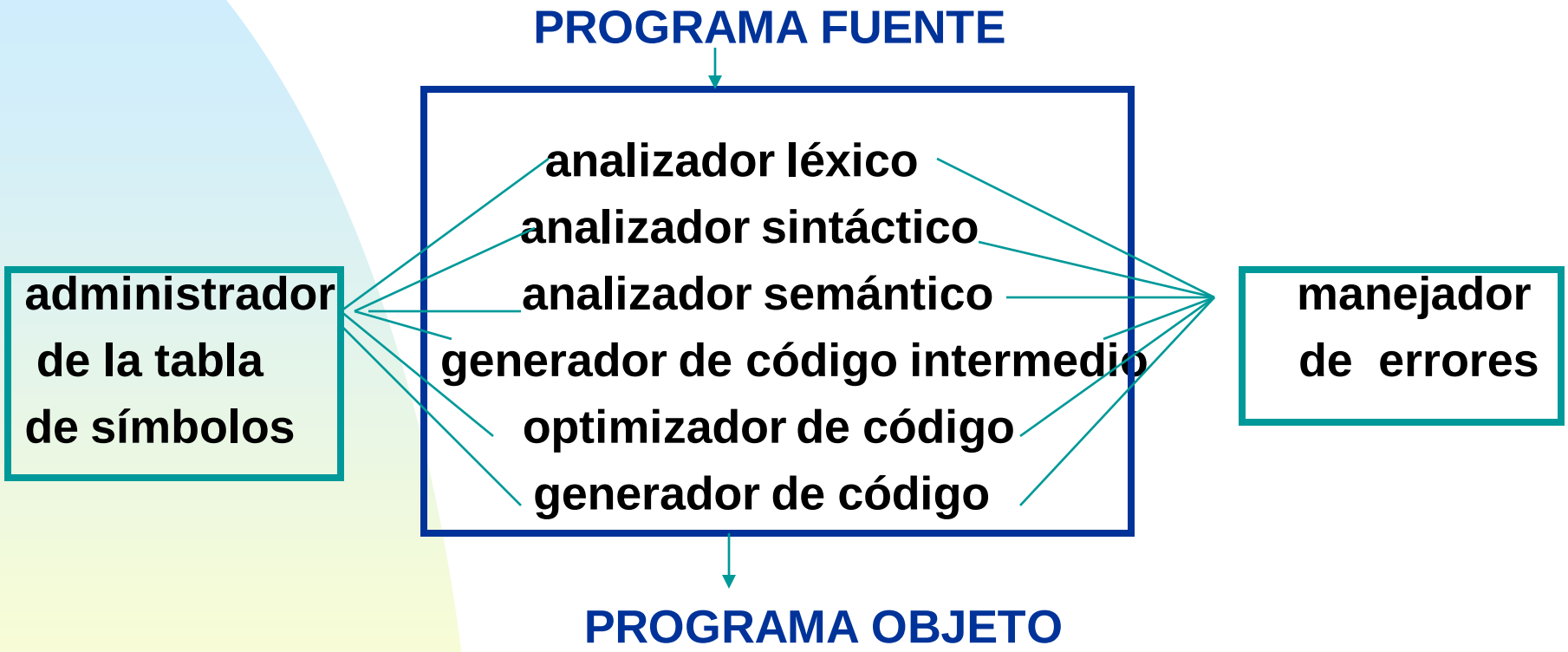
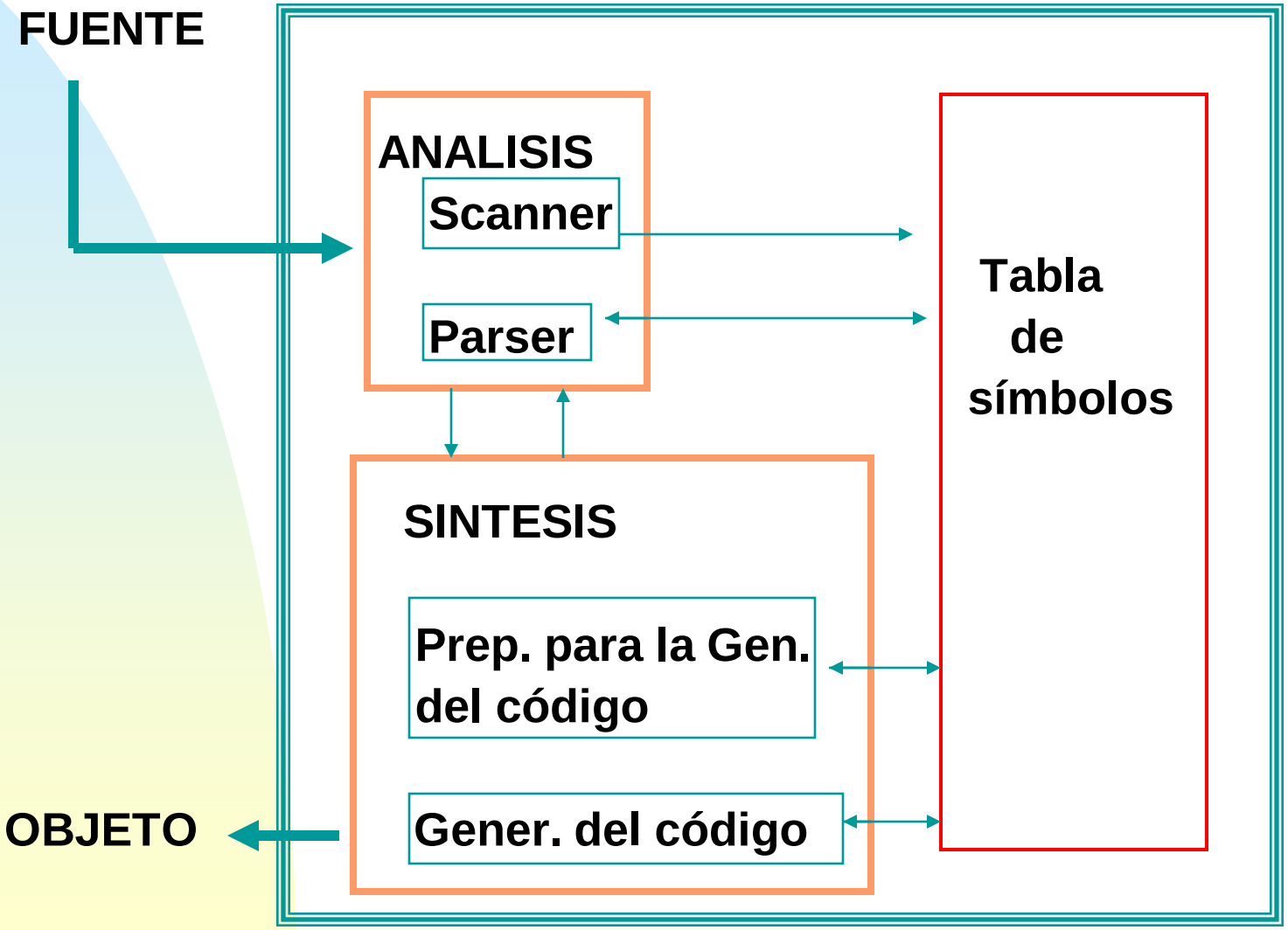
FASES DE UN COMPILADOR

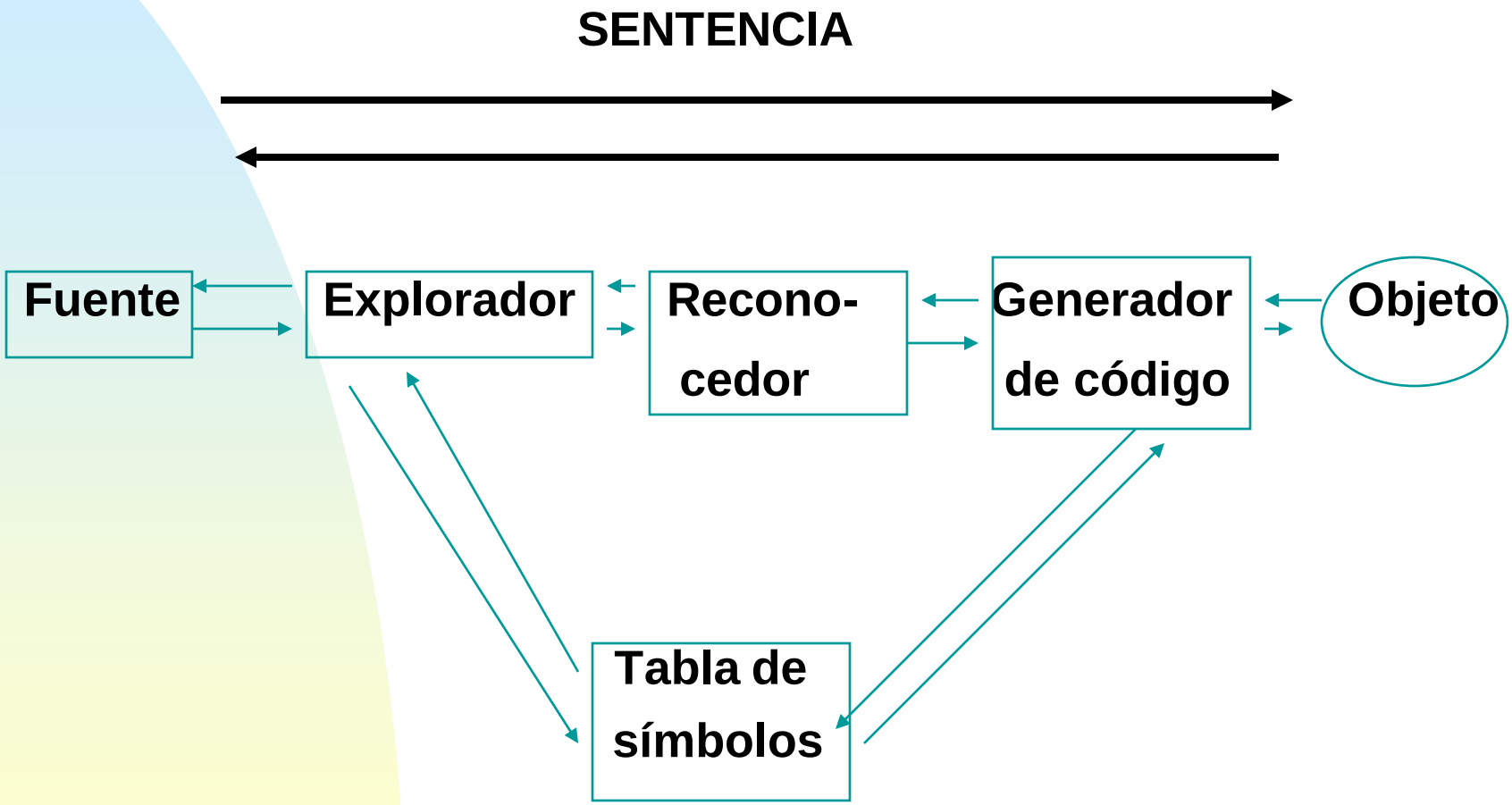


Cada fase transforma al PF de una representación a otra

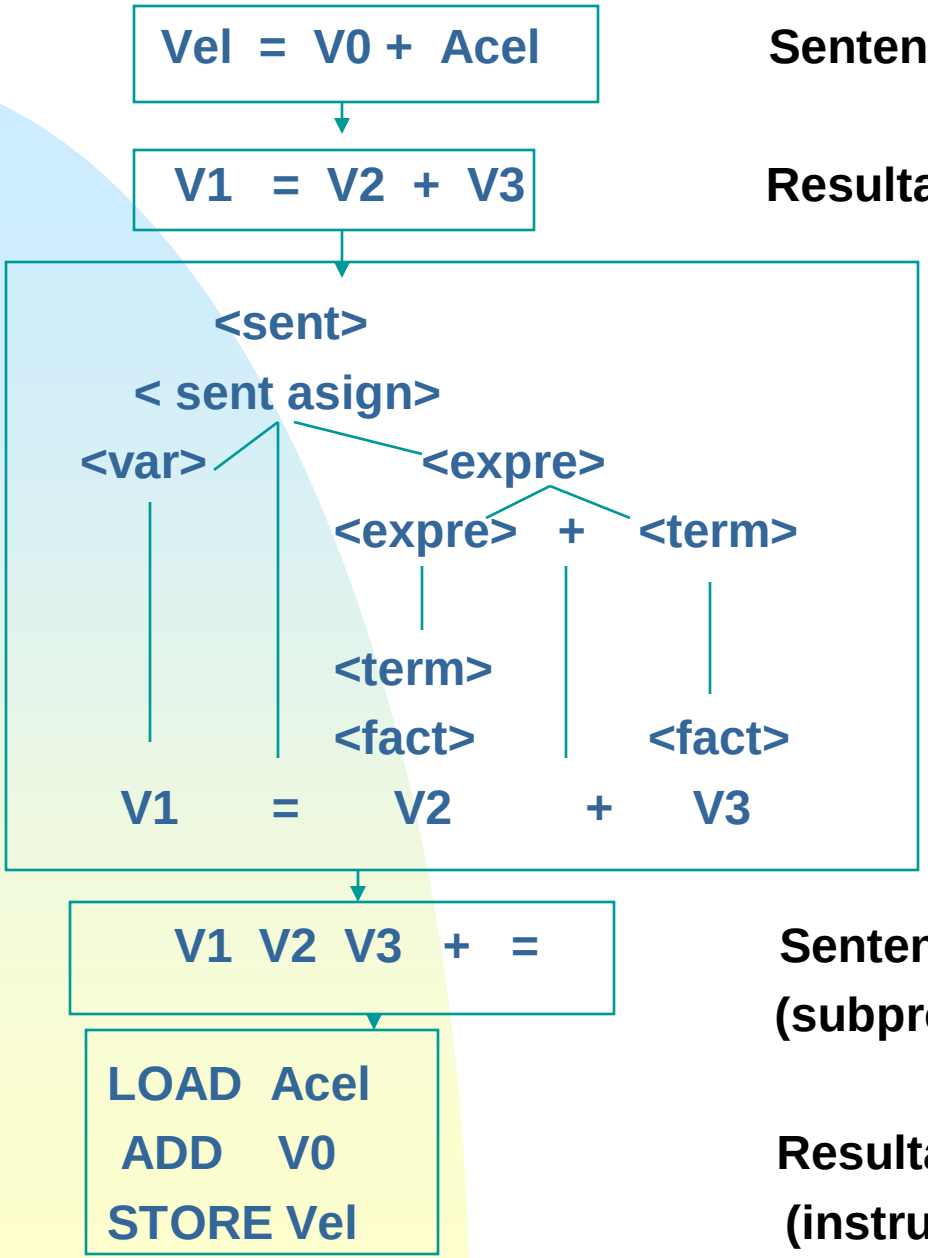
ESQUEMA DE BLOQUES DE UN COMPILADOR



ESTRUCTURA FUNCIONAL DE UN COMPILADOR
(de una pasada)



COMPILACIÓN DE UNA SENTENCIA EJEMPLO



Sentencia fuente a compilar

Resultado del EXPLORADOR

RECONOCEDOR:

Análisis sintáctico:
la sentencia es correcta

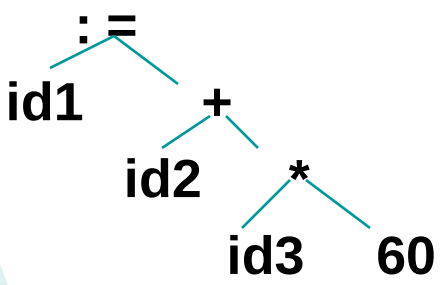
Sentencia en notación polaca
(subproducto del reconocedor)

Resultado del GEN. DE CÓDIGO
(instrucciones para máquina)

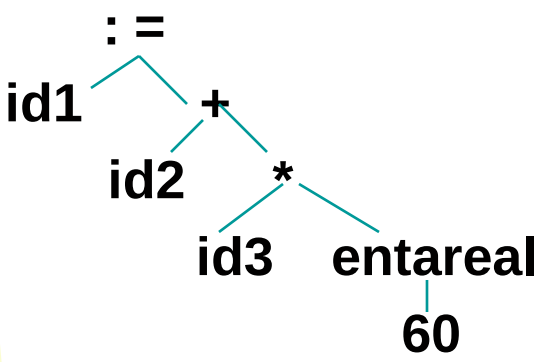
posicion := inicial + velocidad * 60

- A. Lex: id1 : = id2 + id3 * 60

- A. Sint:



- A. Seman:



ADMINISTRADOR DE LA TABLA DE SÍMBOLOS

- **TABLA DE SÍMBOLOS**

  Estructura de datos que contiene un registro por cada identificador, con los campos para los atributos:

  -- Información sobre la memoria asignada

  -- tipo

  -- Si es nombre de procedimiento (número, tipo y método de paso de cada argumento)

- **Permite encontrar rápidamente cada ID y almacenar o consultar datos de ese registro**

- **En el Análisis Léxico se detectan los ID y se introducen en la Tabla de Símbolos**

- **Las fases restantes introducen información sobre los ID y después la utilizan**

DETECCIÓN E INFORMACIÓN DE ERRORES

- **Cada fase puede encontrar errores y debe tratarlo para continuar con la Compilación, permitiendo detectar más errores**

- **Las fases de Analisis Sintáctico y Semántico manejan la mayoría de los errores**

- **En el Anáilsis Semántico se detectan errores donde la estructura sintáctica es correcta pero no tiene significado la operación**

  (Por. ej. sumar dos ID , donde uno es el nombre de una matriz y el otro un nombre de procedimiento)

GENERACIÓN DE CÓDIGO INTERMEDIO

- **Se genera una representación intermedia explícita del PF**
- **Esta representación debe ser fácil de producir y de traducir al programa objeto**
- **Una de ellas es el *"código intermedio de 3 direcciones"***
  (Cada posición de memoria puede actuar como registro)
  (Cada instrucción tiene como máximo 3 operandos)
- **Ejemplo**

```
t1 := entareal (60)
t2 :=  id3 + t1
t3 := id2  + t2
id1 := t3
```

OPTIMIZACIÓN DE CÓDIGO

- **Trata de mejorar el código intermedio para que resulte un código de máquina más rápido de ejecutar**

- **En el ejemplo:** **t1 : = id3 * 60.0**
  id1 : = id2 + t1

  La conversión a real se hace en compilación
  No necesita t2 ni t3.

- ***Compiladores optimizadores*** **: La fase de optimación ocupa una parte significativa del tiempo del compilador**

- **Hay optimaciones sencillas que mejoran el tiempo de ejecución del programa sin retardar mucho la compilación**

GENERACIÓN DE CÓDIGO

- La fase final genera ***código objeto*** (en general código de máquina recolalizable o código ensamblador)

- Se seleccionan las posiciones de memoria para las variables usadas por el programa .

  Se traduce cada una de las instrucciones intermedias a una secuencia de instrucciones de máquina

- Un aspecto decisivo es la asignación de variables a registros.

- En el ejemplo, utilizando los registros 1 y 2:

```
MOV    id3, R2
MUL    %  60.0, R2
MOV    id2, R1
ADD    R2, R1
MOV    R1, id1
```

PROGRAMAS RELACIONADOS CON UN COMPILADOR

- **PREPROCESADORES (producen la entrada para un comp.)**
  - Procesamiento de Macros
  - Inclusión de archivos
  - Preprocesadores " racionales" (*estruct. de control)*
  - Extensiones a lenguajes (*bases de datos)*

- **ENSAMBLADORES**

  Producen código ensamblador que se pasa a un ensamblador para su procesamiento (versión mnemotécnica del código de máquina: nombres de operaciones y nombres de direcciones de memoria)

- **ENSAMBLADO DE DOS PASADAS (lecturas del archivo IN)**
  - *Primera:* Identificadores - Tabla de símbolos
  - *Segunda*: Traduce códigos de operaciones e identificadores
    El resultado es código de maquina relocalizable

- **CARGADORES Y EDITORES DE ENLACE**

  Modifica las direcciones relocalizables y ubica en memoria.
  Forma un solo prog. desde varios archivos relocalizables

AGRUPAMIENTO DE FASES EN LA IMPLEMENTACION

ETAPA INICIAL Y ETAPA FINAL

Inicial : **Fases que dependen del lenguaje fuente**
Hasta cierta optimación

Final : Partes que dependen de la maq. objeto y del leng. intermedio

PASADAS

Se agrupan las actividades de varias fases en una misma pasada (lectura de un archivo de entrada y escritura de un archivo de salida)

REDUCCION DEL NUMERO DE PASADAS

Pocas pasadas --> Varias fases dentro de una pasada --> Prog. completo en memoria en representación intermedia Fusión de código intermedio y objeto

ALGUNOS TIPOS ESPECIALES DE COMPILADORES

- **COMPILE- LINK- GO**

  Se compilan segmentos por separado y luego se montan todos los objetos producidos en un módulo cargable listo

- **COMPILADOR DE VARIAS PASADAS**

  No es más lento. Ocupa poca memoria. Fácil de mantener

- **COMPILADOR INCREMENTAL (o interactivo)**

  Se pueden compilar solo las modificaciones

- **AUTOCOMPILADOR**

  Comp. escrito en el propio leng. que traduce. Portabilidad.

- **METACOMPILADOR**

  Programa al que se le especifica el lenguaje para el que se quiere un comp. y produce el comp. como resultado

- **DECOMPILADOR**

  Traduce de código máquina a leng. de alto nivel