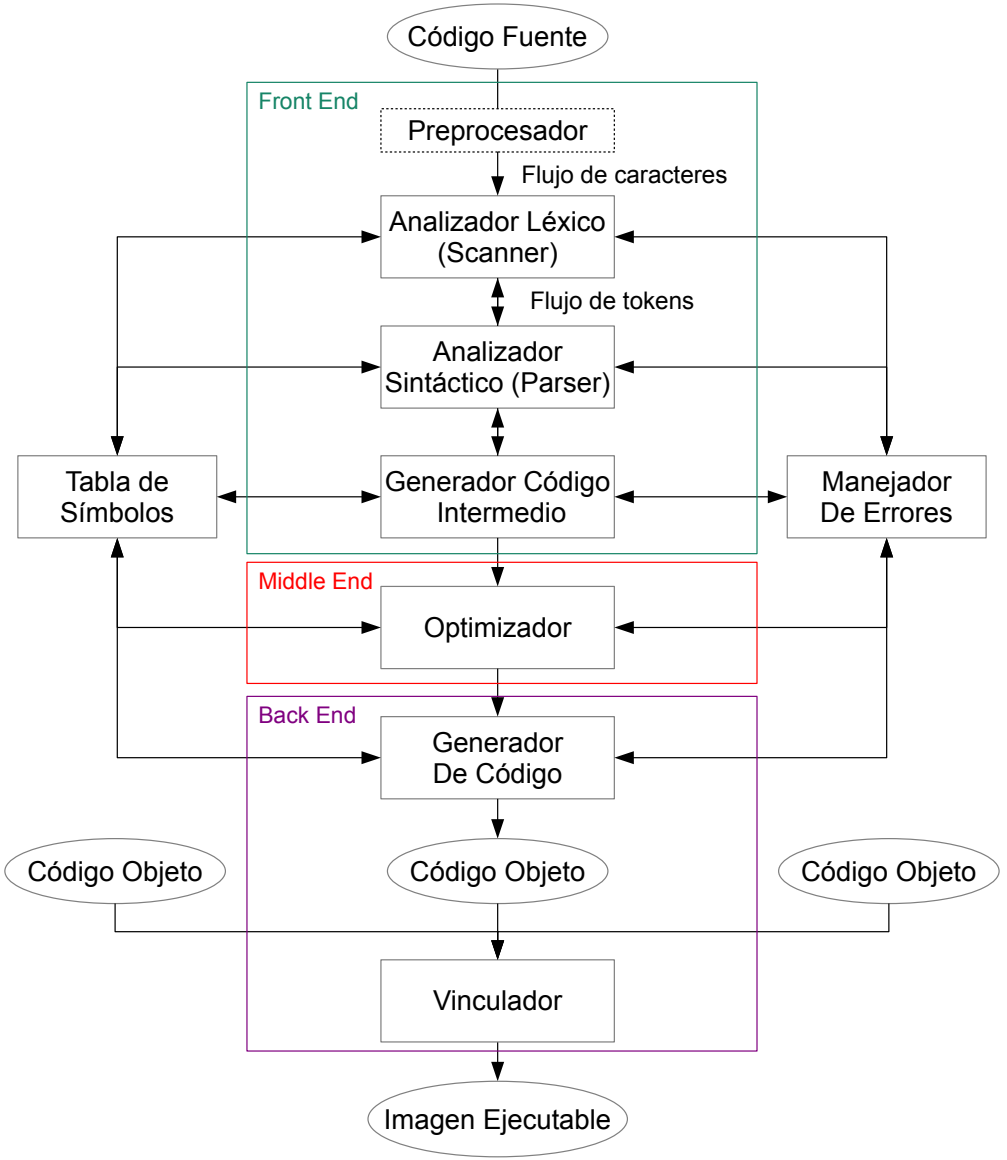
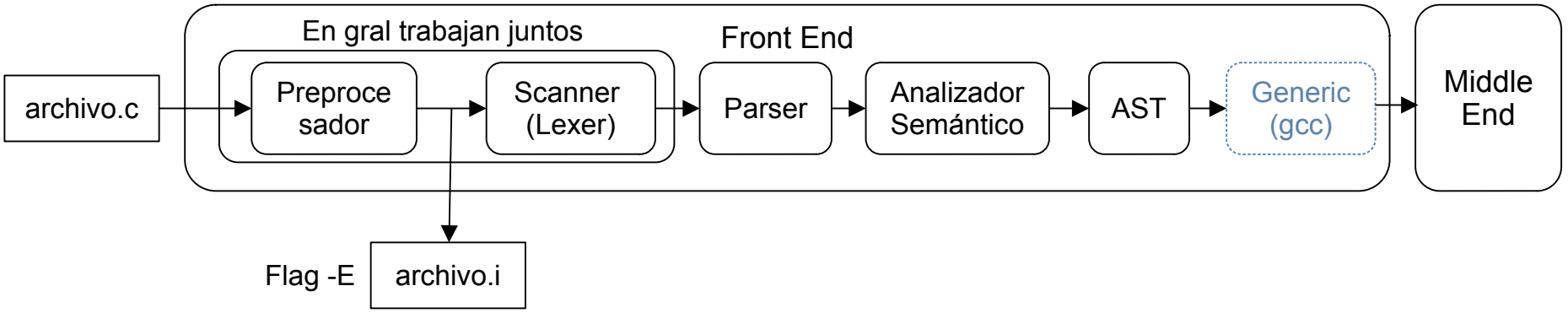
Introducción

- Análisis (Front end)
  - Scanner: es el analizador léxico, recibe una secuencia de caracteres (el código fuente), reconoce los lexemas de los distintas categoría léxicas (LR) y genera como salida una sequencia de tokens
  - Parser: es el analizador sintáctico, recibe los tokens generados por el scanner y verifica que cumplan la GIC del lenguaje
  - Analizador semántico: realiza comprobaciones que incluyen el contexto, como el tipo de las variables, que no hay definiciones duplicadas y otras. A su vez genera un primer código intermedio
- Síntesis (Back end)
  - Optimización del código intermedio
  - Generación del código para la máquina concreta donde debe ejecutar

Etapas

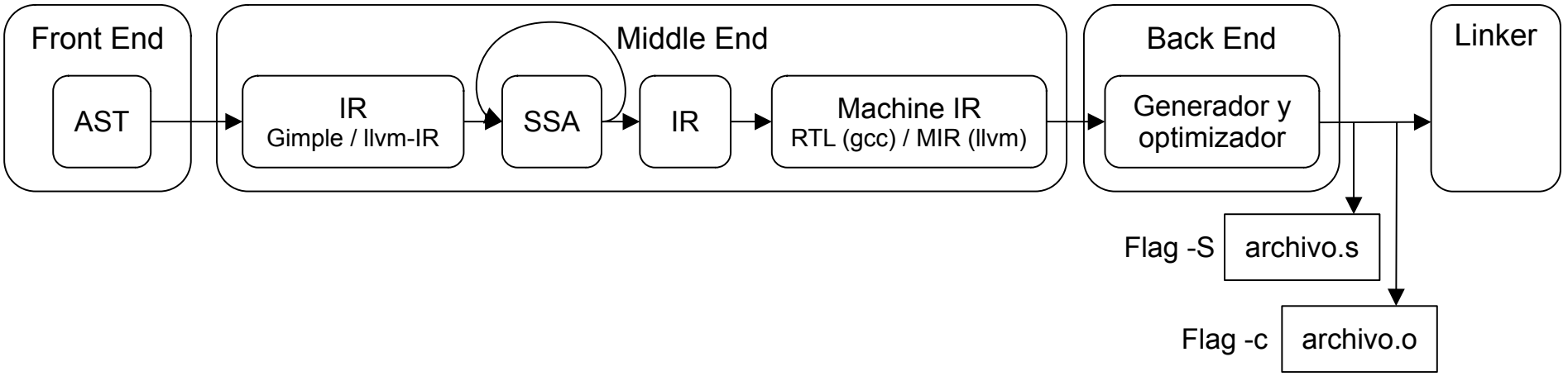

Código Fuente
Front End
Preprocesador
Flujo de caracteres
Analizador Léxico (Scanner)
Flujo de tokens
Analizador Sintáctico (Parser)
Tabla de Símbolos
Generador Código Intermedio
Manejador De Errores
Middle End
Optimizador
Back End
Generador De Código
Código Objeto
Código Objeto
Código Objeto
Vinculador
Imagen Ejecutable

Front End



AST : Abstract Syntax Tree

Middle y Back End



IR : intermediate representation
SSA: static single assignment
(para hacer optimizaciones)
RTL : register transfer language

Ejemplo con varios fuentes

Archivo main.c

```c
#include <stdio.h>
#include "mate.h"
#define CUATROPTOUNO 4.1

int main(void)
{
    double a = 21.32;
    double b = CUATROPTOUNO;
    double/*COMENTARIO*/z;

    printf("glob = %f\n"
           ,glob_x_defecto);
    z = division(a, b);
    printf("%4.2f / %4.2f = %4.2f\n"
           , a, b, z);

    b = 0.0;
    z = division(a, b);
    printf("%4.2f / %4.2f = %4.2f\n"
           , a, b, z);
    return 0;
}
```

Salida:

```
glob = 1.000000
21.32 / 4.10 = 5.20
21.32 / 0.00 = 21.32
```

Archivo mate.h

```c
#ifndef MATE_H_INCLUDED
#define MATE_H_INCLUDED

double division(double dividendo,
                double divisor);
extern double glob_x_defecto;

#endif // MATE_H_INCLUDED
```

Archivo mate.c

```c
#include "mate.h"
double glob_x_defecto = 1;

double division(double dividendo,
                double divisor)
{
    if (divisor == 0.0)
        divisor = glob_x_defecto;
    return dividendo/divisor;
}
```

Buenas Prácticas

- El código de las funciones y definiciones de variables van en los fuentes (archivos .c)
- Por cada fuente tendré un archivo encabezado (.h) con el mismo nombre, conteniendo las declaraciones de las funciones y variables (extern) que quiera dar a conocer del fuente correspondiente.
- Usar guardas de inclusión en los archivos encabezado.

Comandos del compilador (gnu)

- Para hacer una imagen ejecutable a partir de varios fuentes, el formato general es:
  - gcc fuente1.c fuente2.c … fuenteN.c -o programa
- Algunas de las opciones más relevantes
  - g: generara símbolos para el debugger
  - std=xxx: usar el estándar xxx
  - pedantic-errors: adherir estrictamente al estándar
  - Wall: dar todas las advertencias posibles
    - Wextra: da algunas adevertencias más.

Compilación parcial

- Para solo preprocesar
  - gcc -E fuente.c -o fuente.i
- Para generar lenguaje ensamblador
  - gcc -S fuente.c -o fuente.s
- Para generar el archivo objeto pero no vincular
  - gcc -c fuente.c -o fuente.o

Makefile

Licencia