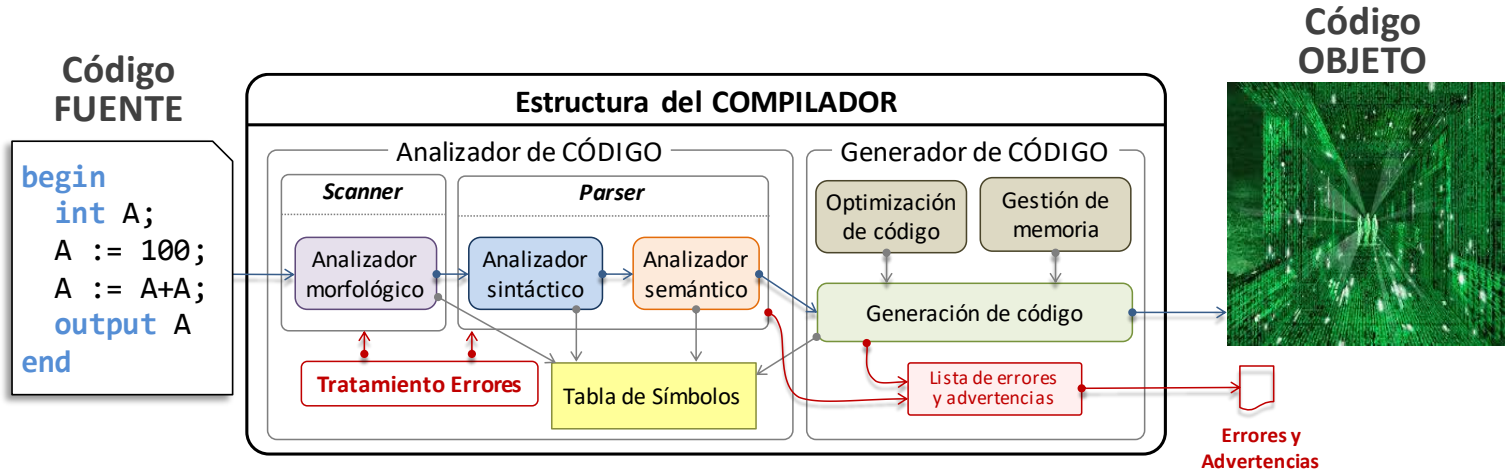
Introducción a los procesadores de lenguaje - Resumen



El compilador GCC - Resumen

PREPROCESADO (ej1)

```
$ gcc -E file.c > file.pp
$ cpp file.c > file.pp
$ more file.pp
```

COMPILACIÓN (ej1, …, ej12)

```
$ gcc -S file.c
$ gcc –Wall -S file.c
```

ENSAMBLADO (ej1, …, ej12)

```
$ as file.s –o file.o
$ file file.o
```

ENLAZADO (ej1)

```
$ ld -o execute archivo.o -lc
  ld: warning: cannot find entry symbol _start; defaulting to 08048184
```

```
$ ld -o execute /usr/lib/gcc-lib/i386-linux/2.95.2/collect2 -m
elf_i386 -dynamic-linker /lib/ld-linux.so.2 -o file /usr/lib/crt1.o
/usr/lib/crti.o /usr/lib/gcc-lib/i386-linux/2.95.2/crtbegin.o -
L/usr/lib/gcc-lib/i386-linux/2.95.2 file.o -lgcc -lc -lgcc
/usr/lib/gcc-lib/i386-linux/2.95.2/crtend.o /usr/lib/crtn.o
```

El compilador GCC

EN UN SOLO PASO

El proceso anterior se hace un solo paso:

```
$ gcc -o execute file.c
$ ./execute
```

Ver detalle de la compilación:

```
$ gcc -Wall -v -o execute execute.c
```

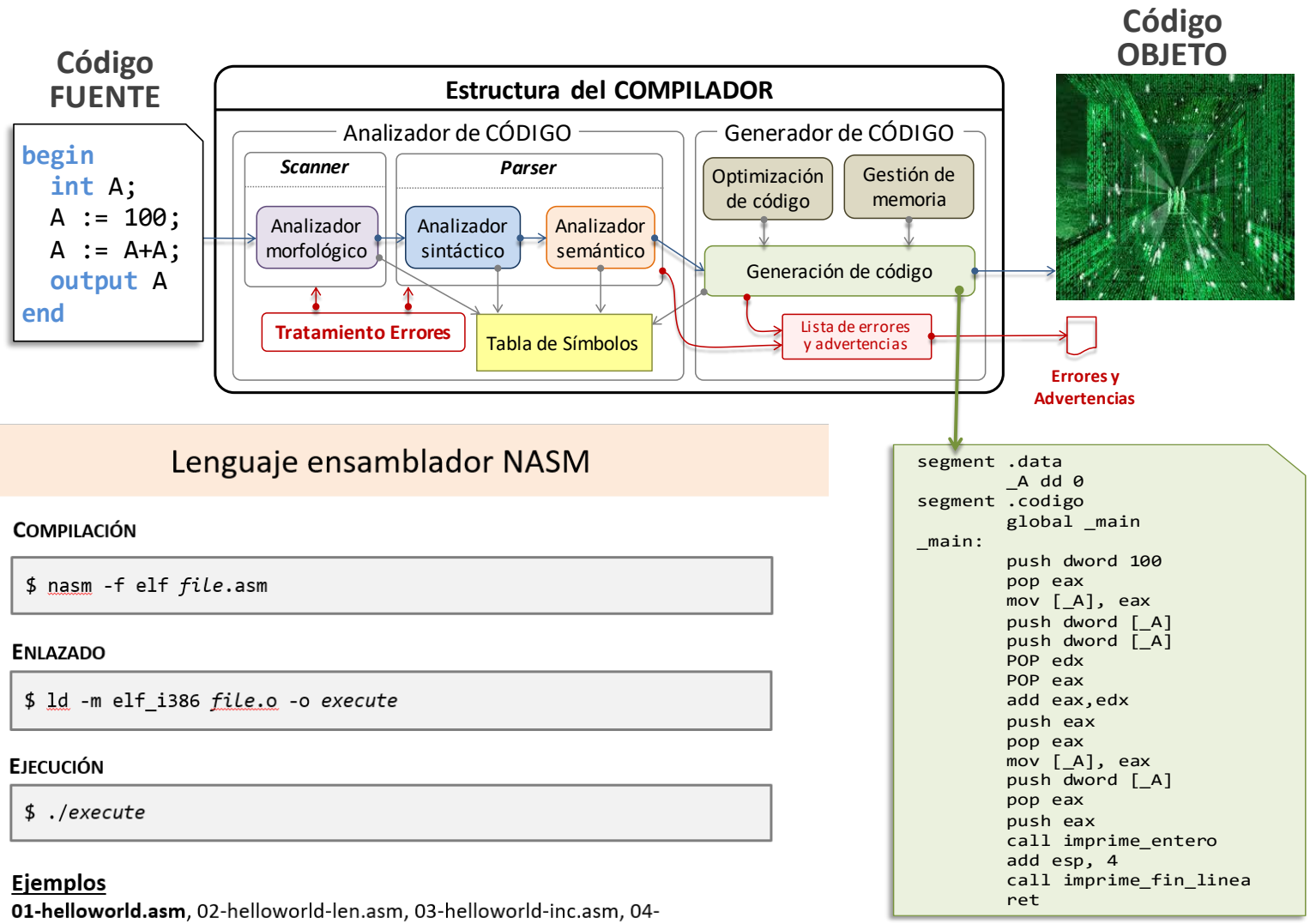
Enlace estático: los binarios de las funciones se incorporan al código binario del ejecutable. Fichero más grande.

```
$ gcc –static -o file file.c
$ ls –l file
```

Enlace dinámico: el código de las funciones permanece en la biblioteca; el ejecutable cargará en memoria la biblioteca y ejecutará la parte de código

```
$ gcc -o file file.c
$ ls –l file
```

Lenguaje ensamblador NASM - Resumen



Lenguaje ensamblador NASM

COMPILACIÓN

1. `$ nasm -f elf file.asm`

ENLAZADO

2. `$ ld -m elf_i386 file.o -o execute`

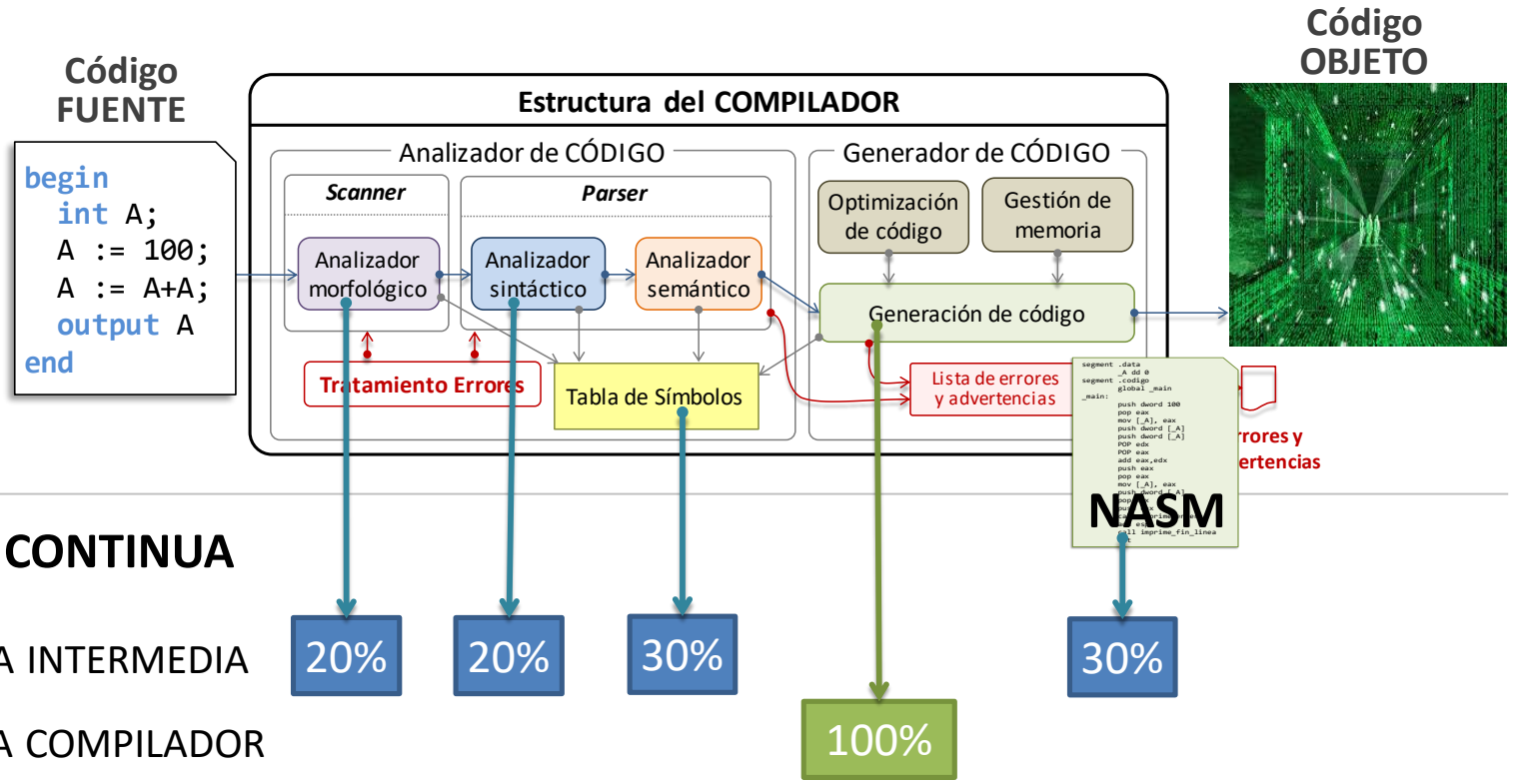
EJECUCIÓN

3. `$ ./execute`

Ejemplos

01-helloworld.asm, 02-helloworld-len.asm, 03-helloworld-inc.asm, 04-helloworld-if.asm, 05-helloworld-args.asm, 06-helloworld-input.asm, 07-helloworld-itoa.asm, 08-calculator-addition.asm, 09-calculator-subtraction.asm, 10-calculator-multiplication.asm, 11-calculator-division.asm

Normativa



Evaluación continua

[40%] Entrega intermedia

[40%] Entrega compilador

[20%] Examen final

[+1.5] Nota de clase

Pruebas y ejercicios (más de 5)

Evaluación NO continua

[30%] Entrega compilador (más 5 puntos)

[70%] Examen final (más 5 puntos)

Lenguaje ensamblador NASM - Estructura

```
; Comentarios del programa

; Inclusión de librería NASM
%include '<nombre_librería>'
...

segment .data
  ; Declaración de variables inicializadas
  _<variable> <tamaño> <valor_inicial>
segment .bss
  ; Declaración de variables NO inicializadas
  _<variable> <tamaño> <cantidad>

segment .text
  ; Definición de programa principal
  global  main
  extern <function_extern>, ...
  ...

    ; Definición de funciones
    _<nombre_función>:
      ; Instrucciones de la función
      ...
      ret
    ...

    main:
      ; Instrucciones del programa principal
      ...
      ret
```

Sección de datos (opcional)

Sección de código

1 COMPILACIÓN

```
$ nasm -g -o file.o -f elf file.asm
```

2 ENLAZADO

```
$ gcc -o execute file.o extern.o
```

3 EJECUCIÓN

```
$ ./execute
```

LIBRERÍA: **alfalib.o**

```
scan_int, print_int, scan_float,
print_float, scan_boolean,
print_blank,print_endofline,
print_string, print_boolean,
alfa_malloc, alfa_free, ld_float
```

Lenguaje ensamblador NASM - Ejemplo

01-library.asm

```
; Manejo de librerías
_quit :
  mov ebx, 0 ;Devolver Status 0 para salir sin errores
  mov eax, 1 ;Invocar SYS_EXIT
  int 80h
  ret
```

01-hellohello.asm

```
; Hello World Program NSAM
%include '01-library.asm'

segment .data
  _msg db 'Hello World!', 0Ah ; Constante de cadena

segment .text
  global  main
    _hello_world:
      mov edx, 13      ; Número de bytes a escribir
      mov ecx, _msg    ; Mover de memoria a ecx
      mov ebx, 1       ; Escribir a STDOUT
      mov eax, 4       ; Invocar SYS_WRITE
      int         80h
      ret

    main:
      call _hello_world
      call _quit
```

Registros 32 bits

```
eax, ecx, edx, ebx
```

Puntero de pila

```
esp
```

Registros 64 bits

```
ax, cx, dx, bx
```

Declaración de datos

```
db, dw, dd, dq, dt
```

Registros 32 bits

```
mov reg, reg
mov reg, [_var]
mov [_var], reg
```

Tamaño

```
resb (1 byte)
resw (2 byte)+
resd (4 byte)
```

Apilar

```
push dwork reg
push dwork [_var]
```

Apilar

```
pop dwork reg
pop dwork [_var]
```

Suma

```
add eax edx
```

Resta

```
sub eax edx
```

Negación

```
neg eax
```

División enteros

```
idiv ecx
```

Multiplicación enteros

```
imul ecx
```

Comparación

```
cmp eax edx
```

Conjunción

```
and eax edx
```

Disjunción

```
or eax edx
```

Saltos condicionales

```
je etq (eax==edx)
jne etq (eax!=edx)
jle etq (eax<=edx)
jge etq (eax>=edx)
jl etq (eax<edx)
jg etq (eax>edx)
```

Lenguaje ensamblador NASM – Ejemplo usando alfalib.o

01-hello-alfa.asm

```
; Hello World Program NSAM (Usando alfalib.o)
segment .data
  _msg db 'Hello World!', 0 ; Constante de cadena

segment .bss
  __esp    resd 1

segment .text
  global main
  extern print_endofline, print_string
  _init :
    mov dword [__esp] , esp ; Guarda el puntero de la pila
    ret
  _quit :
    mov dword esp, [__esp]  ; Restauración de puntero de pila
    ret

  main:
    call _init
    push dword _msg  ; Mover de memoria a dword
    call print_string
    add esp, 4
    call print_endofline
    call _quit
    ret
```

Lenguaje ensamblador NASM – Ejercicio en clase

- Recursos comunes
  - Introducción a NASM
  - Descripción de la librería alfalib.o
- Recursos del curso
  - Guía NASM (Semana 2)
  - Ejemplos NASM
  - Ejemplos NASM usando alfalib.o