Índice

- ✓ El compilador GCC
- ✓ Lenguaje NASM

El compilador GCC

GCC (GNU Compiler Collection) es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.

Opciones	
-c	realiza preprocesamiento y compilación, obteniendo el archivo en código objeto; no realiza el enlazado.
-E	realiza solamente el preprocesamiento, enviando el resultado a la salida estándar.
-o <*file*	indica el nombre del archivo de salida, cualesquiera sean las etapas cumplidas.
-I*path*	especifica la ruta hacia el directorio donde se encuentran los archivos marcados para incluir en el programa fuente. No lleva espacio entre la I y la ruta, así: -I/usr/include
-L	especifica la ruta hacia el directorio donde se encuentran los archivos de biblioteca con el código objeto de las funciones referenciadas en el programa fuente. No lleva espacio entre la L y la ruta, así: -L/usr/lib

El compilador GCC

Opciones	
-Wall	muestra todos los mensajes de error y advertencia del compilador, incluso algunos cuestionables pero en definitiva fáciles de evitar escribiendo el código con cuidado.
-g	incluye en el ejecutable generado la información necesaria para poder rastrear los errores usando un depurador, tal como GDB (GNU Debugger).
-v	indica el nombre del archivo de salida, cualesquiera sean las etapas cumplidas.

El compilador GCC - Etapas

PREPROCESADO

1

En esta etapa se interpretan las directivas al preprocesador. Por ejemplo, sustituye #define por su valor.

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

COMPILACIÓN

2

La compilación transforma el código C en lenguaje ensamblador propio del procesador de nuestra máquina. Realiza las dos primeras etapas creando el archivo *file*.s

```
$ gcc -S file.c
$ more file.s
```

ENSAMBLADO

3

El ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto, un archivo binario en lenguaje de máquina ejecutable por el procesador.
El ensamblador se denomina as. crea el archivo en código objeto *file*.o a partir del archivo en lenguaje ensamblador *file*.s.

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

El compilador GCC - Etapas

4

ENLAZADO

Las funciones de C/C++ incluidas en el código, tal como printf(), se encuentran ya compiladas y ensambladas en bibliotecas existentes en el sistema. Es preciso incorporar el código binario de estas funciones a nuestro ejecutable. En esto consiste la etapa de enlace, donde se reúnen uno o más módulos en código objeto con el código existente en las bibliotecas.

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

Da este error por falta de referencias, es necesario escribir algo para obtener un ejecutable.

```
$ 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 circulo.o -lgcc -lc -lgcc
/usr/lib/gcc-lib/i386-linux/2.95.2/crtend.o /usr/lib/crtn.o
```

El compilador GCC - Resumen

PREPROCESADO (ej1)

1

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

COMPILACIÓN (ej1, …, ej12)

2

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

ENSAMBLADO (ej1, …, ej12)

3

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

ENLAZADO (ej1)

4

```
$ 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
```

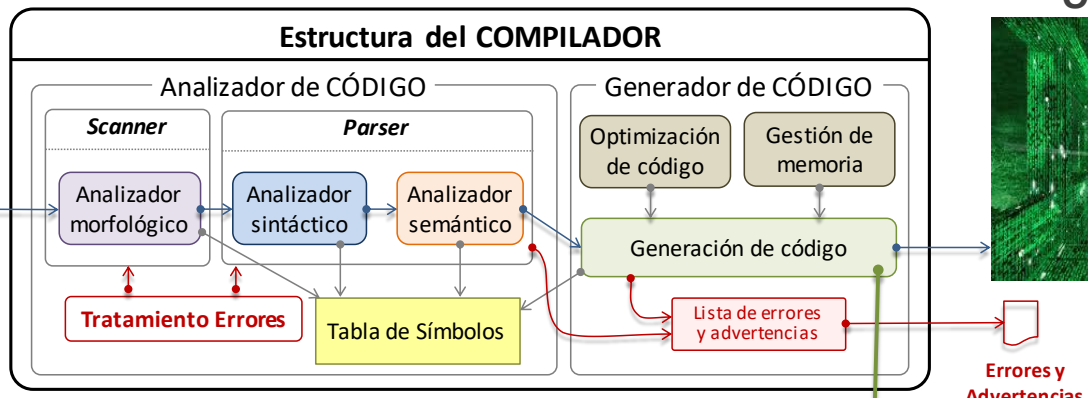
Enlace dinámico: el código de las funciones permanece en la biblioteca; el ejecutable cargará en memoria la biblioteca y ejecutará la parte de código

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

Lenguaje ensamblador NASM - Introducción

Código FUENTE

```
begin
  int A;
  A := 100;
  A := A+A;
  output A
end
```



Código OBJETO

Errores y Advertencias

```
segment .data
        _A dd 0
segment .codigo
        global _main
_main:
        push dword 100
        pop eax
        mov [_A], eax
        push dword [_A]
        push dword [_A]
        POP edx
        POP eax
        add eax,edx
        push eax
        pop eax
        mov [_A], eax
        push dword [_A]
        pop eax
        push eax
        call imprime_entero
        add esp, 4
        call imprime_fin_linea
        ret
```

Lenguaje ensamblador NASM - Introducción

- Se trata de la documentación necesaria para realizar los ejercicios que se le reclaman al estudiante y que persiguen un doble objetivo:
  - Familiarizarse con el lenguaje ensamblador NASM.
  - Utilizar este lenguaje como lenguaje objeto de un compilador.
- No se trata de utilizar NASM de la forma “tradicional” como se puede utilizar en las prácticas de otras asignaturas de programación en ensamblador, sino de utilizarlo como lenguaje objetivo. Esto implicará el uso de un número mucho menor de instrucciones de entre todas las posibles en NASM. El estudiante debe aprender una serie de sentencias básicas, no se trata de buscar el máximo nivel de complejidad del lenguaje.
- Dicho de otra manera, el lenguaje objetivo NASM no es sino un paso más del compilador, y se ha elegido este lenguaje por tener algunas ventajas compiladores de libre distribución en muchos sistemas y plataformas, fácil legibilidad…) como podría haberse elegido cualquier otro. Si bien se considera necesario que el estudiante que va a desarrollar un compilador tenga agilidad y conocimiento de la parte final de su proceso de compilación.

Lenguaje ensamblador NASM

El Netwide Assembler, NASM (URL: ***nasm.us***), es un ensamblador 80x86 y x86-64 diseñado para la portabilidad y la modularidad. Admite una variedad de formatos de archivos de objetos, incluidos: Linux y *BSD a.out, ELF, COFF, Mach-O, 16-bit and 32-bit OBJ (OMF) format, Win32 y Win64. Su sintaxis está diseñada para ser simple y fácil de entender, similar a la sintaxis en el Manual del desarrollador de software Intel con una complejidad mínima. Es compatible con todas las extensiones arquitectónicas x86 actualmente conocidas, y tiene un fuerte soporte para macros.

NASM COMMAND-LINE SYNTAX

```
$ nasm -f <format> <filename> [-o <output>]
$ nasm –h
```

Opciones	
-f	Specifying the Output File Format.
-o	Specifying the Output File Name.

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

Estructura Código ASM

```
; Comentarios iniciales

segment .bss
;
; Declaración de variables
;
segment .data
;
; Declaración de variables inicializadas
;
segment .text
        global    main
;
; Código correspondientes a las funciones
;
        main:
; Código del programa principal
```