FASES DEL GCC

El proceso de compilación consta de las siguientes fases: Preprocesamiento, Compilación (propiamente dicha), Ensamblado y Enlazado.

Para mostrar cada una de las fases se trabajará sobre el siguiente código C:

```
#include <stdio.h>
#define STRING "Hola, mundo!"

int main (void){
    // Imprime por pantalla la cadena STRING
    printf ("Salida: %s\n",STRING);
    return 0;
}
```

1. Preoprocesamiento

Esta fase realiza básicamente tres tares: sustitución de texto, eliminación de comentarios e inclusión de archivos.
Las líneas de código que inician con el caracter # son directivas de preprocesamiento.
La sustitución de texto e inclusión de archivos se indican en el código a través de directivas de preprocesamiento.
En el código de ejemplo la primera directiva de preprocesamiento (#include) incluye un archivo .h (encabezado), y la segunda (#define) pide una sustitución de texto.
Resumiendo, en esta etapa los archivos incluidos y las macros definidas son expandidas y mezcladas en el archivo fuente, creándose un archivo nuevo temporal. Para obtener el mismo se puede ejecutar el gcc con el flag –E

```
gcc -E Fases_del_gcc.c -o Fases_del_gcc.i
```

o utilizando directamente el comando cpp (preprocesador de C):

```
cpp Fases_del_gcc.c > Fases_del_gcc.i
```

A continuación se muestra un fragmento del archivo temporal

```
.
.
.

wint_t __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) _fgetwchar (void);
```

```
 wint_t __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) _fputwchar (wint_t);
 int __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) _getw (FILE*);
 int __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) _putw (int, FILE*);


 wint_t __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) fgetwchar (void);
 wint_t __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) fputwchar (wint_t);
 int __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) getw (FILE*);
 int __attribute__((__cdecl__)) __attribute__ ((__nothrow__)) putw (int, FILE*);
# 2 "Fases_del_gcc.c" 2


int main (void){

  printf ("Salida: %s\n","Hola, mundo!");
  return 0;
}
```

Se puede observar código antes del main. Esto es debido a que el archivo stdio.h ha sido expandido e incluído. Asimismo se puede ver que la macro STRING ha sido reemplazada por la cadena “Hola, mundo!” y que el comentario anterior al printf ha sido eliminado.

2. Compilación (propiamente dicha).

En esta fase el código fuente es realmente compilado produciendo un código ensamblador, específico para una determinada arquitectura. En el caso en que la arquitectura para la que se quiera generar código sea distinta a la que se está utilizando para correr el compilador al proceso se lo denomina compilación cruzada (*cross-compilation*).

El archivo con las instrucciones en assembly se lo puede obtener mediante el siguiente comando (utilizando el flag -S):

```
gcc -S Fases_del_gcc.c -o Fases_del_gcc.s
```

A continuación se muestra el archivo generado:

```
        .file   "Fases_del_gcc.c"
        .def    ___main;        .scl    2;      .type   32;     .endef
        .section .rdata,"dr"
LC0:
        .ascii "Hola, mundo!\0"
LC1:
        .ascii "Salida: %s\12\0"
        .text
        .globl  _main
        .def    _main;  .scl    2;      .type   32;     .endef
_main:
```

```
LFB6:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        andl    $-16, %esp
        subl    $16, %esp
        call    ___main
        movl    $LC0, 4(%esp)
        movl    $LC1, (%esp)
        call    _printf
        movl    $0, %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE6:
        .def    _printf; .scl   2;      .type   32;     .endef
```

3. Ensamblado

En esta fase se convierte el código assembly en código de máquina (código binario, llamado objeto).

El comando a ejecutar es el siguiente:

```
gcc -c Fases_del_gcc.s -o Fases_del_gcc.o
```

o llamando directamente al comando as (ensamblador GNU):

```
as Fases_del_gcc.s -o Fases_del_gcc.o
```

4. Enlazado

En esta fase se asegura que todos los símbolos indefinidos del código sean resueltos. En nuestro ejemplo se debe resolver la llamada a la función printf.
En el caso en que la aplicación que se quiera obtener cuente con más de un archivo fuente la fase de ensamblado generará un archivo objeto por cada uno de ellos. Es en esta fase donde todos estos se enlazan entre sí. El archivo así obtenido es finalmente el ejecutable.

El comando a ejecutar es el siguiente:

```
gcc Fases_del_gcc.o -o Fases_del_gcc.exe
```

En la siguiente imagen se pueden observar cada una de las etapas del gcc y la corrida final de la aplicación Fases_del_gcc

```
Z:\Electronica_Digital_II>gcc -c Fases_del_gcc.s -o Fases_del_gcc.o

Z:\Electronica_Digital_II>gcc Fases_del_gcc.o -o Fases_del_gcc.exe

Z:\Electronica_Digital_II>gcc -E Fases_del_gcc.c -o Fases_del_gcc.i

Z:\Electronica_Digital_II>gcc -S Fases_del_gcc.i -o Fases_del_gcc.s

Z:\Electronica_Digital_II>gcc -c Fases_del_gcc.s -o Fases_del_gcc.o

Z:\Electronica_Digital_II>gcc Fases_del_gcc.o -o Fases_del_gcc.exe

Z:\Electronica_Digital_II>Fases_del_gcc.exe
Salida: Hola, mundo!
```