



Sistema de procesamiento de Lenguaje

Objetivo

El alumno identificará el funcionamiento de las entradas, salidas y los programas que componen el sistema de procesamiento de lenguaje, así como el formato de cada uno de los archivos con los que se trabajan.

Introducción

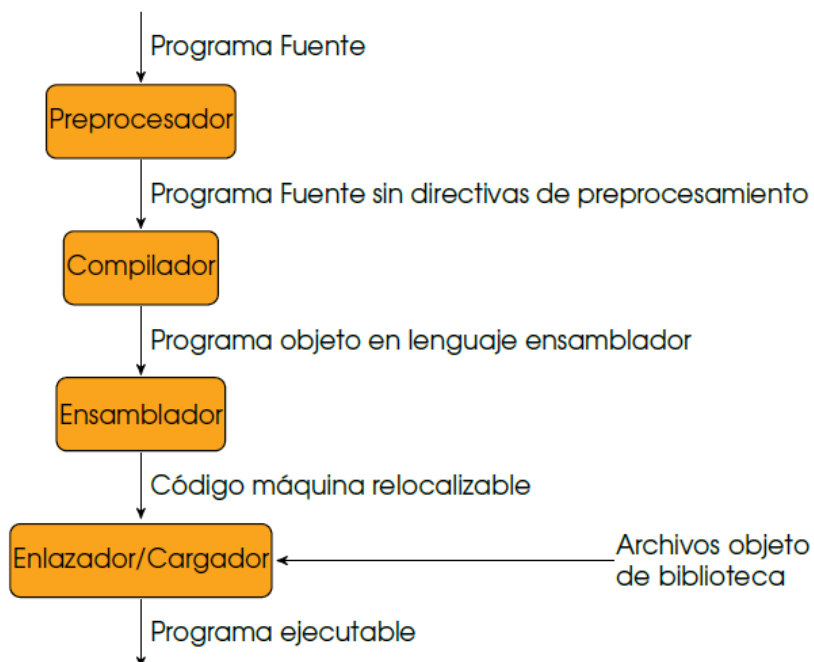
Un compilador es un programa que se encarga de hacer la traducción de un programa fuente escrito en lenguaje de alto nivel a un programa escrito en lenguaje objeto que por lo general es un lenguaje de bajo nivel. El compilador realiza el análisis del código fuente y la síntesis a código objeto.

Para realizar esta traducción el compilador se auxilia de otros programas como lo son el preprocesador, que se encarga de recolectar el programa escrito en módulos en archivos separados, expandir fragmentos de código abreviados de uso frecuentes, llamados macros.

El programa modificado por el preprocesador ingresa al compilador, éste producirá el programa destino escrito en lenguaje ensamblador, que a continuación es procesado por un ensamblador que genera el código de máquina.

Una vez que el programa ha sido ensamblado, es necesario vincular los archivos de código máquina con otros archivos objeto y de biblioteca para que se produzca el código ejecutable.

Finalmente el cargador lleva el archivo objeto a la memoria para su ejecución.



Desarrollo:

1. Deberá tener instalado el compilador gcc o trabajar en un ambiente Linux.
2. Escriba el siguiente código en un archivo llamado programa.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  //define PI 3.1415926535897
4  #ifndef PI
5  #define area(r) (PI*r*r)
6  #else
7  #define area(r) (3.1416*r*r)
8  #endif
9  /*
10 * Este es un programa de prueba, para verificar
11 * el funcionamiento del sistema de procesamiento de lenguaje
12 */
13 int main(void){
14     printf("Hola Mundo!\n"); // Funcion para imprimir hola mundo
15     float mi_area = area(3); //soy un comentario ...
16     printf("Resultado: %f\n", mi_area);
17     return 0;
18 }
19
```



The screenshot shows a code editor window titled 'programa.c'. The code is identical to the one in the previous block, but with some formatting differences: line 11 has a small square icon before 'int', line 12 has a small square icon before 'printf', and line 13 has a small square icon before 'float'. The code is as follows:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  //define PI 3.1415926535897
4  #ifndef PI
5  #define area(r) (PI*r*r)
6  #endif
7  /*
8  * Este es un programa de prueba, para verificar
9  * el funcionamiento del sistema de procesamiento de lenguaje
10 */
11 int main(void){
12     printf("Hola mundo!\n"); // Función para imprimir hola mundo.
13     float mi_area=area(3); // soy un comentario ...
14     printf("Resultado: %f\n", mi_area);
15     return 0;
16 }
```

3. Usar el siguiente comando: **cpp programa.c > programa.i**

- (a) Busque los archivos .h y revise su contenido
- (b) Compare el contenido de programa.i con el de stdio.h y stdlib.h, indique de forma general las similitudes entre los archivos .h y el .i
- (c) Observe lo que ocurrió con los comentarios y las directivas de preprocesador.

```
Símbolo del sistema
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Daira Aketzalli>cd "C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\Compiladores\Programas"

C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\Compiladores\Programas>cpp programa.c > programa.i

C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\Compiladores\Programas>
```

Podemos ver que se creó un nuevo programa, se agregó código y al final de todo el código agregado está la función main que creamos en el archivo programa.c, sin embargo, no se encuentran las instrucciones ifdef y define, ni las bibliotecas que pusimos en nuestro programa.c.

```
programa.c [x]  programa.i [x]  programa.s [x]
1  # 1 "programa.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "programa.c"
5  # 1 "d:\\mingw\\include\\stdio.h" 1 3
6  # 38 "d:\\mingw\\include\\stdio.h" 3
7
8  # 39 "d:\\mingw\\include\\stdio.h" 3
9  # 56 "d:\\mingw\\include\\stdio.h" 3
10 # 1 "d:\\mingw\\include\\_mingw.h" 1 3
11 # 55 "d:\\mingw\\include\\_mingw.h" 3
12
13 # 56 "d:\\mingw\\include\\_mingw.h" 3
14 # 66 "d:\\mingw\\include\\_mingw.h" 3
15 # 1 "d:\\mingw\\include\\msvcrtver.h" 1 3
16 # 35 "d:\\mingw\\include\\msvcrtver.h" 3
17
18 # 36 "d:\\mingw\\include\\msvcrtver.h" 3
19 # 67 "d:\\mingw\\include\\_mingw.h" 2 3
20
21
22
23
24
25
26 # 1 "d:\\mingw\\include\\w32api.h" 1 3
27 # 35 "d:\\mingw\\include\\w32api.h" 3
28
29 # 36 "d:\\mingw\\include\\w32api.h" 3
30 # 59 "d:\\mingw\\include\\w32api.h" 3
31 # 1 "d:\\mingw\\include\\sdkddkver.h" 1 3
32 # 35 "d:\\mingw\\include\\sdkddkver.h" 3
33
34 # 36 "d:\\mingw\\include\\sdkddkver.h" 3
35 # 60 "d:\\mingw\\include\\w32api.h" 2 3
36 # 74 "d:\\mingw\\include\\_mingw.h" 2 3
37 # 57 "d:\\mingw\\include\\stdio.h" 2 3
38 # 69 "d:\\mingw\\include\\stdio.h" 3
39 # 1 "d:\\mingw\\lib\\gcc\\mingw32\\6.3.0\\include\\stddef.h" 1 3 4
40 # 216 "d:\\mingw\\lib\\gcc\\mingw32\\6.3.0\\include\\stddef.h" 3 4
41
42 # 216 "d:\\mingw\\lib\\gcc\\mingw32\\6.3.0\\include\\stddef.h" 3 4
43 typedef unsigned int size_t;
```

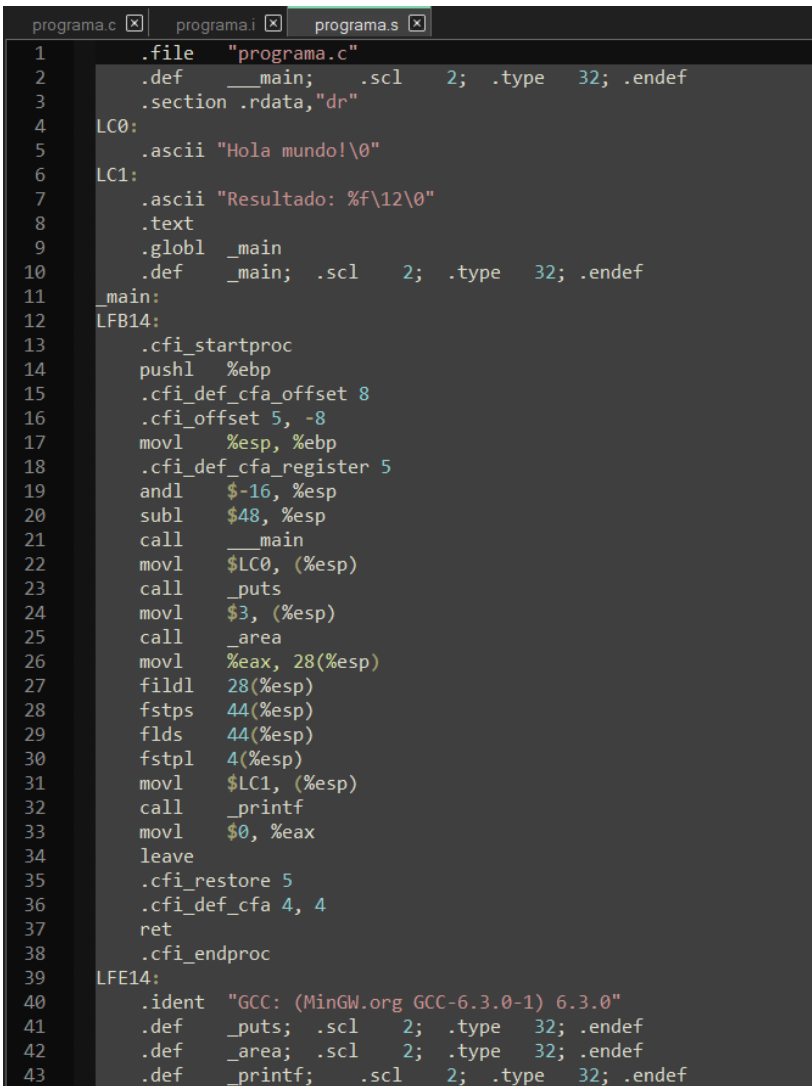
4. Ejecute la siguiente instrucción: gcc -S programa.i

- (a) ¿Qué opción se agrega para que gcc muestre todas las advertencias durante este proceso?
- (b) ¿Qué le indica a gcc la opción -S?
- (c) ¿Qué contiene el archivo de salida y cuál es su extensión?

```
C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\
Compiladores\Programas>gcc -S programa.i
programa.c: In function 'main':
programa.c:13:16: warning: implicit declaration of function 'area' [-Wimplicit-function-dec
laration]
    float mi_area=area(3);    // soy un comentario ...
                      ^~~~~

C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\
Compiladores\Programas>_
```

Al hacer uso de esta instrucción, se crea un nuevo archivo con extensión .s, en él se encuentran nuevas instrucciones escritas en lenguaje ensamblador.



```
programa.c | programa.i | programa.s
1  .file "programa.c"
2  .def __main; .scl 2; .type 32; .endef
3  .section .rdata,"dr"
4  LC0:
5  .ascii "Hola mundo!\0"
6  LC1:
7  .ascii "Resultado: %f\12\0"
8  .text
9  .globl _main
10 .def __main; .scl 2; .type 32; .endef
11 _main:
12 LFB14:
13 .cfi_startproc
14 pushl %ebp
15 .cfi_def_cfa_offset 8
16 .cfi_offset 5, -8
17 movl %esp, %ebp
18 .cfi_def_cfa_register 5
19 andl $-16, %esp
20 subl $48, %esp
21 call __main
22 movl $LC0, (%esp)
23 call _puts
24 movl $3, (%esp)
25 call _area
26 movl %eax, 28(%esp)
27 fildl 28(%esp)
28 fstps 44(%esp)
29 flds 44(%esp)
30 fstpl 4(%esp)
31 movl $LC1, (%esp)
32 call _printf
33 movl $0, %eax
34 leave
35 .cfi_restore 5
36 .cfi_def_cfa 4, 4
37 ret
38 .cfi_endproc
39 LFE14:
40 .ident "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
41 .def _puts; .scl 2; .type 32; .endef
42 .def _area; .scl 2; .type 32; .endef
43 .def _printf; .scl 2; .type 32; .endef
```

5. Ejecute la siguiente instrucción: `as programa.s -o programa.o`

- (a) ¿Formule una hipótesis sobre el contenido del archivo `.o`?
- (b) ¿Qué representa el contenido del programa `.o` de acuerdo con el diagrama de un sistema de procesamiento de lenguaje?

```
C:\Users\Daira Aketzalli\OneDrive - UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO\Documentos\9th\Compiladores\Programas>as programa.s -o programa.o
```

```
programa.o: programa.s.o: programa.s.o: programa.s.o
1  | .text
2  | .gcc: (MinGW.org GCC-6.3.0-1) 6.3.0
3  | .section .text
4  | .type 0x00000000
5  | .file "programa.c"
6  | .main
```

Se obtiene el código de máquina relocizable (¿)

6. Encuentre la ruta de los siguientes archivos en el equipo de trabajo:

- `Scrt1.o` ó `crt1.o`
- `crti.o`
- `crtbeginS.o` ó `ctrbegin.o`
- `crtendS.o` ó `ctrend.o`
- `crtn.o`

No la encontré 😞

7. Ejecute el siguiente comando, sustituyendo las rutas que encontró en el paso anterior:

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie /ruta/para/Scrt1.o /ruta/para/crti.o /ruta/para/crtbeginS.o -L/usr/lib/gcc/x86_64-pc-linux-gnu/7.3.1 -L/usr/lib -L/usr/lib -L/usr/lib programa.o -lgcc -as-needed -lgcc s -no-as-needed -lc -lgcc -as-needed -lgcc s -no-as-needed /ruta/para/crtendS.o /ruta/para/crtn.o -o ejecutable
```

O bien

```
ld -o programa -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie /usr/lib/Scrt1.o /ruta/para/crti.o /ruta/para/crtbeginS.o programa.o -lc /ruta/para/crtendS.o /ruta/para/crtn.o
```

(a) En caso de que el comando `ld` mande errores, investigue como enlazar un programa utilizando el comando `ld`.

(b) ¿Qué se generó al ejecutar el comando anterior?

8. Una vez que se enlazo el código máquina relocizable, podemos ejecutar el programa con la siguiente instrucción en la terminal: `./programa`

9. Quite el comentario de la macro `#define PI`

- (a) Genere nuevamente el archivo `.i`. De preferencia asigne un nuevo nombre.
- (b) ¿Cambio algo en la ejecución final?

10. Escribe un segundo programa en lenguaje C

En él agregue 4 directivas del preprocesador de C (cpp)*. Las directivas elegidas deben jugar algún papel en el significado del programa, ser distintas entre sí y ser diferentes de las utilizadas en el primer programa (aunque no están prohibidas, si las requieren).

Puede consultar la lista de directivas en su documentación en línea: CPP - Index of directives

(<http://gcc.gnu.org/onlinedocs/cpp/Index-of-Directives.html>).

O bien, revisar la entrada para este preprocesador en la herramienta man en Linux : \$ man cpp

(a) Explique la utilidad general de las directivas usadas y su función en particular para su programa.

11. Escriba sus resultados y conclusiones.

Al seguir los pasos descritos en este documento podemos ver el proceso que sigue un compilador (de c en este caso) para verificar que un programa esté bien escrito. Podemos comparar los programas creados a partir de las instrucciones proporcionadas con los pasos que sigue el compilador, pasando del programa fuente al programa fuente sin directivas, programa en lenguaje ensamblador, código máquina relocizable y por último la meta, que es un programa ejecutable.