**PRACTICAL-1**

**Aim : Understand modules of compilation process with the help of program. (Preprocessor, Compiler, Assembler, Linker/Loader).**

1. **Preprocessing** is the first pass of any compilation. It processes include-files, conditional compilation instructions and macros.
2. **Compilation** is the second pass. It takes the output of the preprocessor, and the source code, and generates assembler source code.
3. **Assembly** is the third stage of compilation. It takes the assembly source code and produces an assembly listing with offsets. The assembler output is stored in an object file.
4. **Linking** is the final stage of compilation. It takes one or more object files or libraries as input and combines them to produce a single

|  |
| --- |
| The compilation is a process of converting the source code into object code.  The compilation process can be divided into four steps, i.e.,   * **Preprocessor** * **Compiler** * **Assembler** * **Linker** |
|  |

* **Preprocessor**

The C compilation begins with pre-processing of source file. Pre-processor is a small software that accepts C source file and performs below tasks.

* Remove comments from the source code.
* Macro expansion.
* Expansion of included header files.

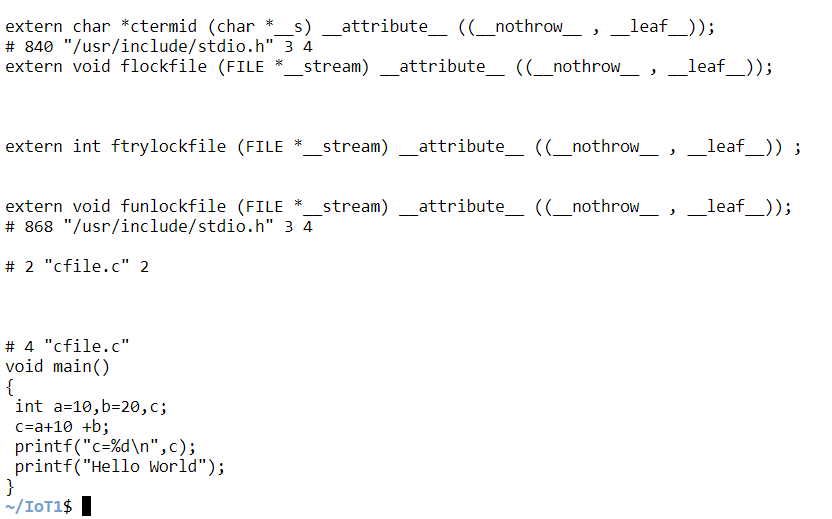
In C , For Preprocessing Any C Code You Have to Write command,

**Code: gcc -E cfile.c**

It will Give Output in that preprocessor inserts content of header files to our source code file.Pre-processor generated file is larger than the original source file.

For storing o/p of Pre-Procesing ,

**Output**



* **Compiler**

In next phase of C compilation the compiler comes in action. It accepts temporary pre-processed <file-name>.i file generated by the pre-processor and performs following tasks.

* Check C program for syntax errors.
* Translate the file into intermediate code i.e. in assembly language.
* Optionally optimize the translated code for better performance.

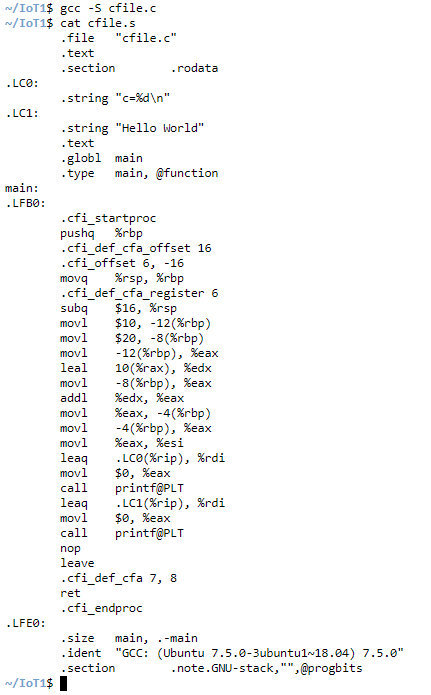
For Compilation You have to write,

**Code: gcc -S cfile.c**

**cat cfile.s**

or to store compilation into other file ,

**Output:**

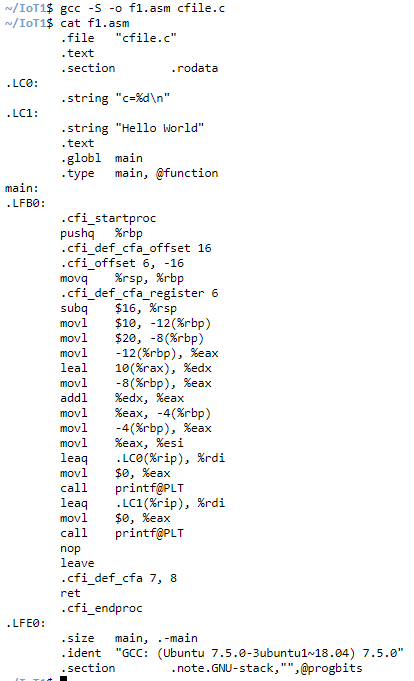


**Code: gcc -S -o f1.asm cfile.c**

**cat f1.asm**

After compiling it generates an intermediate code in assembly language as <file-name.s> file. It is assembly version of our source code.

**Output:**



* **Optimization**

Optimization is a program transformation technique, which tries to improve the code by making it consume less resources (i.e. CPU, Memory) and deliver high speed.

* Optimized code has faster execution speed.
* Optimized code utilizes the memory efficiently.
* Optimized code gives better performance.

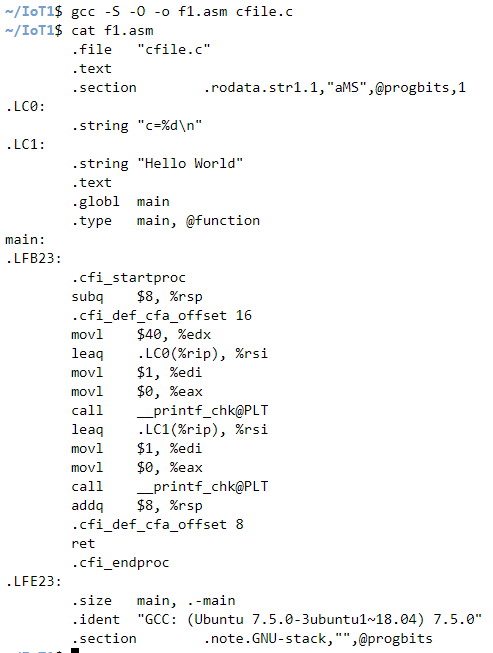
For optimize code you have to write ,

**Code: gcc -S -O -o f1.asm cfile.c**

**cat f1.asm**

It will generate one optimize file whose name is we have mentioned in command . Content of this optimize file and <file-name.s> will be same but optimizer do some code optimization for faster execution . It is not mendatory that optimizer optimizes code each time but whenever it is required to do then optimizer will definitely do that .

**Output:**



* Assembly

Moving on to the next phase of compilation. Assembler accepts the compiled source code (compilation.s) and translates to low level machine code. After successful assembling it generates <file-name.o> (in Linux) or <file-name.obj> (in Windows) file known as object file. In our case it generates the compilation.o file.

For generating Assembly code you have to write ,

**Code: gcc -C cfile.s**

**cat a.out**

It will generate .obj or .o file . This file is encoded in low level machine language and cannot be viewed using text editors. However, if you still open this in notepad, it look like.

**Output:**



* Linker/Loader

Finally, the linker comes in action and performs the final task of compilation process. It accepts the intermediate file <file-name.o> or <file-name.obj> generated by the assembler.

It links all the function calls with their original definition. Which means the function printf() gets linked to its original definition.

Linker generates the final executable file (.exe in windows).

For generating linker\loader file you have to type ,

* gcc -o name pr1.c
* ./name

**Code: gcc -C cfile.s -o j1**

**cat j1**

**Output:**



**Code: ./j1**

**Output:**

