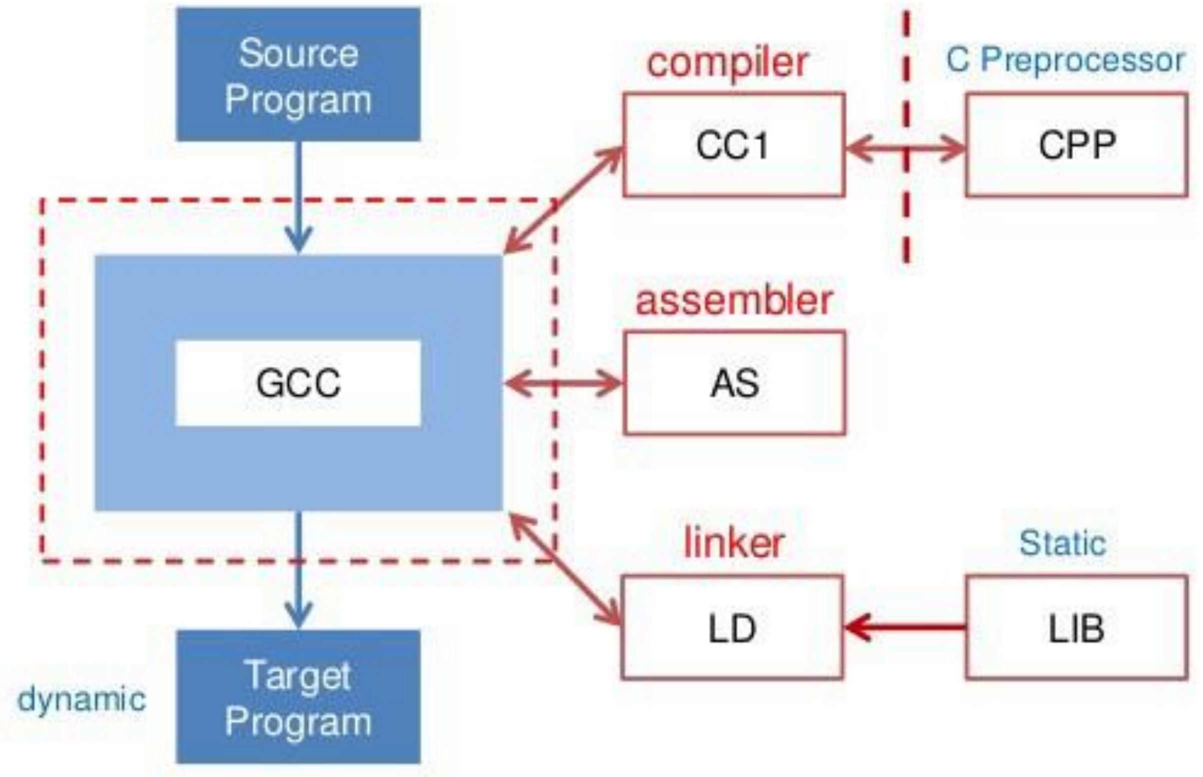
Compilation Process in GCC

hitbullseye

GCC compiler



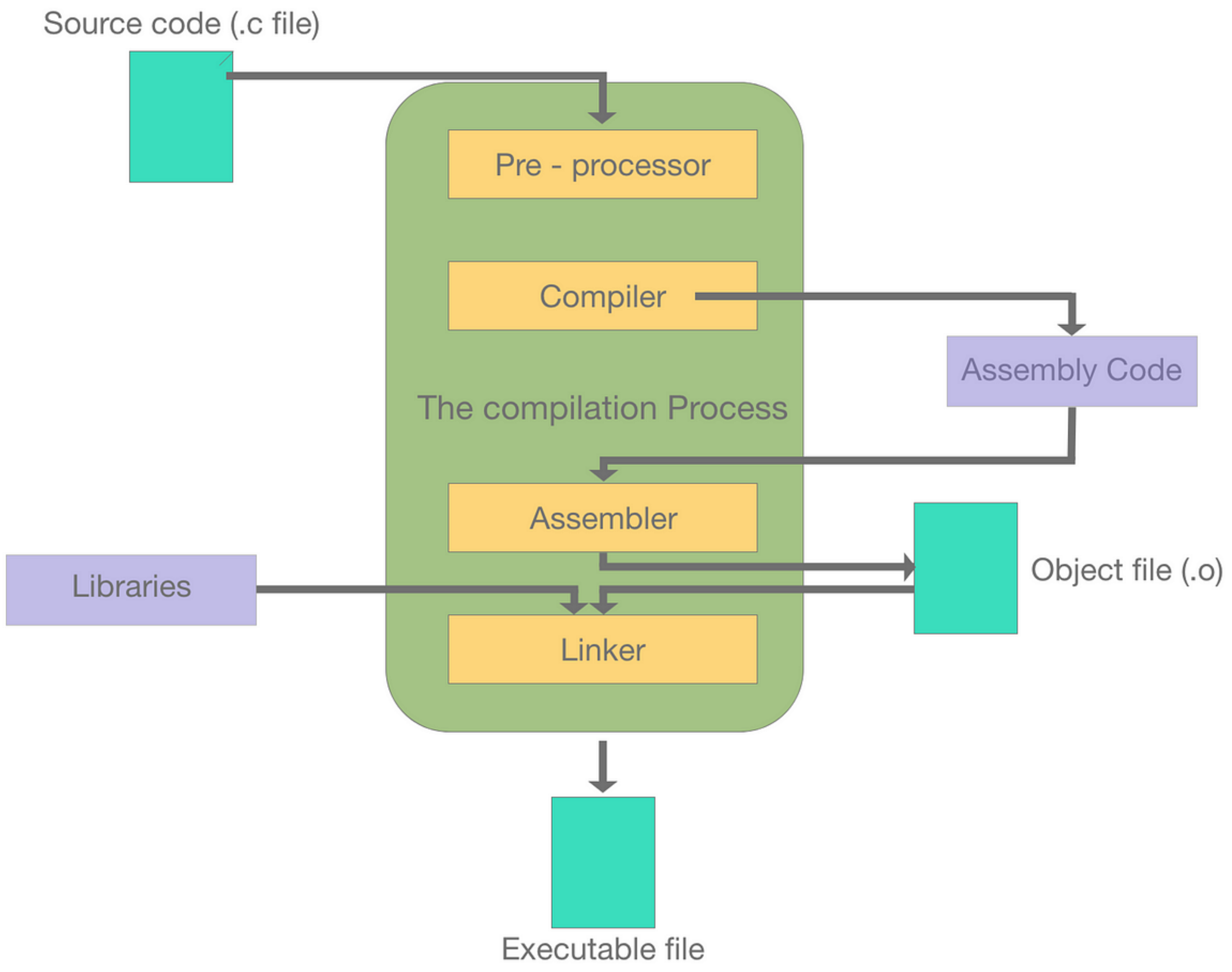
GCC is a collection that invokes compiler, assembler and linker…

An Overview of Compilation Process

- The sequence of commands executed by a single invocation of GCC consists of the following stages:
  - Preprocessing (to expand macros)
  - Compilation (from source code to assembly language)
  - Assembly (from assembly language to machine code)
  - Linking (to create the final executable)

What is Compilation

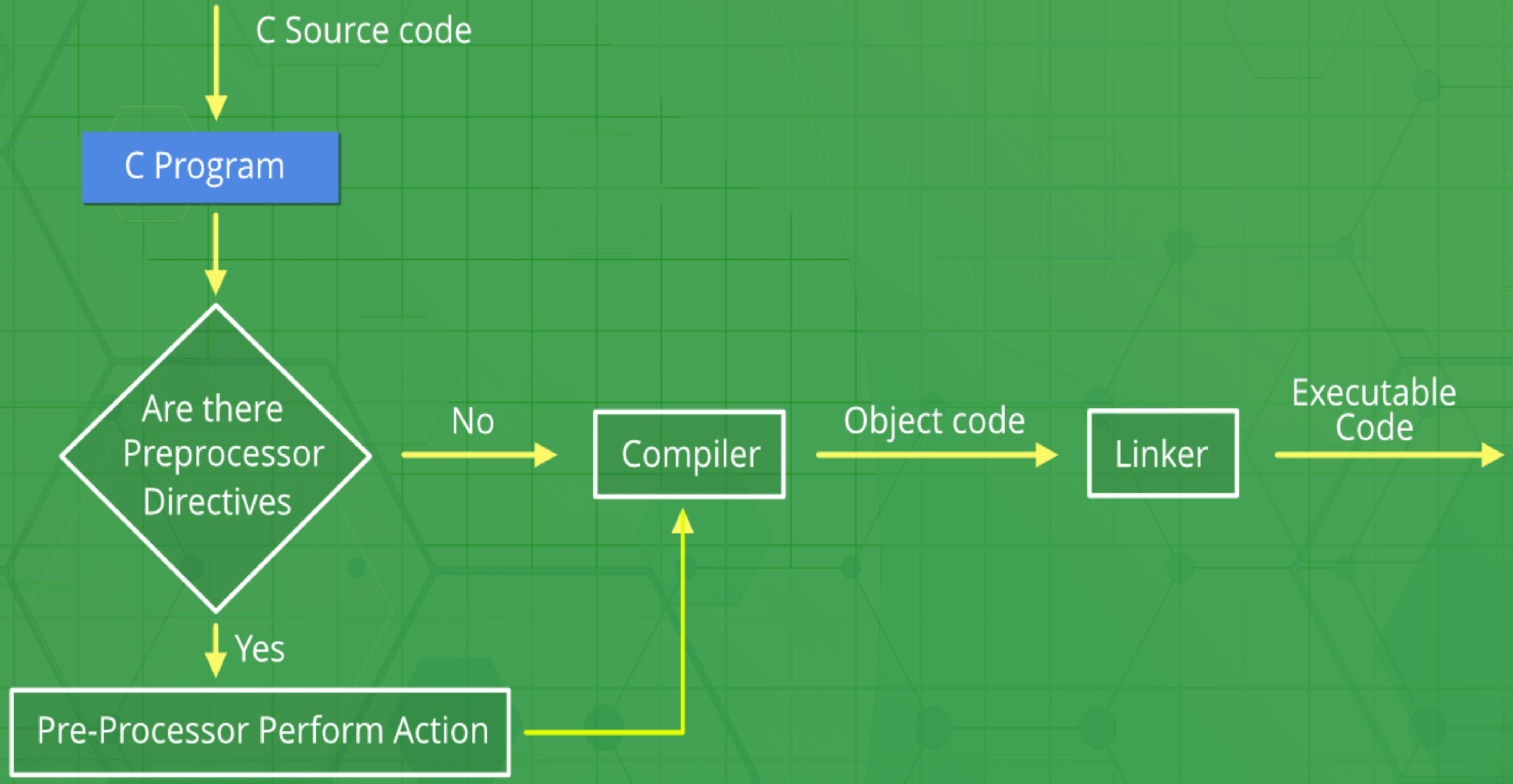
- The compilation process in C is converting an understandable human code into a Machine understandable code and checking the syntax and semantics of the code to determine any syntax errors or warnings present in our C program. Suppose we want to execute our C Program written in an IDE (Integrated Development Environment). In that case, it has to go through several phases of compilation (translation) to become an executable file that a machine can understand.


Source code (.c file)
Pre - processor
Compiler
Assembly Code
The compilation Process
Assembler
Libraries
Object file (.o)
Linker
Executable file

1. Preprocessing

- Pre-processing is the first step in the compilation process in C performed using the pre-processor tool (A pre-written program invoked by the system during the compilation). All the statements starting with the # symbol in a C program are processed by the pre-processor, and it converts our program file into an intermediate file with no # statements. Under following pre-processing tasks are performed :
  - **Comments Removal**
  - **Macros Expansion**
  - **File Inclusion**
  - **Conditional Compilation**

Preprocessor in C



Comments Removal

- Comments in a C Program are used to give a general idea about a particular statement or part of code actually, comments are the part of code that is removed during the compilation process by the pre-processor as they are not of particular use for the machine. The comments in the below program will be removed from the program when the pre-processing phase completes.

Macros expansion

- Macros are some constant values or expressions defined using the #define directives in C Language. A macro call leads to the macro expansion. The pre-processor creates an intermediate file where some pre-written assembly level instructions replace the defined expressions or constants (basically matching tokens). To differentiate between the original instructions and the assembly instructions resulting from the macros expansion, a '+' sign is added to every macros expanded statement.

File Inclusion

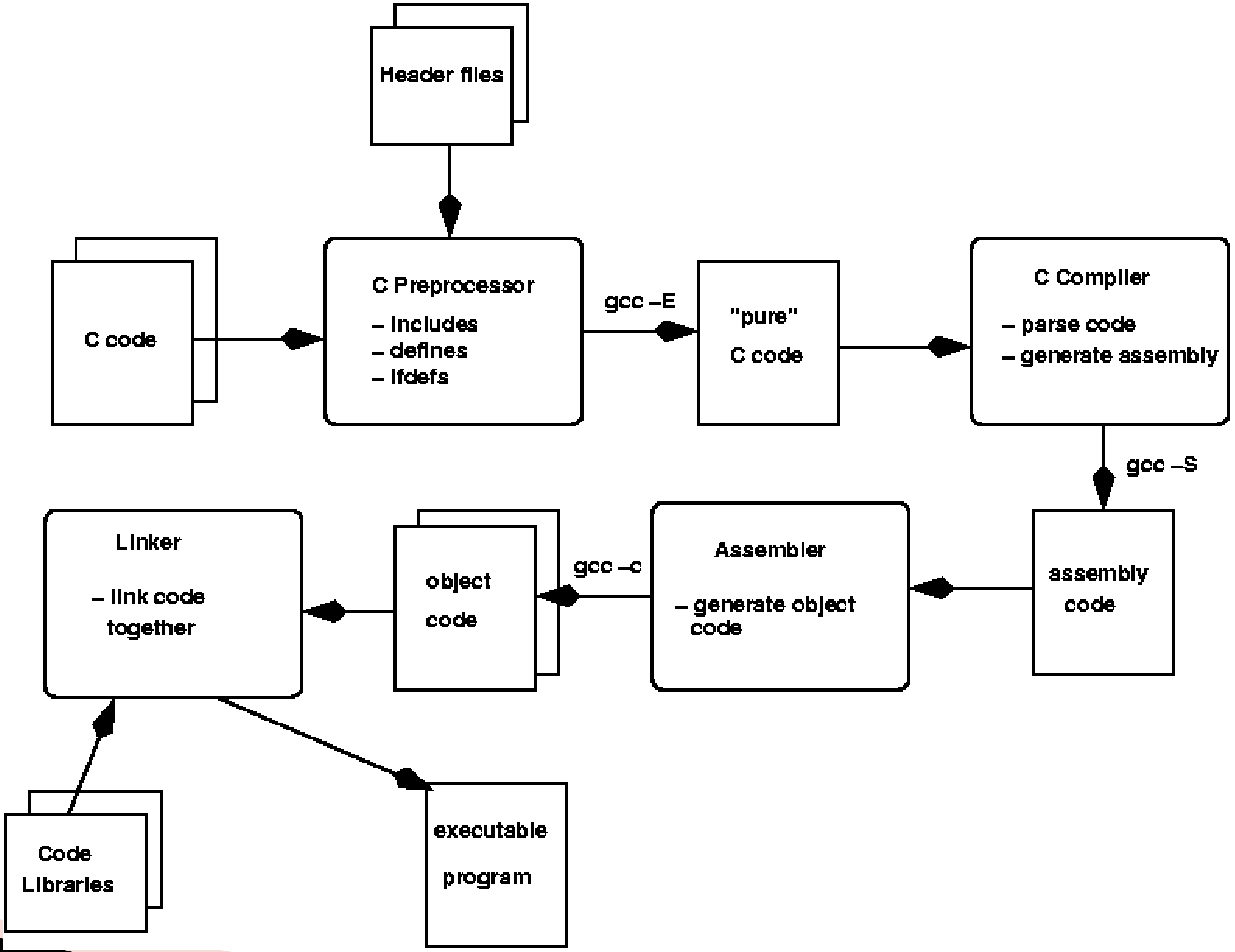
- File inclusion in C language is the addition of another file containing some pre-written code into our C Program during the pre-processing. It is done using the #include directive. File inclusion during pre-processing causes the entire content of filename to be added to the source code, replacing the #include<filename> directive, creating a new intermediate file. Example: If we have to use basic input/output functions like printf() and scanf() in our C program, we have to include a pre-defined standard input output header file i.e. stdio.h.

Conditional Compilation

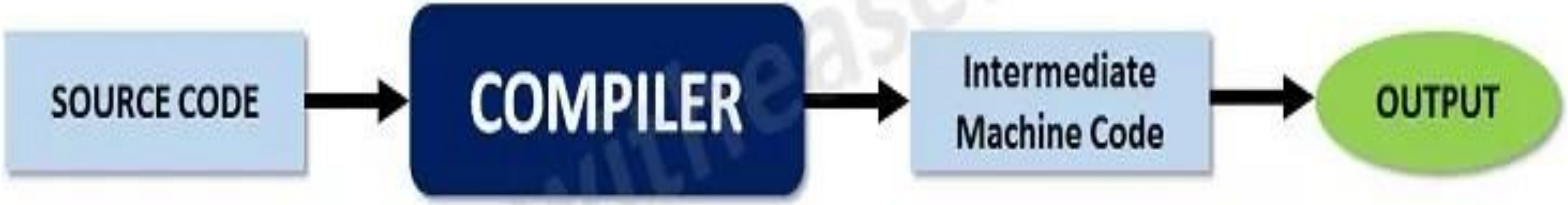
- Conditional compilation is running or avoiding a block of code after checking if a macro is defined or not (a constant value or an expression defined using #define). The preprocessor replaces all the conditional compilation directives with some pre-defined assembly code and passes a newly expanded file to the compiler. Conditional compilation can be performed using commands like #ifdef, #endif, #ifndef, #if, #else and #elif in a C Program.

2. Compiling

- Compiling phase in C uses an inbuilt compiler software to convert the intermediate (.i) file into an Assembly file (.s) having assembly level instructions (low-level code). To boost the performance of the program C compiler translates the intermediate file to make an assembly file. Assembly code is a simple English-type language used to write low-level instructions (in micro-controller programs, we use assembly language). The whole program code is parsed (syntax analysis) by the compiler software in one go, and it tells us about any syntax errors or warnings present in the source code through the terminal window.


Header files
C code
C Preprocessor
– includes
– defines
– ifdefs
gcc –E
"pure"
C code
C Compiler
– parse code
– generate assembly
gcc –S
assembly
code
Assembler
– generate object
code
gcc –c
object
code
Linker
– link code
together
Code
Libraries
executable
program

How Compiler works?



3. Assembling

- Compiling phase in C uses an inbuilt compiler software to convert the intermediate (.i) file into an Assembly file (.s) having assembly level instructions (low-level code). To boost the performance of the program C compiler translates the intermediate file to make an assembly file. Assembly code is a simple English-type language used to write low-level instructions (in micro-controller programs, we use assembly language). The whole program code is parsed (syntax analysis) by the compiler software in one go, and it tells us about any syntax errors or warnings present in the source code through the terminal window.

4. Linking

- Linking is a process of including the library files into our program. Library Files are some predefined files that contain the definition of the functions in the machine language and these files have an extension of .lib. Some unknown statements are written in the object (.o/.obj) file that our operating system can't understand. You can understand this as a book having some words that you don't know, and you will use a dictionary to find the meaning of those words. Similarly, we use Library Files to give meaning to some unknown statements from our object file. The linking process generates an executable file with an extension of .exe in DOS and .out in UNIX OS.

Conclusion

- **Compilation process in C is also known as the process of converting Human Understandable Code (C Program) into a Machine Understandable Code (Binary Code))**
- **Compilation process in C involves four steps: pre-processing, compiling, assembling, and linking.**
- **The preprocessor tool helps in comments removal, macros expansion, file inclusion, and conditional compilation. These commands are executed in the first step of the compilation process.**

hitbullseye

Conclusion

- Compiler software helps boost the program's performance and translates the intermediate file to an assembly file.
- Assembler helps convert the assembly file into an object file containing machine-level code. Linker is used for linking the library file with the object file. It is the final step in compilation to generate an executable file.

Follow Ups

- It is advised to go through Google and search for the common commands useful and try to implement them on the GCC compiler.

THANK YOU