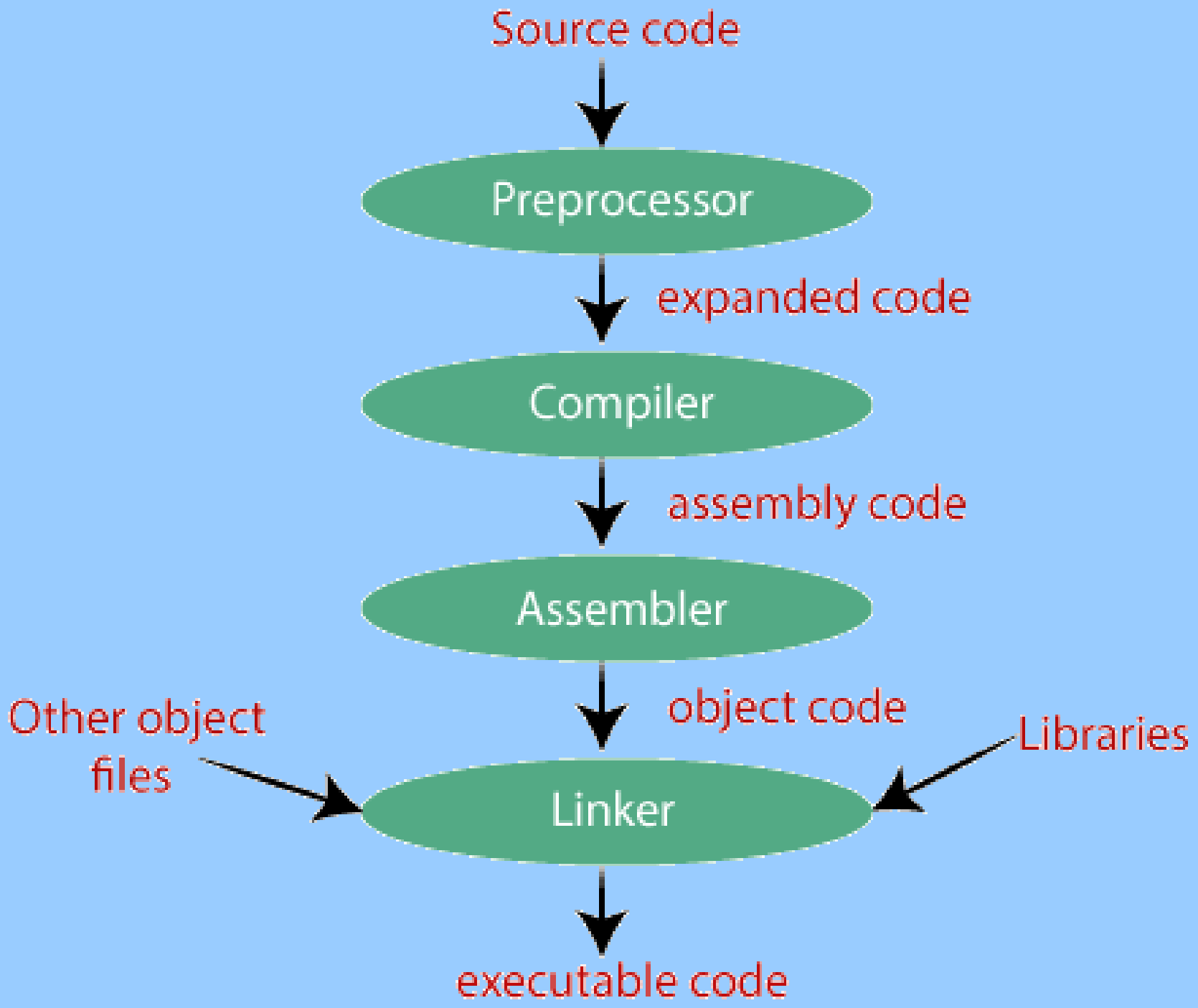
Compilation process

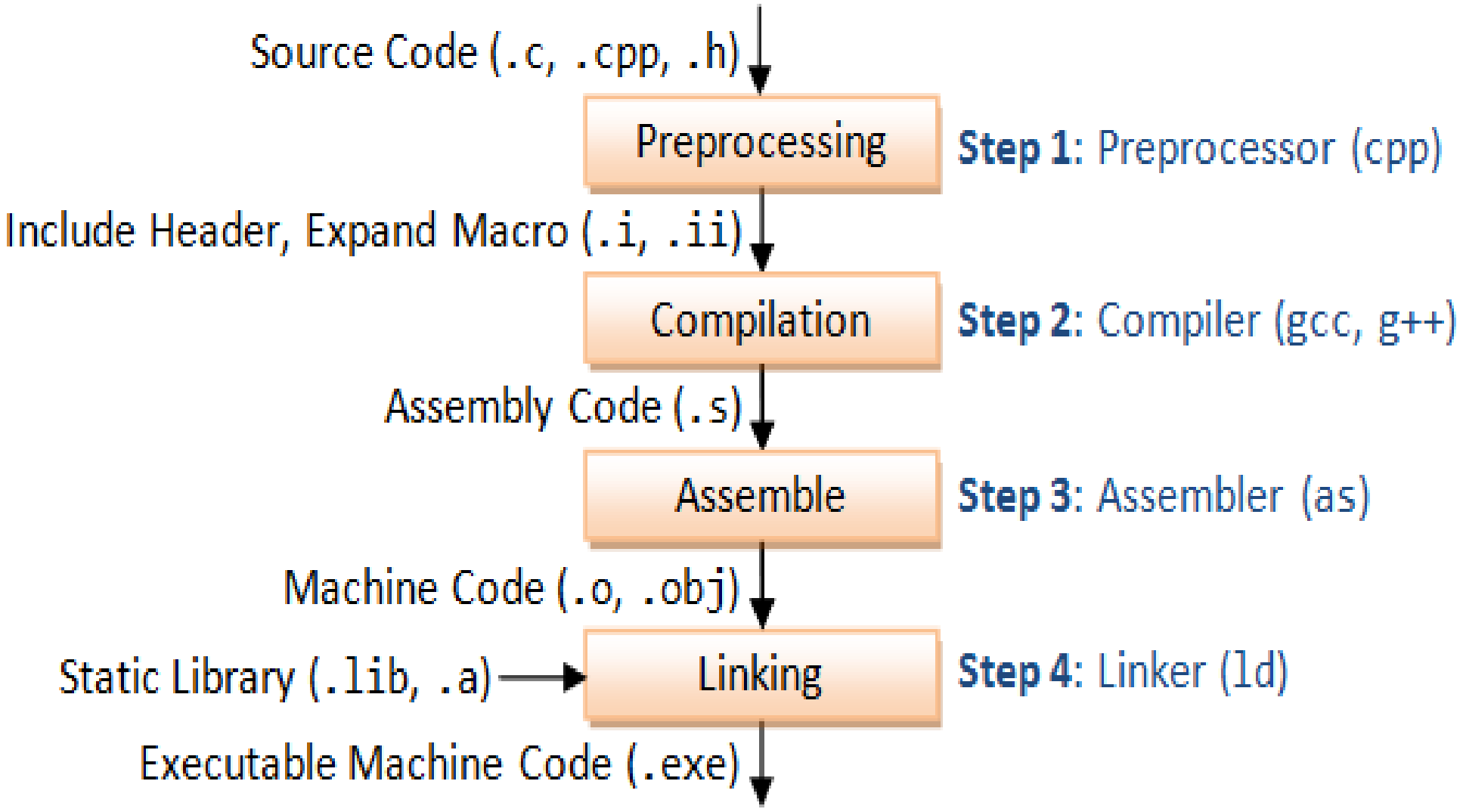
Break the command "gcc -o hello hello.c ..."

into preprocessing, compilation, assembly, and linking.

Input	→	Program	→	Output
Source code	→	Preprocessor	→	Expanded source
Expanded code	→	Compiler	→	Assembly source
Assembly source	→	Assembler	→	Object code
Object code	→	Linker	→	Executable code
Executable code	→	Loader	→	Execution


Source code
Preprocessor
expanded code
Compiler
assembly code
Assembler
object code
Other object files
Libraries
Linker
executable code

A More Detailed View



Compilation process

Two programs: A simple program that adds two number (not including any library), and “Hello world” with i/o library included.

Example1: Source program

```
int test_fun(int x){
  return x + 17;
}
int main(void){
  int x = 1;
  int y;
  y = test_fun(x);
  return 0;
}
```

Example2: Source program

```
#include <stdio.h>

int main (void){
  printf ("Hello, world!\n");
  return 0;
}
```

Compilation process

Step 1 (**Preprocessing**): cpp hello.c hello.i

Step 2 (**Compilation**): gcc -S hello.i -o hello.s

Step 3 (**Assembly**): as hello.s -o hello.o
The file contains the machine code for program hello

Step 4 (**Linking**): gcc hello.o (for linking) and produces a.out

Step 5 (**Execution**): ./a.out (to load and execute)

Hello, world!

To See Files Generated At Various Stages

Preprocessing

gcc -E main.c > preprocessor

cat preprocessor

```
#include <stdio.h>

/**
 * main - Prints a message
 *
 * Return: 0
 */
int main(void)
{
   prtinf("Hello World\n");
   return (0);
}
```



```
code of stdio.h file
int main(void)
{
   prtinf("Hello World\n");
   return (0);
}
```

Compilation

gcc -S main.c

cat main.s

```
    .file   "main.c"
        .text
        .globl  main
        .type   main, @function
    main:
    .LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
    .LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609"
        .section    .note.GNU-stack,"",@progbits
```

Assembler

gcc -c main.c

cat -v main.o | head

```
^ELF^B^A^A^@^@^@^@^@^@^@^@^@^A^@>^@^A^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^P^B^@^@^@^@^@^@^@^@^@@^@^@^@^@^@@^@^K^
@^H^@UHM-^IM-eM-8^@^@^@^@]M-C^@GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0
20160609^@^T^@^@^@^@^@^@^AzR^@^Ax^P^A^[^L^G^HM-^P^A^@^@^\
^@^@^@^\^@^@^@^@^@^@^@^K^@^@^@^@A^N^PM-
^F^BC^M^FF^L^G^H^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^A^@^@^@^D^@M-qM-^?
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^A^@^@^@^@^
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^B^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^C^@^C^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^C^@^E^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^C^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^C^@^D
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^H^@^@^@^R^@^A^@^@^@^@^
@^@^@^@^@^K^@^@^@^@^@^@^@^@main.c^@main^@^@^@^@
```

bin: The output executables go here, both for the app and for any tests and spikes.

build: This folder contains all object files, and is removed on a clean.

doc: Any notes, like assembly notes and configuration files, are here.

include: All project header files. All necessary third-party header files that do not exist under /usr/local/include are also placed here.

lib: Any libs that get compiled by the project, third party or any needed in development. Prior to deployment, third party libraries get moved to /usr/local/lib where they belong, leaving the project clean enough to compile on our Linux deployment servers. It cn be used to test different library versions than the standard.

spike: If one writes smaller classes or files to test technologies or ideas, and keep them around for future reference. They go here, where they do not dilute the real application’s files, but can still be found later.

src: The application and only the application’s source files.

test: All test code files. You do write tests, no?