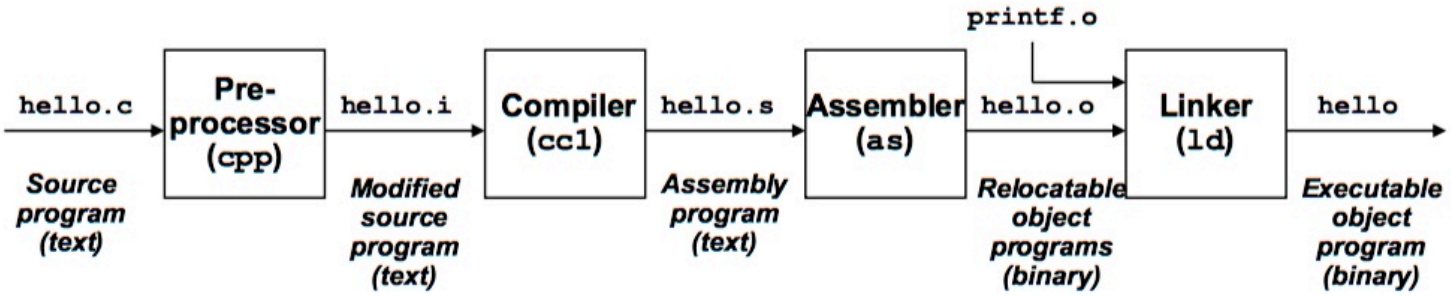
Compilation

```c
// Sample program
#include <stdio.h>

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



Preprocessor Phase

- preprocessor (cpp) modifies the origianl C program according to preprocessor **directives**
  - Ex: `#include <stdio.h>`
- “Read the contents of stdio.h and insert it directly into the program text”
- creates another C program, typically with the **.i** suffix

Compilation Phase

- compiler (cc1) translates **`hello.i`** into the text file **`hello.s`** containing an *assembly-language program*
- each line below describes **one** low-level machine-language instruction in textual form

```
main:
    subq    $8, %rsp
    movl    $.LC0, %edi
    call    puts
    movl    $0, %eax
    addq    $8, %rsp
    ret
```

Assembly Phase

- assembler (as) translates **`hello.s`** into machine-language instructions
    - packages them in a form known as a *relocatable object program*
    - stores result in object file **`hello.c`**
- this file is a binary file containing 17 bytes to encode the instructions for function main

Linking Phase

- **`hello`** calls the `printf()` function (part of the *standard C library*)
- this functions resides in a separate precompiled object file called **`printf.o`**
- must be merged with our hello.c program
- linker (ld) handles the merging
- result is the executable object file, **`hello`**, that is ready to be loaded into memory and executed by the system