5/2

Last Class → Signals.

Today.

Source file (s) ⟶ Executable File.
compilation system

hello.c

Preprocessor.

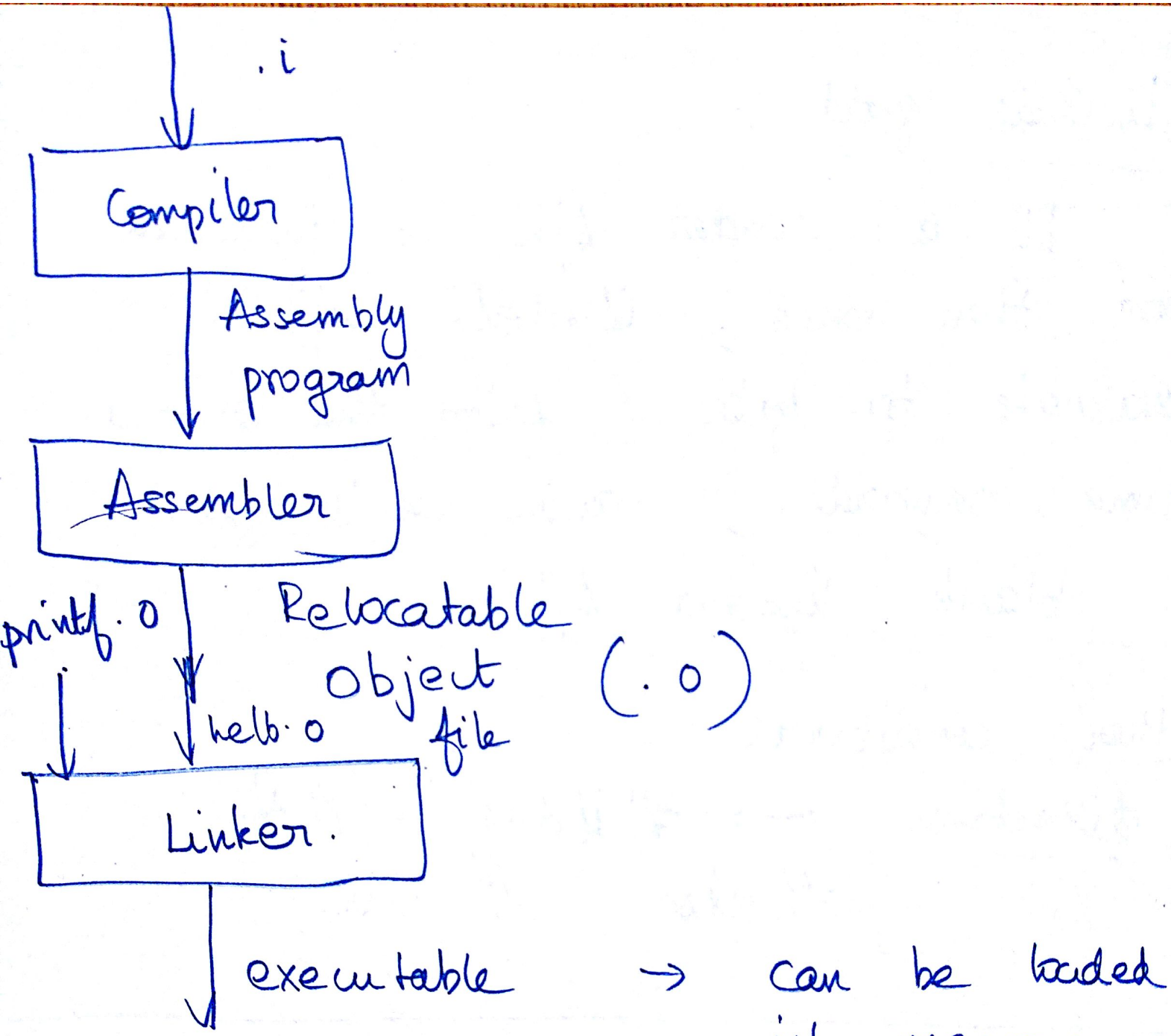
What does this do?

① Macro Substitution.

② Include the header files

③ Remove the comments.

a fragment code that has been given a name.



→ can be loaded into memory and executed as a process.

Two "Asides"

① Preprocessor directives //

(P6).

```
# ifndef HEADER_H
# define HEADER_H

int sum (int a, int b);

# endif
```

empty macro.

will process statements here only if HEADER_H ~~has to be~~ is not defined.

So that we don't define any identifiers more than once.

include guard

Include guard

If a header file is included more than once, ifndef will evaluate to false (from the second time onwards) and we'll get a blank header file.

Other conditional directives → #ifdef #if #else #elif.

---

② Extern.

Difference between declaring / defining a function.

```
int sum (int a, int b);
```

↳ declaration

```
int sum (int a, int b)
{
    return a+b;
}
```

↳ definition.

What about variables?

```
int a = 3;
```

? ← declare / definition.

```
int b;
```

? → declaration / definition

```
b = 4;
```

↳ assignment /

compiler allocates memory for b

How can be just declare a variable?

compiler knows that variable by that type and name exists

BUT it does not need to allocate memory for it. (it is allocated elsewhere).

extern ! → Declare without defining.

extern int c ; // no space will be allocated.

C++

```
int sum ( int a, int b);
```

↳ declaration

```
int sum (int a, int b)
{
    return a+b;
}
```

↳ definition.

What about for variables?

```
int a = 3;
```

? ← declare / definition.

```
int b;
```

? → declaration / definition

```
b = 4;
```

↓ assignment /

compiler allocates memory for b