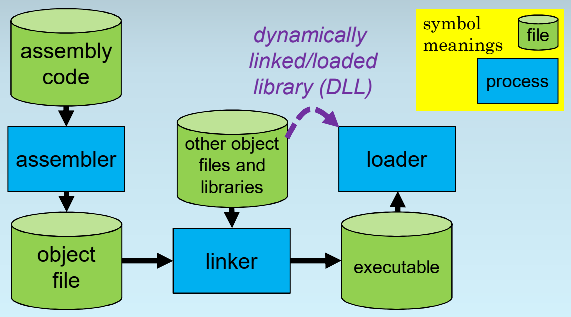
Process Same as Before with Assembly Code



A Compiler Turns Preprocessed Source into Assembly

But doesn't the compiler turn C code into an executable?

Actually, no.

As shown in the diagram, a compiler
- **turns preprocessed source code**
- (with header files incorporated,
- and macros expanded)
- **into assembly code**.

A Compiler Can Also Invoke Other Programs

A **compiler can also execute**
(by default, but optionally)
- a **preprocessor**,
- an **assembler**, and
- a **linker**.

What if you don't want all of the steps?*
- Use -E to obtain preprocessed output.
- Use -S to obtain assembly code.
- Use -c to obtain an object file (.o).

*These are the gcc options.

Too Many Possible Combinations of Language and ISA

Why are compilers built in two parts?

Imagine developing a compiler…
- languages: C, C++, Pascal, Java, and more
- ISAs: x86, ARM, PowerPC, Power, and more

Do you develop a separate compiler
- **for every language/ISA combination?**
- 10 languages, 10 ISAs → 100 compilers!

No.