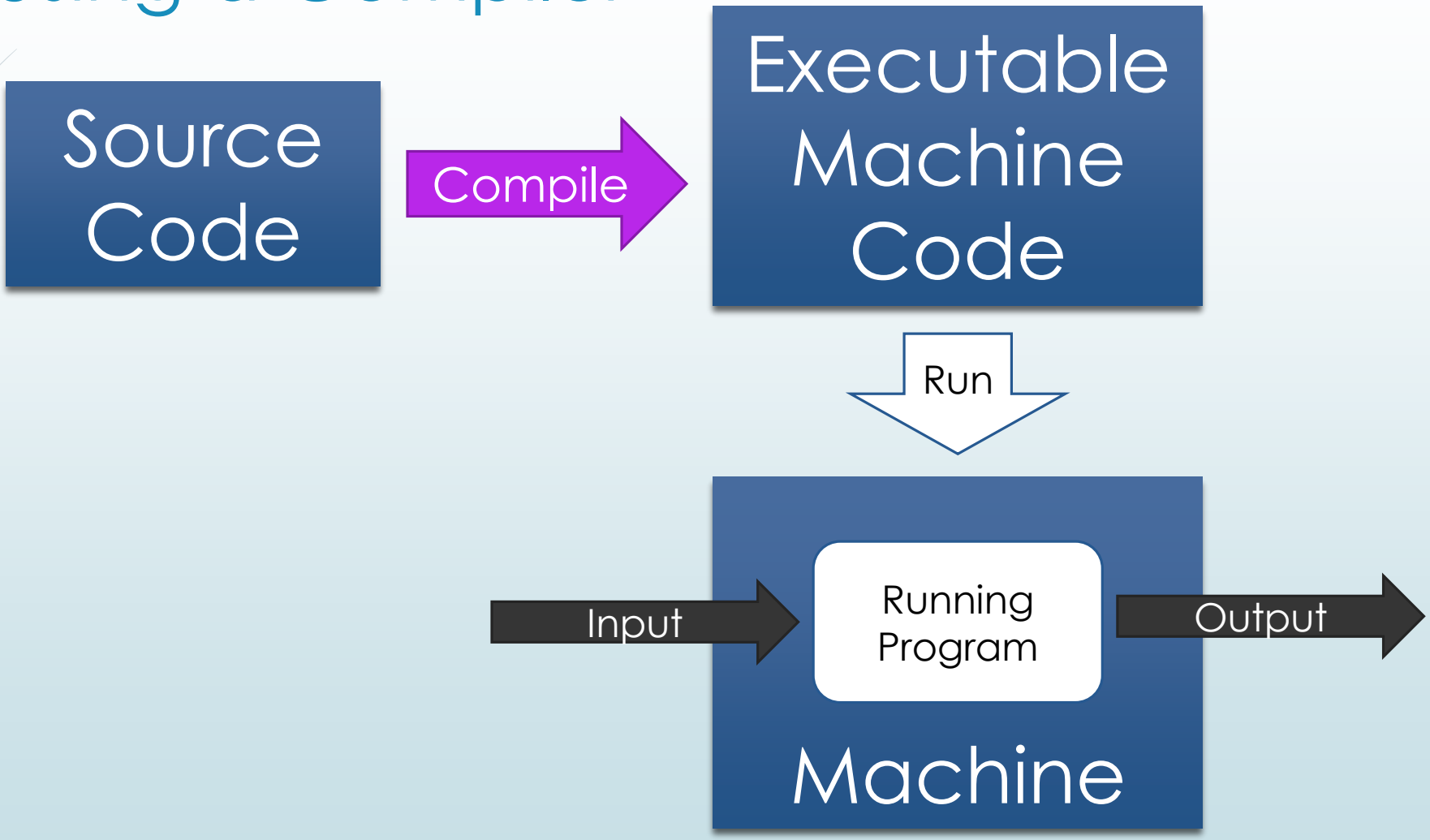
# EECS 280 – Lab 1

## The Standard Compilation Sequence



5/5/2016

# Using a Compiler



# The Compilation Sequence


1. Preprocessing
2. Compilation Proper
3. Assembly
4. Linking

# Preprocessing

- Takes care of any preprocessor directives
  - e.g. #include, #define

```
g++ -E stats.cpp -o hello.ii
```

# Compilation Proper

- Convert source into *assembly instructions*
  - This is the big one. It's quite complicated.

- Many languages (including C++) have separate compilation
  - Each source file is compiled independently

```
g++ -S hello.ii -o hello.s
g++ -S lib.ii -o lib.s
```

# Assembly

- Convert assembly instructions into a binary *object file*
- The code is not human-readable anymore!

```
g++ -c hello.s -o hello.o
g++ -c lib.s -o lib.o
```

# Linking

- Source files have been compiled separately until this point.
- Linking essentially connects the *definition* or *implementation* of a function with places where it is used.

```
g++ hello.o lib.o -o hello.exe
```

# Using g++

- If we use g++ without any special flags, but default the entire compilation process is performed.

```
g++ hello.cpp lib.cpp -o hello.exe
```