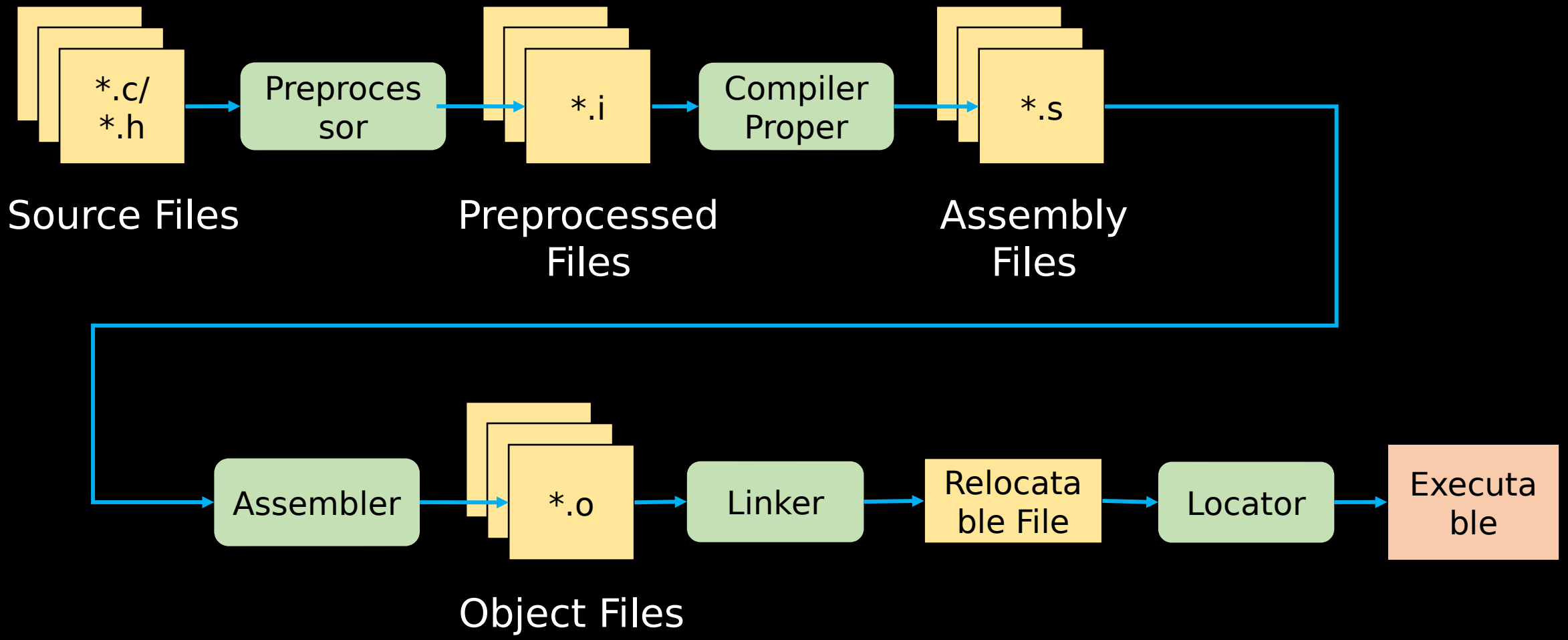
Embedded Software Essentials

Compiling and Invoking GCC

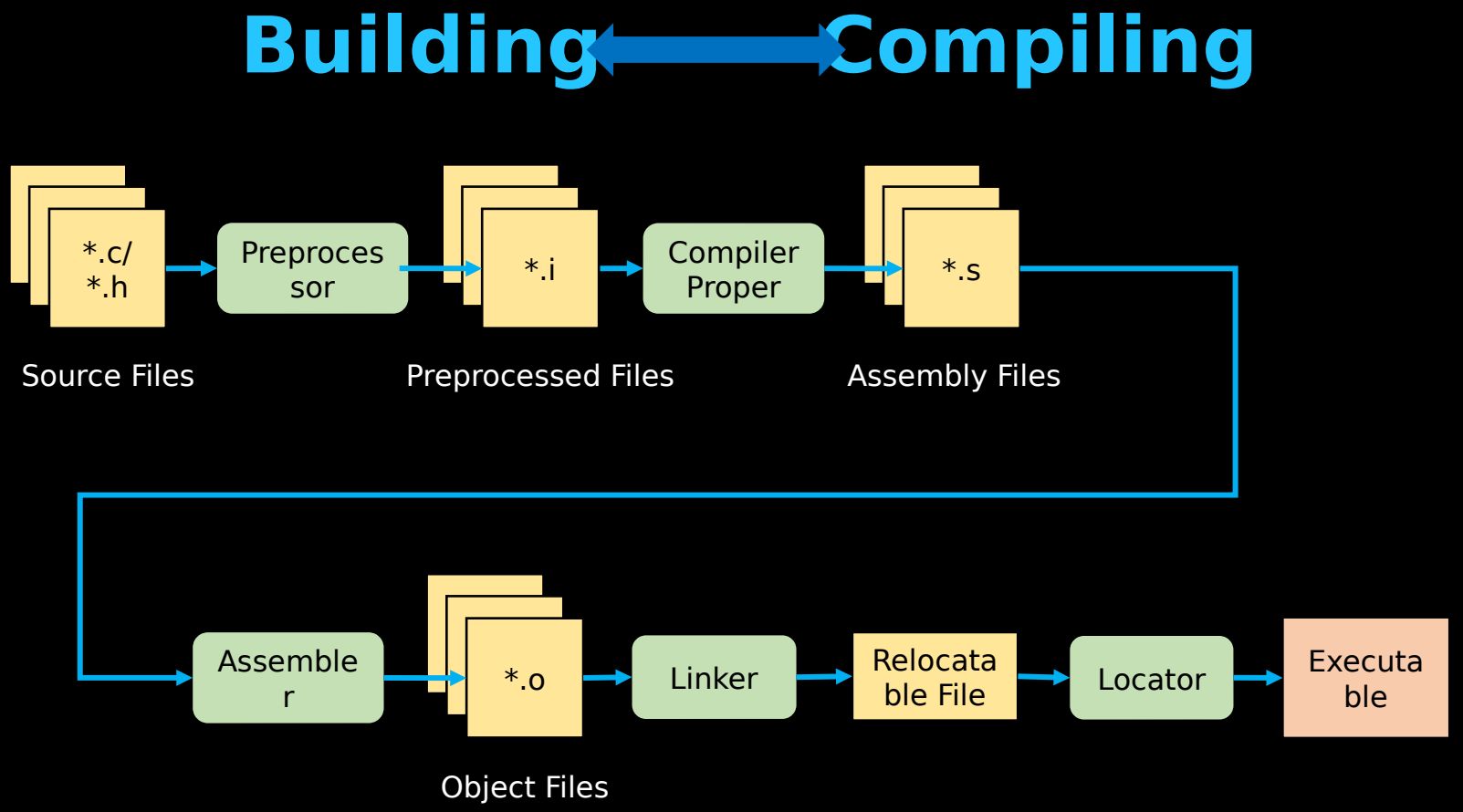
C1 M2 V2

Copyright

Build Process (linear)

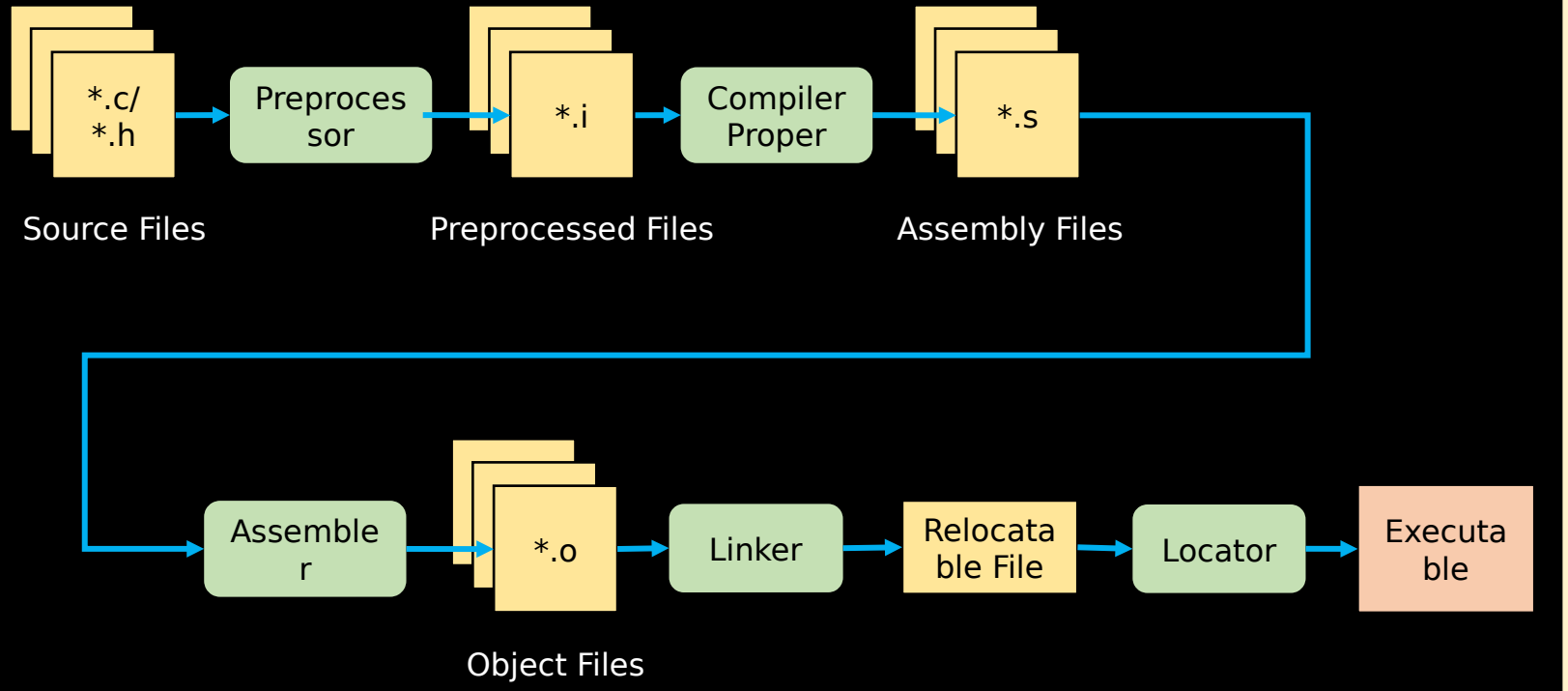


Build Process (linear)

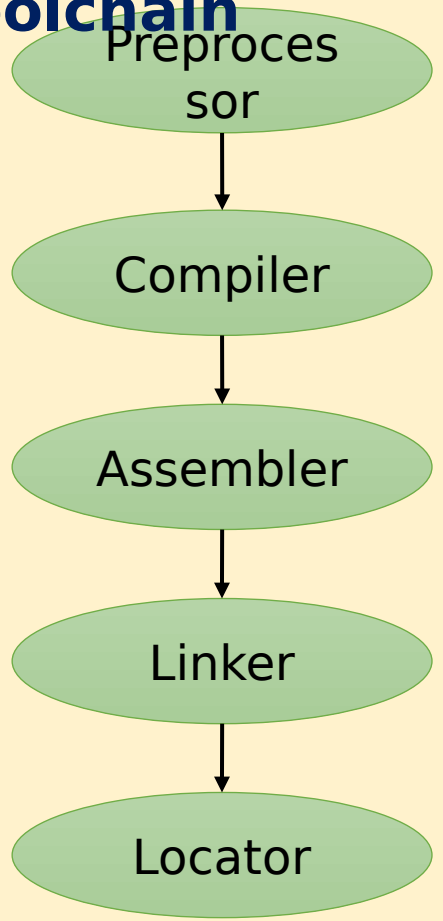


Build Process (linear)

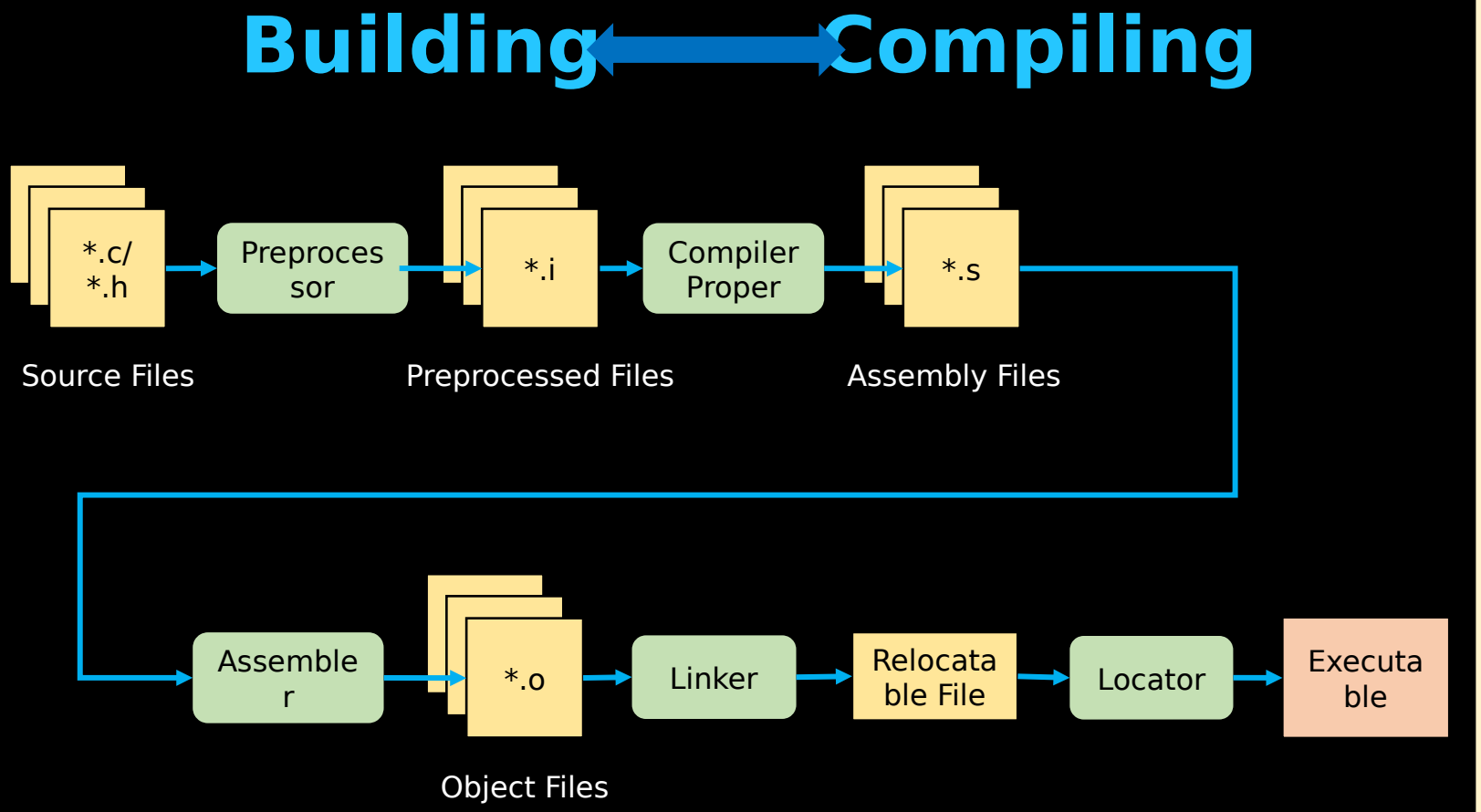
Building ⟷ **Compiling**

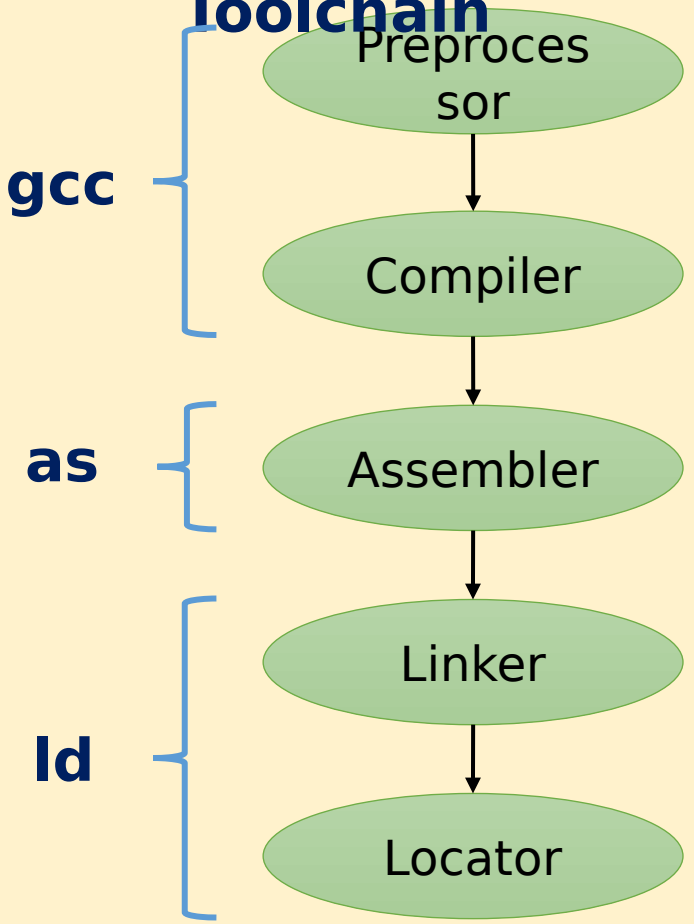


GNU's Compiler Collection (GCC) Toolchain



Build Process (linear)

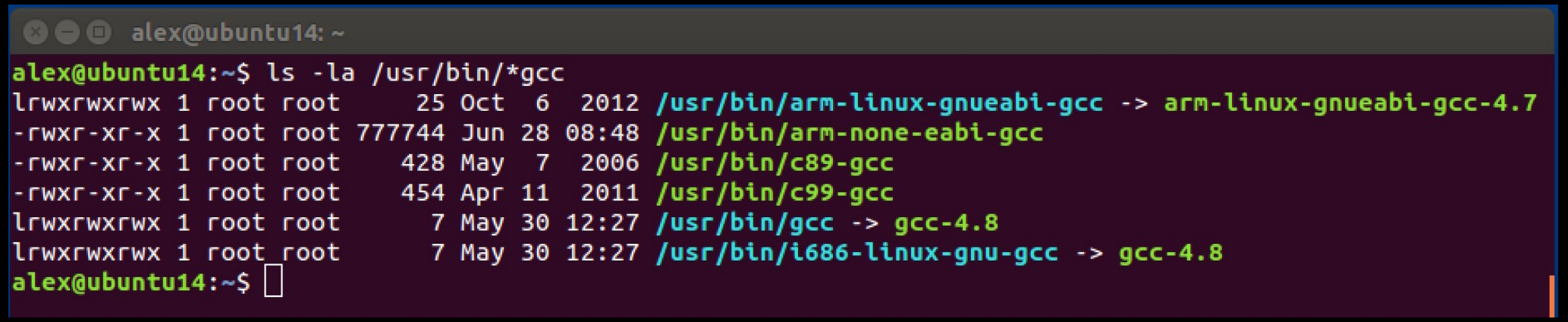




GCC Tool Check

Many compilers toolchains can be installed

```
$ ls –la /usr/bin/*gcc
```



<ARCH>-<VENDOR>-<OS>-<ABI>

arm-none-eabi-gcc	arm-linux-gnueabi-gcc
-Arch = ARM	-Arch = ARM
-Vendor = N/A	-Vendor = N/A
-OS = None (Bare-Metal)	-OS = Linux OS
-ABI = EABI	-ABI = GNUEABI

Compilers

Native Compiler:

- **gcc -> gcc-4.8**

For code run on the host machine

Cross Compiler:

- **arm-none-eabi-gcc**

For code run on the target processor

Compilers

Cross Compiler:

- **arm-none-eabi-gcc**

For code run on the target processor

Showd all tools in the Cross-Compiler Toolchain

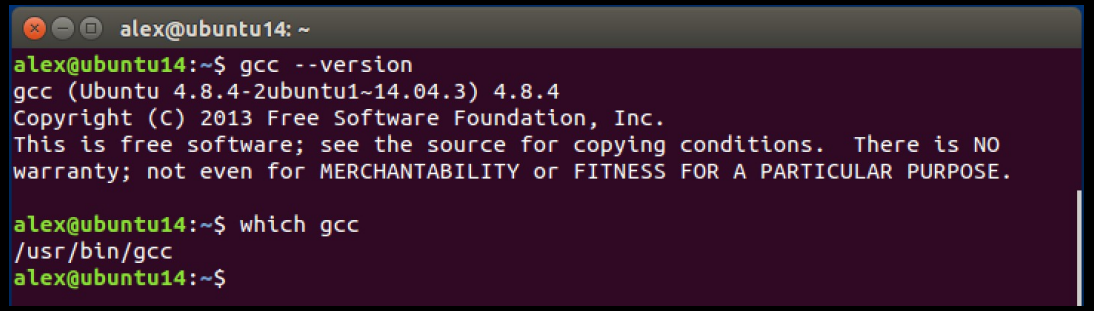
```
$ ls –la /usr/bin/arm-none-eabi*
```

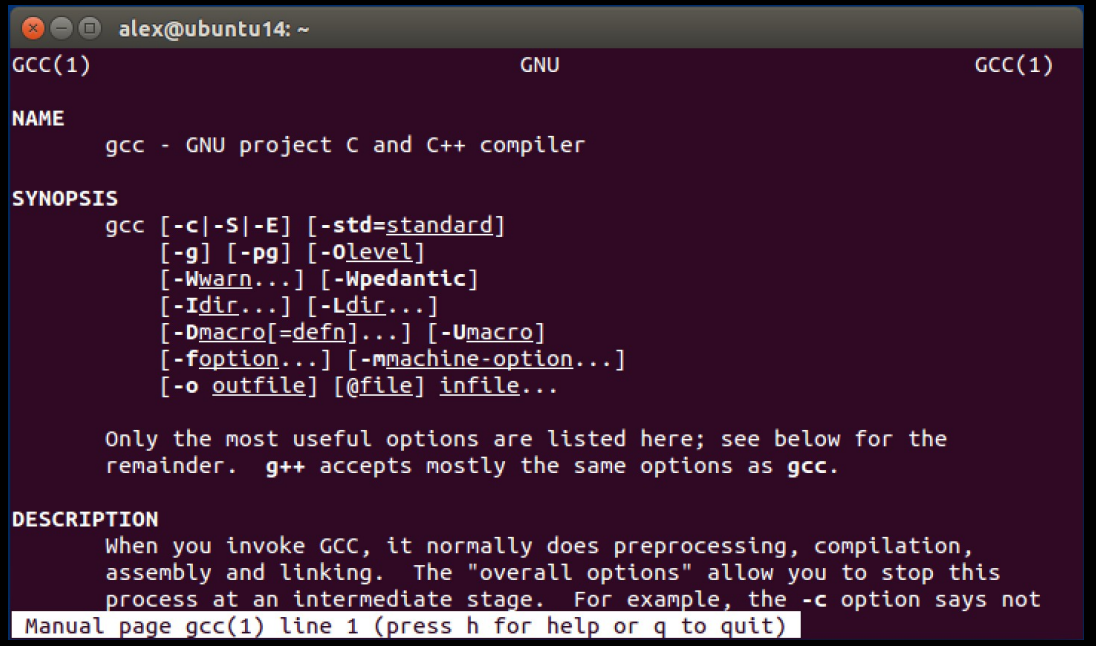
GCC Tool Check

```
$ gcc --version
```

```
$ which gcc
```

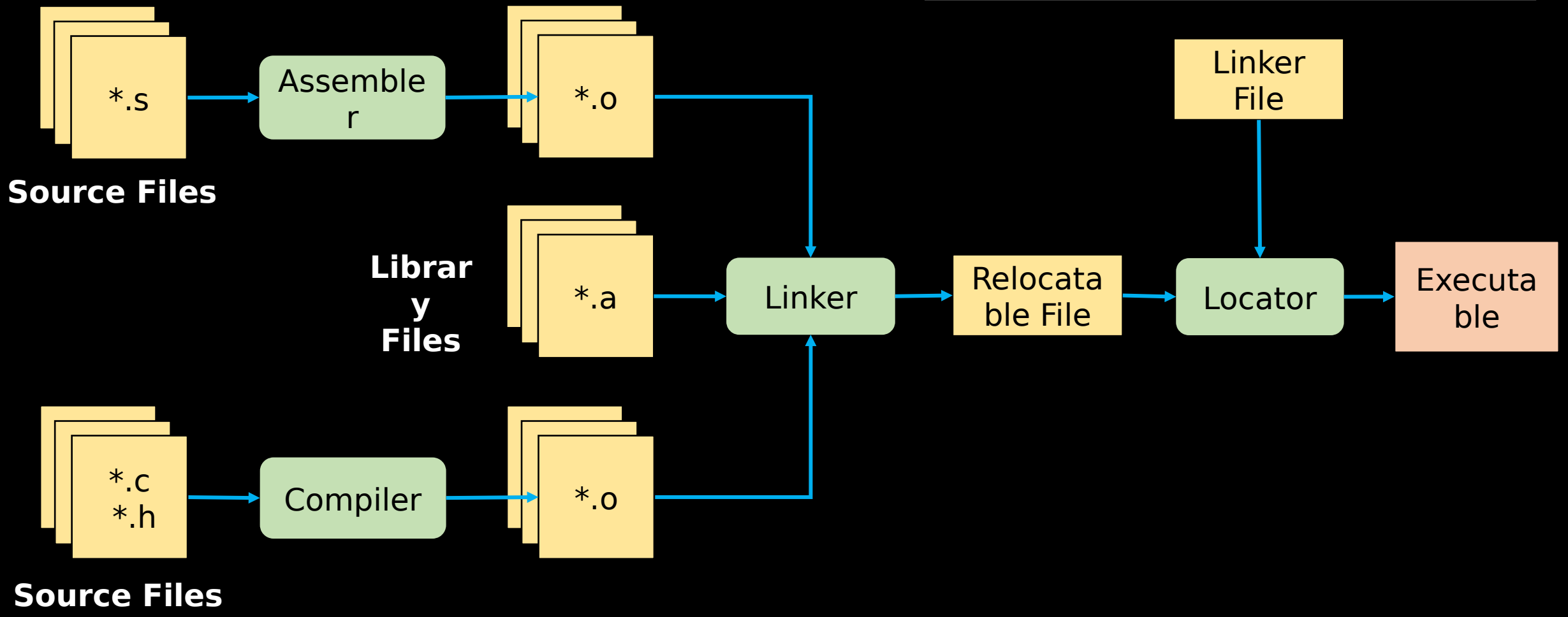
```
$ man gcc
```





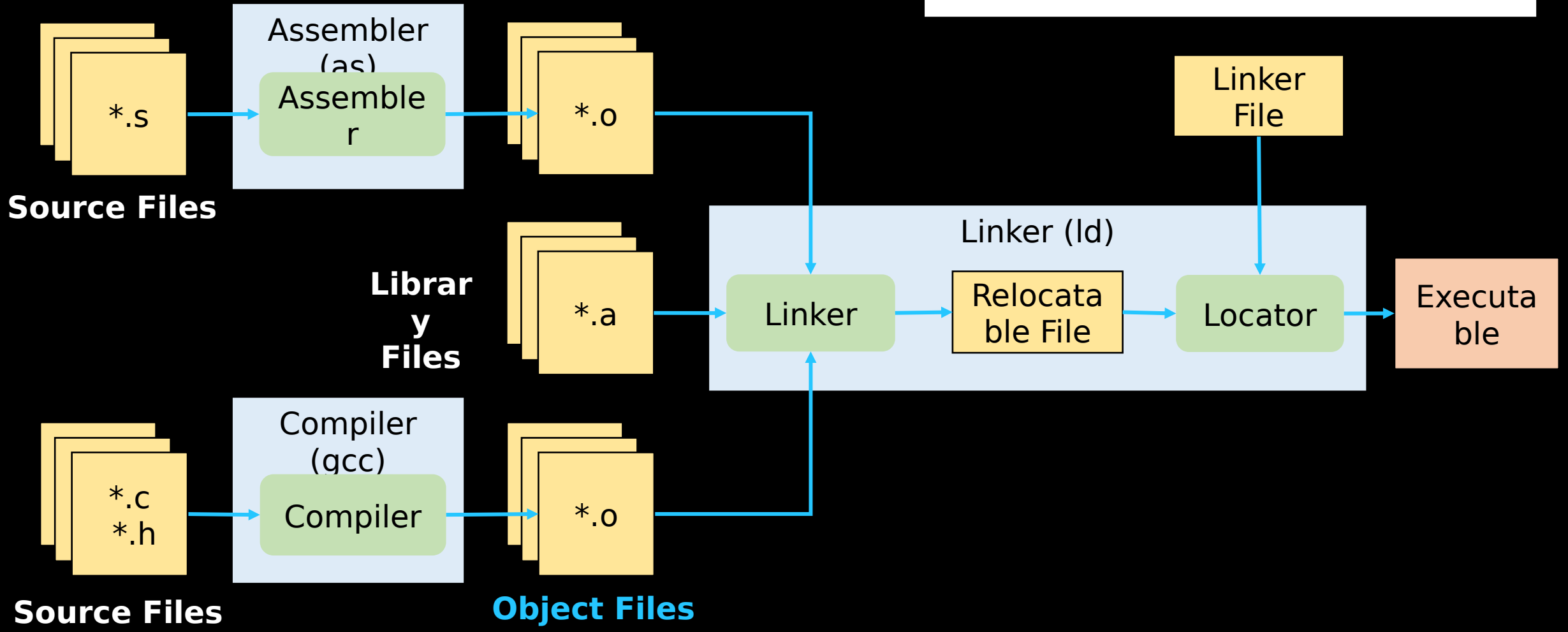
Typical Build Process

Assembly Files	.s Extension
✉	.o Extension
Object Files ✉	.a Extension
Library Files ✉	(with .h)
Executable File	Extension Varies
✉	



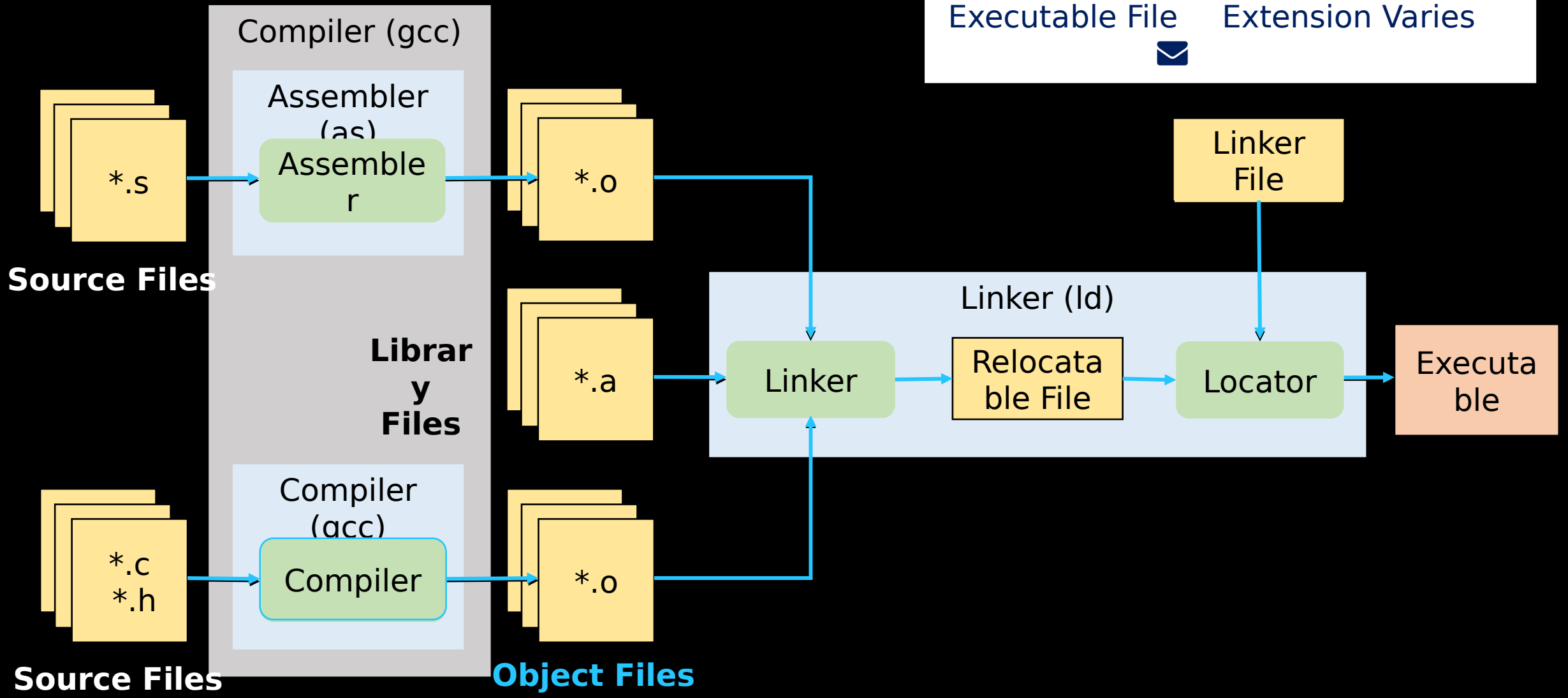
Typical Build Process

Assembly Files	.s Extension
Object Files	.o Extension
Library Files	.a Extension (with .h)
Executable File	Extension Varies



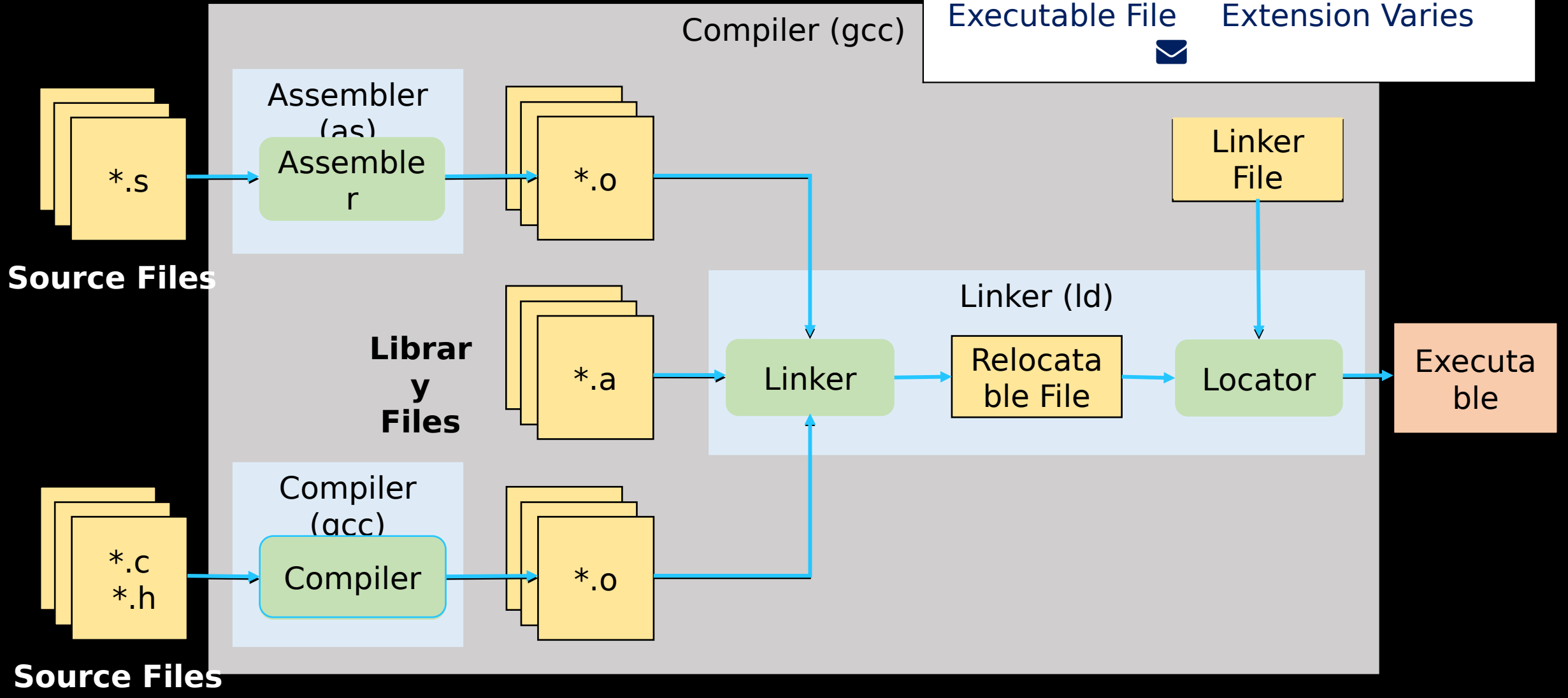
Typical Build Process

Assembly Files	.s Extension
Object Files	.o Extension
Library Files	.a Extension
Executable File	(with .h)
	Extension Varies



Typical Build Process

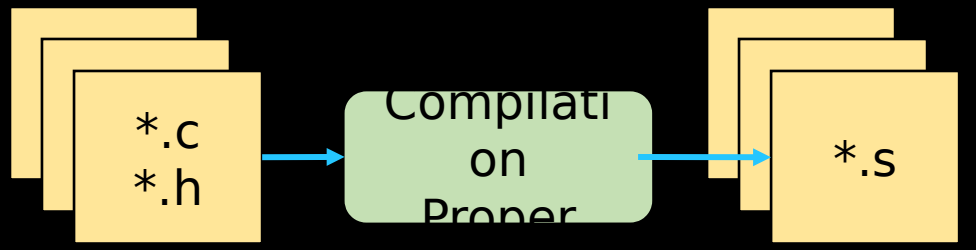
Assembly Files	.s Extension
Object Files	.o Extension
Library Files	.a Extension
Executable File	(with .h)
	Extension Varies



Compilation Proper

C-Programming (High Level Language)

```
int x = 0;
int y = 20;
int z = 5;
    …
while (y >= z) {
  y = y – z;
  x++;
}
```



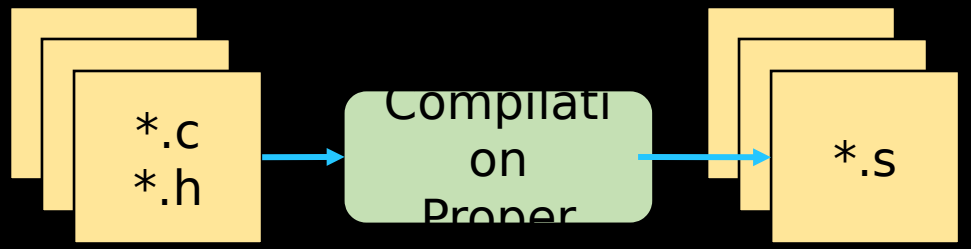
Source Files

Assembly Files

Compilation Proper

C-Programming (High Level Language)

```
int x = 0;
int y = 20;
int z = 5;
    …
while (y >= z) {
  y = y - z;
  x++;
}
```

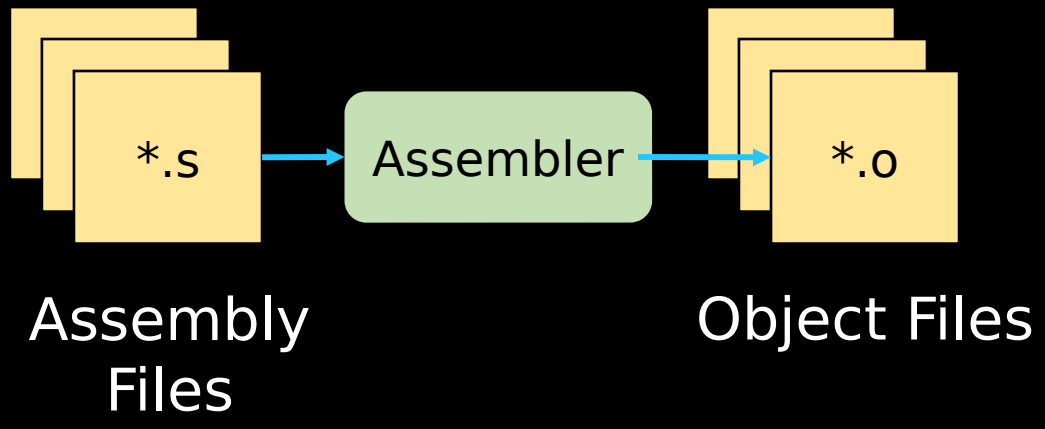


High level language translated to low level language via compiler

ARM Assembly Language

```
    ldr   r2, y
    ldr   r3, z
    ldr   r4, x
LOOP:
    sub   r2, r3
    inc   r4
    cmp   r2, r3
    bgt   LOOP
    str   r2, y
    str   r4, x
```

Assembly to Machine Code




```
movs    r3, #0
```

Operation **Operand**

Assembly Language	Machine Code
`movs r3, #0`	0x2300
`strb r3, [r7, #7]`	0x71fb
`adds r3, r7, #7`	0x1dfb
`str r3, [r7, #8]`	0x60bb
`adds r1, r7, #7`	0x1df9
`adds r3, r7, #6`	0x1dbb

General Compiler Flags

Option & Format	Purpose
`-c`	Compile and Assemble File, Do Not Link
`-o <FILE>`	Compile, Assemble, and Link to OUTPUT_FILE
`-g`	Generate Debugging Information in Executable
`-Wall`	Enable All Warning Messages
`-Werror`	Treat All Warnings as Errors
`-I<DIR>`	Include this <DIR> to Look for Header Files
`-ansi` `-std=STANDARD`	Specify Which Standard Version to Use (ex: c89,c99)
`-v`	Verbose Output from GCC

Architecture Specific Compiler Flags

Option & Format	Purpose
`-mcpu=[NAME]`	Specifies Target ARM Processor and Architecture (ex: cortex-m0plus)
`-march=[NAME]`	Target ARM Architecture (ex: armv7-m, thumb)
`-mtune=[NAME]`	Target ARM Processor (ex: cortex-m0plus)
`-mthumb`	Generate code in Thumb States (ISA)
`-marm`	Generate code in ARM State (ISA)
`-mthumb-interwork`	Generate code that supports calling between ARM and Thumb (ISA)
`-mlittle-endian`	Generate code for Little Endian Mode
`-mbig-endian`	Generate code for Big Endian Mode