

Embedded Software Essentials

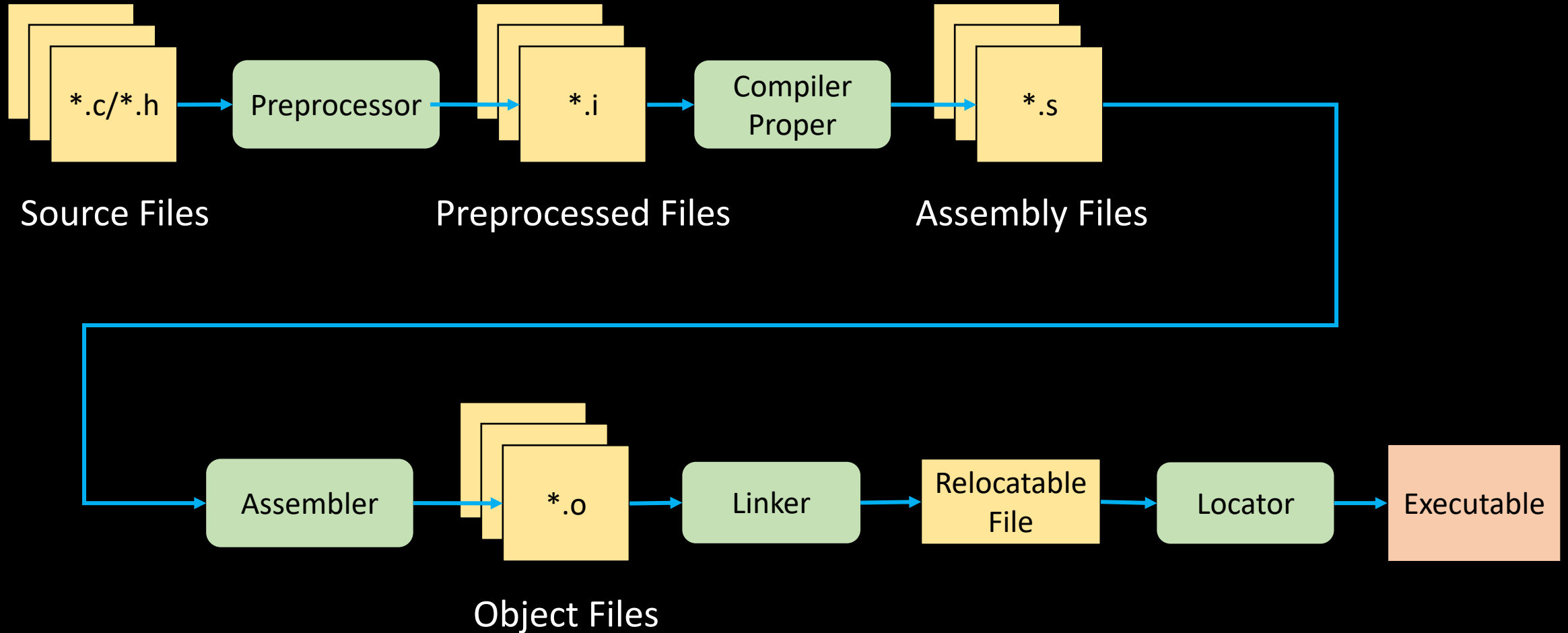
Compiling and Invoking GCC

C1 M2 V2

Copyright

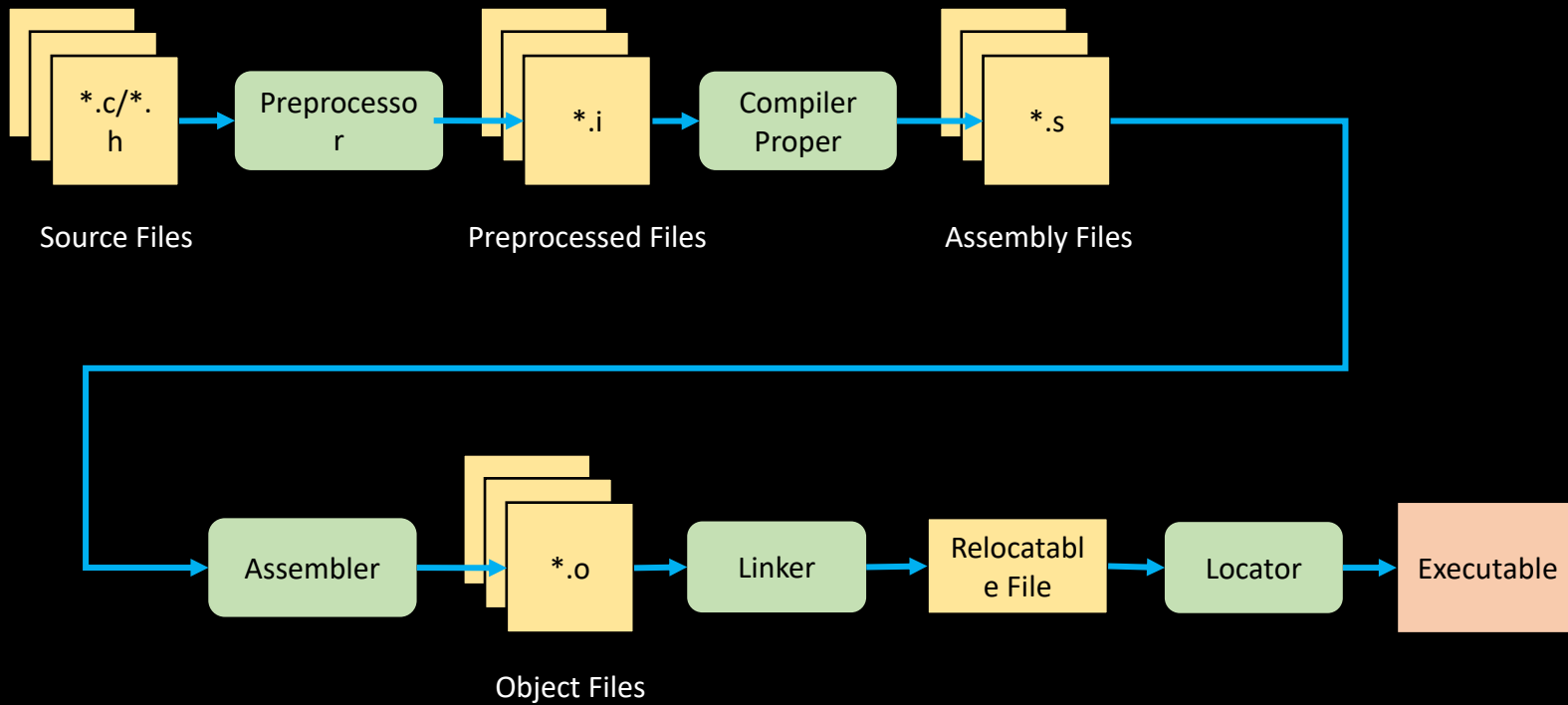
- Copyright (C) 2017 by Alex Fossdick. Redistribution, modification or use of this presentation is permitted as long as the files maintain this copyright. Users are permitted to modify this and use it to learn about the field of embedded software. Alex Fossdick and the University of Colorado are not liable for any misuse of this material.

Build Process (linear)



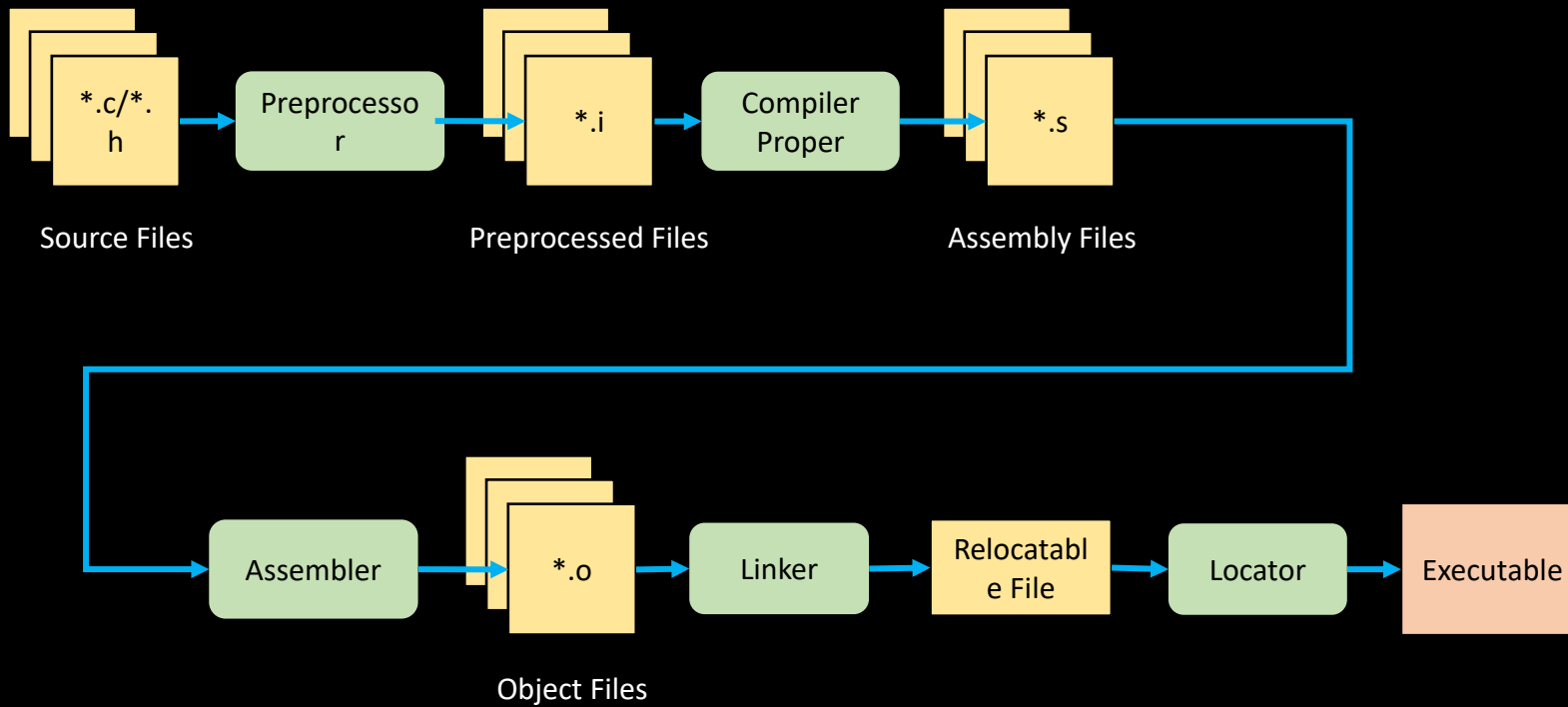
Build Process (linear)

Building ↔ **Compiling**

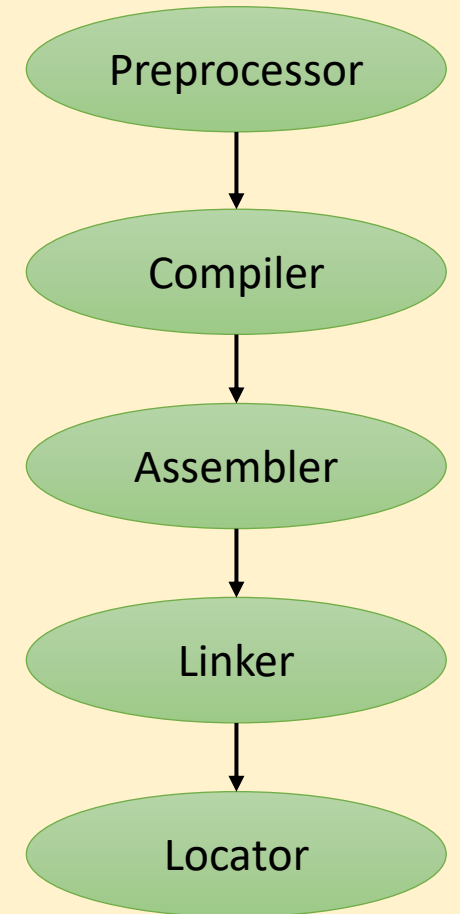


Build Process (linear)

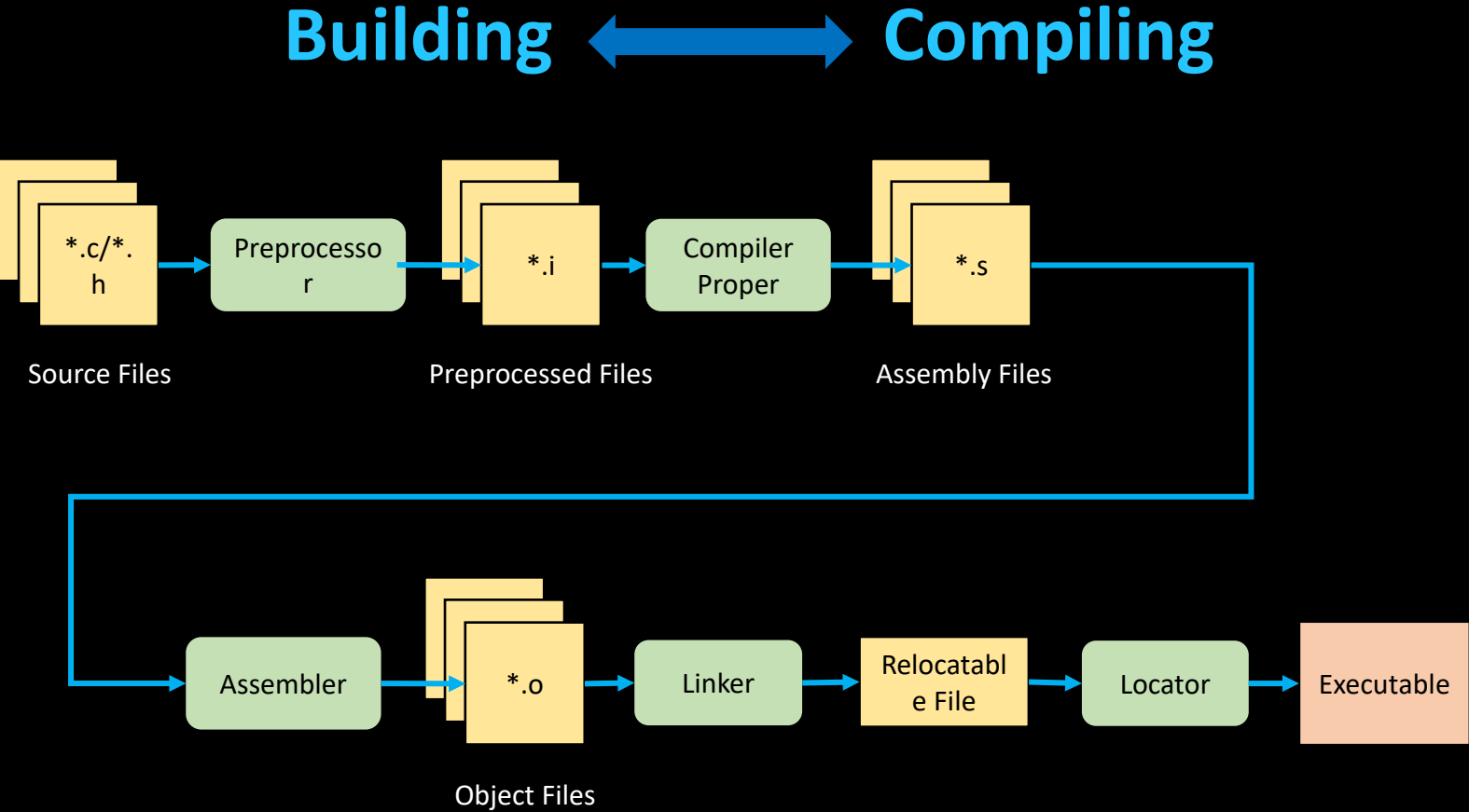
Building ↔ **Compiling**



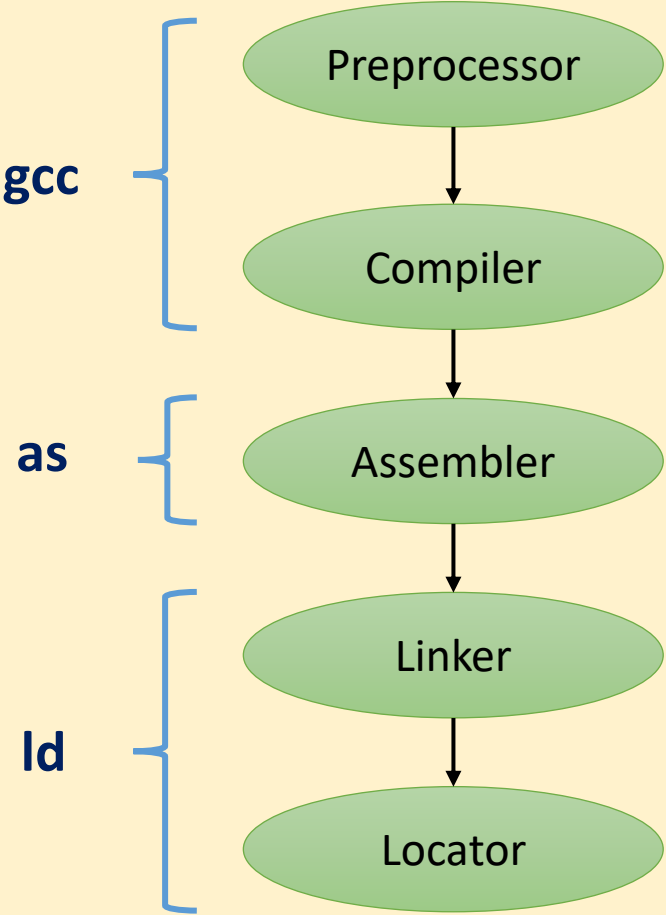
GNU's Compiler Collection (GCC) Toolchain



Build Process (linear)



GNU's Compiler Collection (GCC) Toolchain



GCC Tool Check

Many compilers toolchains can be installed

```
$ ls -la /usr/bin/*gcc
```

```
alex@ubuntu14: ~  
alex@ubuntu14:~$ ls -la /usr/bin/*gcc  
lrwxrwxrwx 1 root root      25 Oct  6  2012 /usr/bin/arm-linux-gnueabi-gcc -> arm-linux-gnueabi-gcc-4.7  
-rwxr-xr-x 1 root root 777744 Jun 28 08:48 /usr/bin/arm-none-eabi-gcc  
-rwxr-xr-x 1 root root   428 May  7  2006 /usr/bin/c89-gcc  
-rwxr-xr-x 1 root root   454 Apr 11  2011 /usr/bin/c99-gcc  
lrwxrwxrwx 1 root root      17 May 30 12:27 /usr/bin/gcc -> gcc-4.8  
lrwxrwxrwx 1 root root      37 May 30 12:27 /usr/bin/i686-linux-gnu-gcc -> gcc-4.8  
alex@ubuntu14:~$
```

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

arm-none-eabi-gcc

-Arch = ARM

-Vendor = N/A

-OS = None (Bare-Metal)

-ABI = EABI

arm-linux-gnueabi-gcc

-Arch = ARM

-Vendor = N/A

-OS = Linux OS

-ABI = GNUEABI

Compilers

```
alex@ubuntu14: ~  
alex@ubuntu14:~$ ls -la /usr/bin/*gcc  
lrwxrwxrwx 1 root root 25 Oct 6 2012 /usr/bin/arm-linux-gnueabi-gcc -> arm-linux-gnueabi-gcc-4.7  
-rwxr-xr-x 1 root root 777744 Jun 28 08:48 /usr/bin/arm-none-eabi-gcc  
-rwxr-xr-x 1 root root 428 May 7 2006 /usr/bin/c89-gcc  
-rwxr-xr-x 1 root root 454 Apr 11 2011 /usr/bin/c99-gcc  
lrwxrwxrwx 1 root root 7 May 30 12:27 /usr/bin/gcc -> gcc-4.8  
lrwxrwxrwx 1 root root 7 May 30 12:27 /usr/bin/i686-linux-gnu-gcc -> gcc-4.8  
alex@ubuntu14:~$
```

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

```
alex@ubuntu14: ~  
alex@ubuntu14:~$ ls -la /usr/bin/*gcc  
lrwxrwxrwx 1 root root 25 Oct 6 2012 /usr/bin/arm-linux-gnueabi-gcc -> arm-linux-gnueabi-gcc-4.7  
-rwxr-xr-x 1 root root 777744 Jun 28 08:48 /usr/bin/arm-none-eabi-gcc  
-rwxr-xr-x 1 root root 428 May 7 2006 /usr/bin/c89-gcc  
-rwxr-xr-x 1 root root 454 Apr 11 2011 /usr/bin/c99-gcc  
lrwxrwxrwx 1 root root 7 May 30 12:27 /usr/bin/gcc -> gcc-4.8  
lrwxrwxrwx 1 root root 7 May 30 12:27 /usr/bin/i686-linux-gnu-gcc -> gcc-4.8  
alex@ubuntu14:~$
```

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
```

```
alex@ubuntu14: ~  
alex@ubuntu14:~$ gcc --version  
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4  
Copyright (C) 2013 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
alex@ubuntu14:~$ which gcc  
/usr/bin/gcc  
alex@ubuntu14:~$
```

```
$ which gcc
```

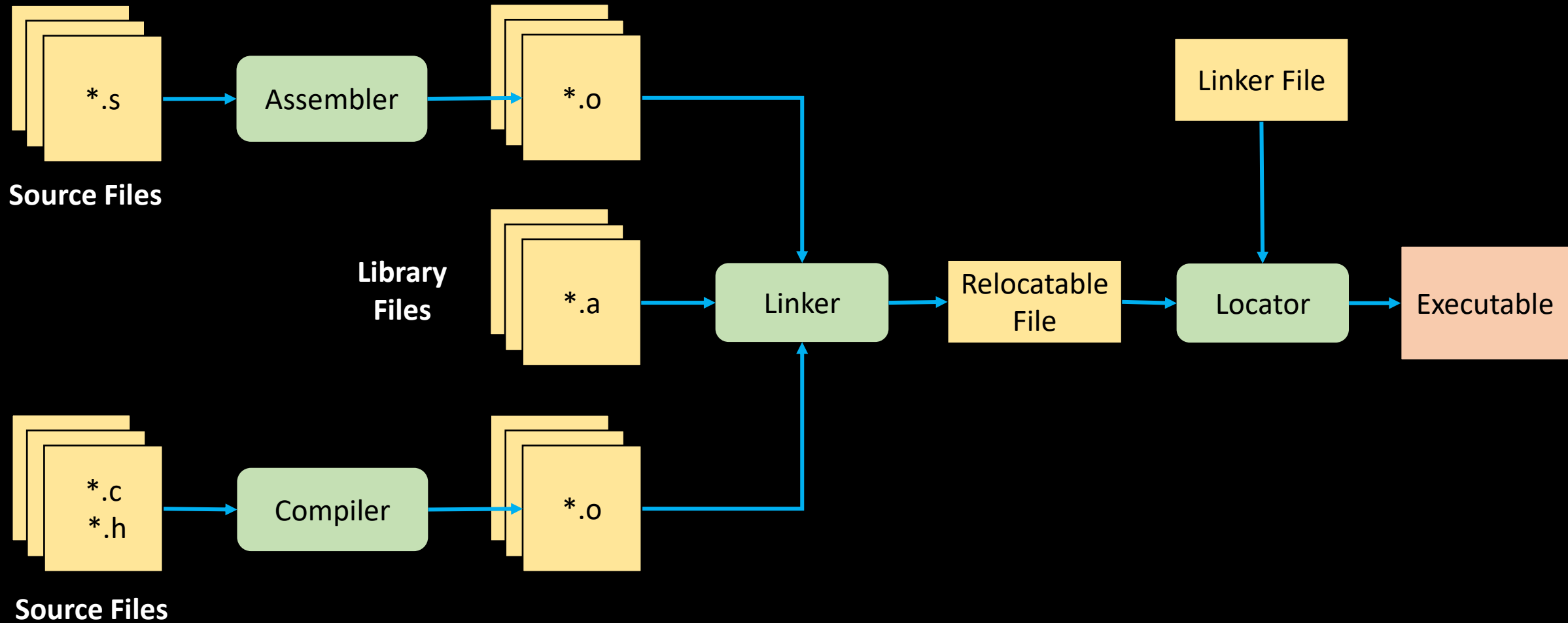
```
alex@ubuntu14: ~  
GCC(1)                                GNU                                GCC(1)  
  
NAME  
    gcc - GNU project C and C++ compiler  
  
SYNOPSIS  
    gcc [-c|-S|-E] [-std=standard]  
        [-g] [-pg] [-Olevel]  
        [-Wwarn...] [-Wpedantic]  
        [-Idir...] [-Ldir...]  
        [-Dmacro[=defn]...] [-Umacro]  
        [-foption...] [-mmachine-option...]  
        [-o outfile] [@file] infile...  
  
    Only the most useful options are listed here; see below for the  
    remainder.  g++ accepts mostly the same options as gcc.  
  
DESCRIPTION  
    When you invoke GCC, it normally does preprocessing, compilation,  
    assembly and linking.  The "overall options" allow you to stop this  
    process at an intermediate stage.  For example, the -c option says not  
    Manual page gcc(1) line 1 (press h for help or q to quit)
```

```
$ man gcc
```



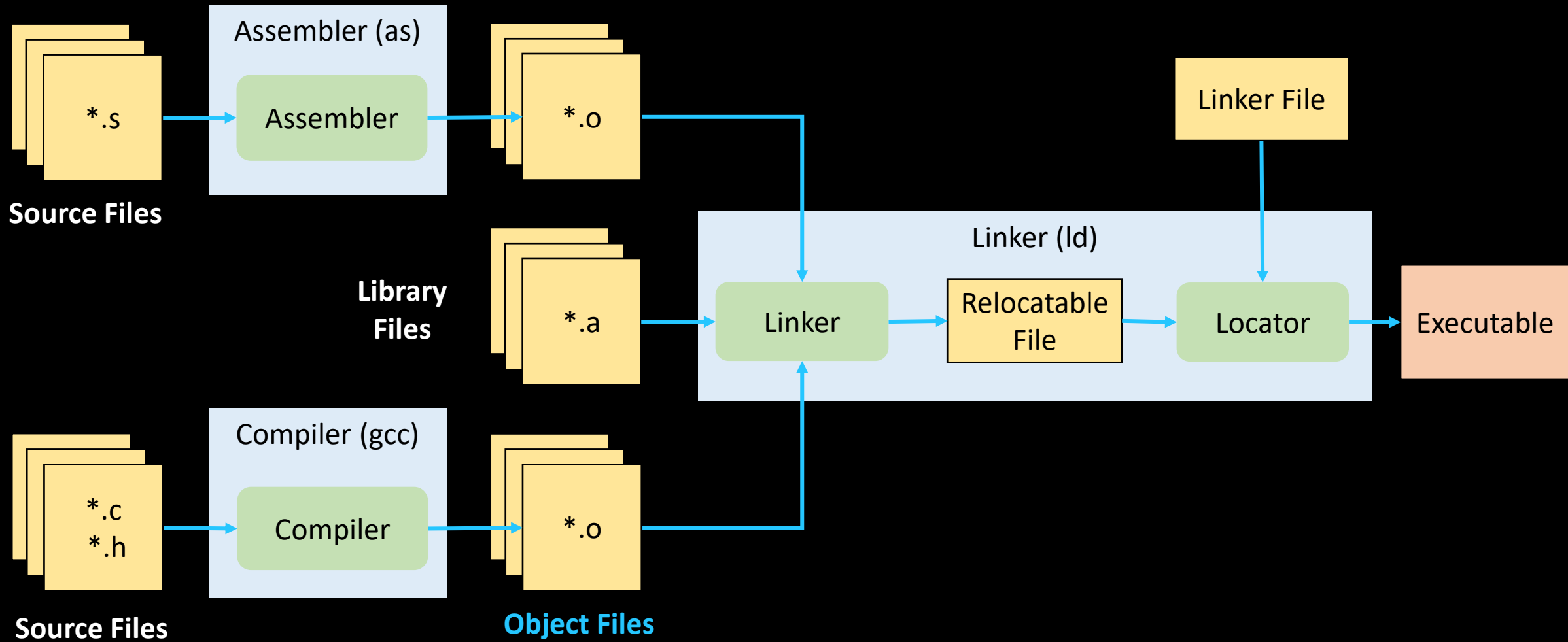
Typical Build Process

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

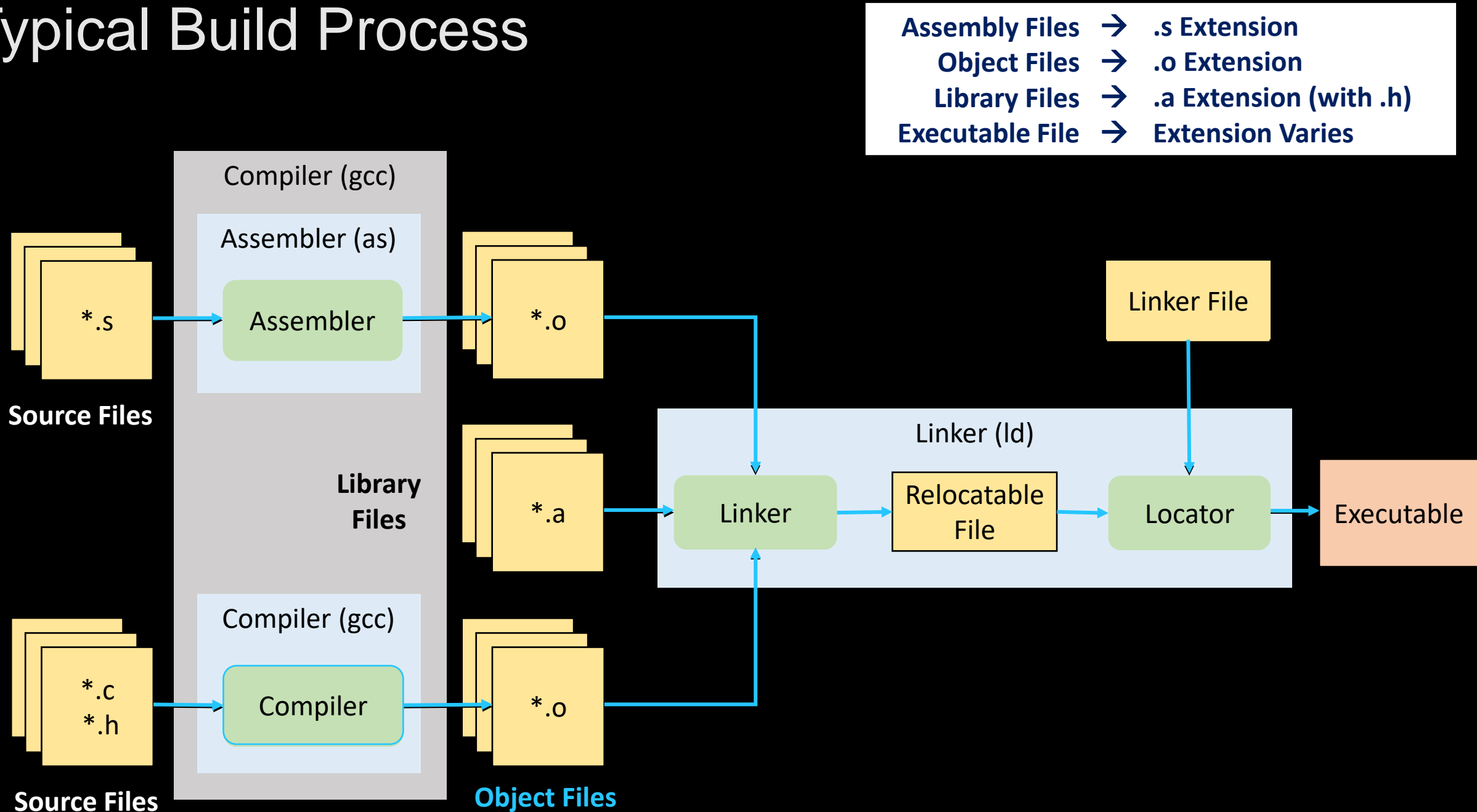


Typical Build Process

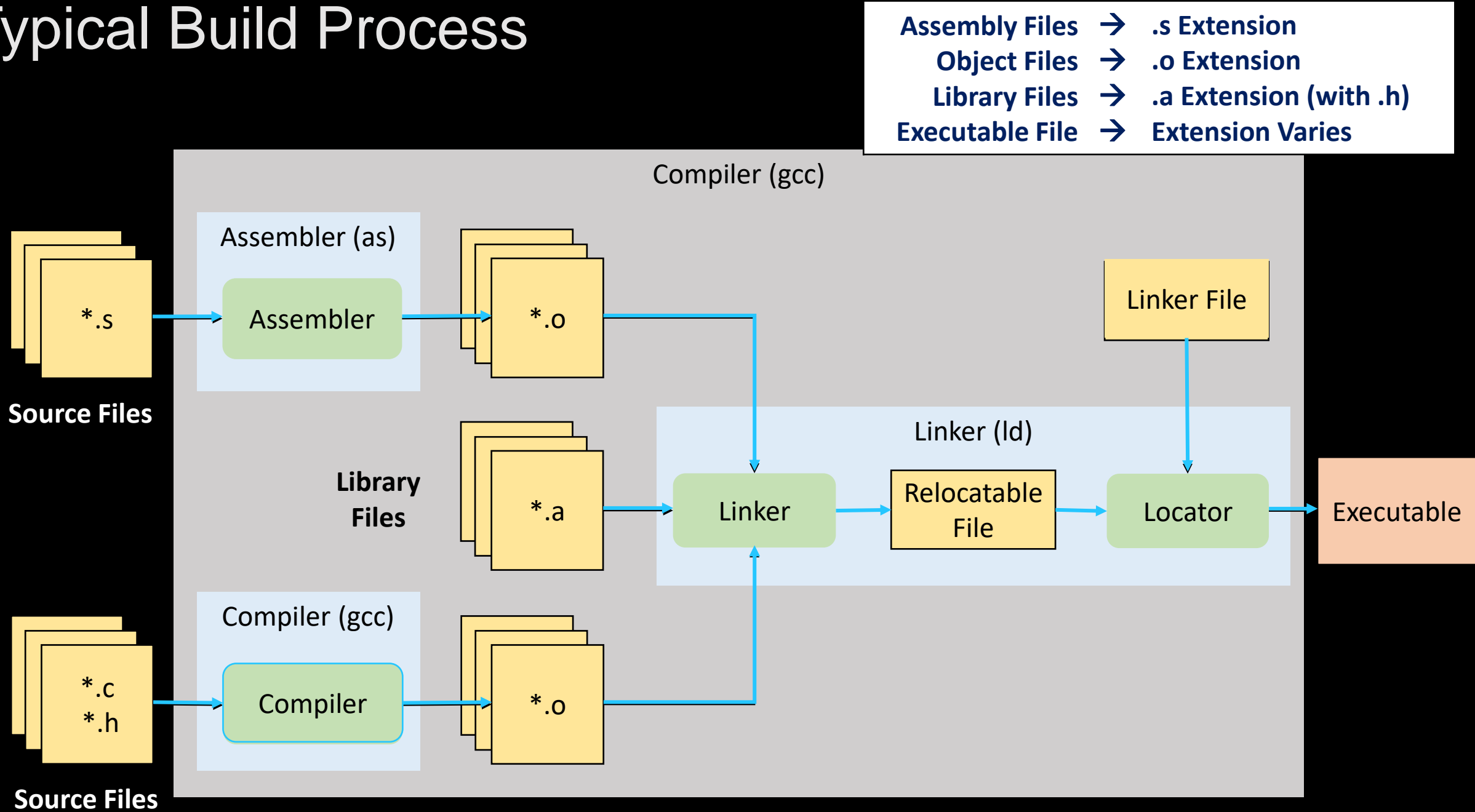
Assembly Files	→	.s Extension
Object Files	→	.o Extension
Library Files	→	.a Extension (with .h)
Executable File	→	Extension Varies



Typical Build Process



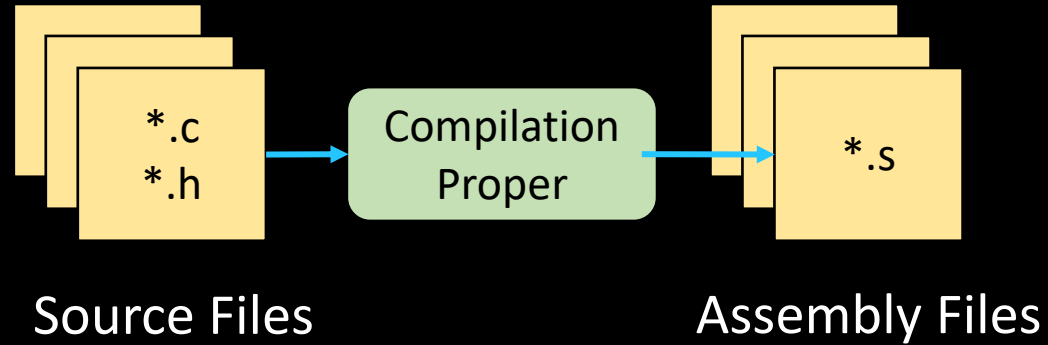
Typical Build Process



Compilation Proper

C-Programming (High Level Language)

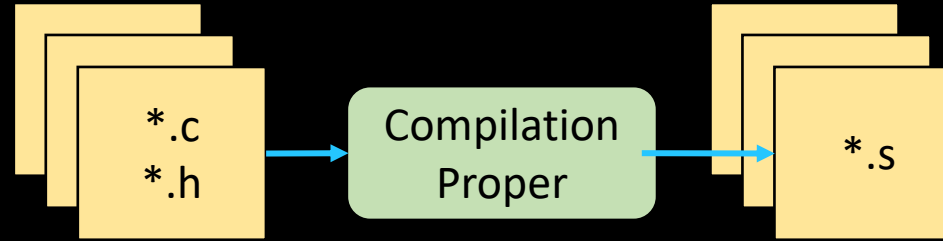
```
int x = 0;
int y = 20;
int z = 5;
...
while (y >= z) {
    y = y - z;
    x++;
}
```



Compilation Proper

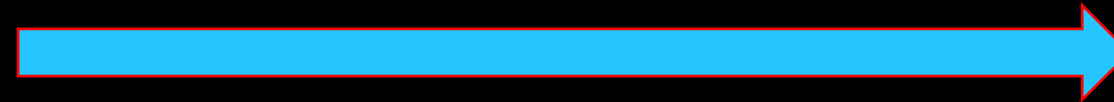
C-Programming (High Level Language)

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



Source Files

Assembly Files



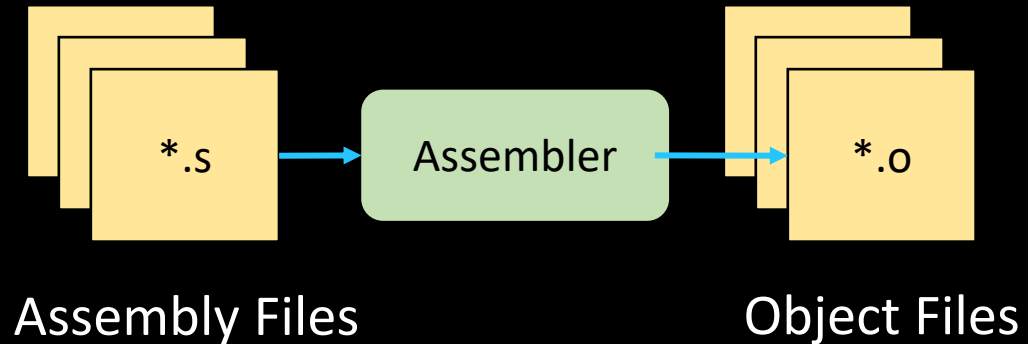
High level language
translated to low level
language via compiler

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
<code>movs r3, #0</code>	<code>0x2300</code>
<code>strb r3, [r7, #7]</code>	<code>0x71fb</code>
<code>adds r3, r7, #7</code>	<code>0x1dfb</code>
<code>str r3, [r7, #8]</code>	<code>0x60bb</code>
<code>adds r1, r7, #7</code>	<code>0x1df9</code>
<code>adds r3, r7, #6</code>	<code>0x1dbb</code>

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