Embedded Software Essentials

Make

C1 M2 V6

Copyright

Building Manually [S1a]

- Building can be tedious
  - Many GCC flags
  - Many independent commands
  - Many build targets
  - Many supported architectures
  - Many source files

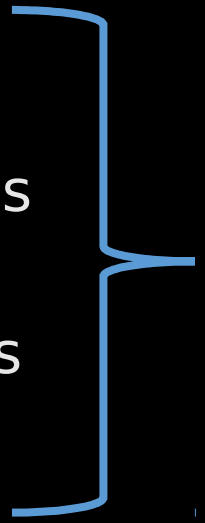
- Building manually can
  - Cause Consistency issues
  - Waste development time

Linux Kernel Example

- *.c Files: 23,000+
- *.h Files: 18,000+
- *.S Files: 1,400+

Building Manually [S1b]

- Building can be tedious
  - Many GCC flags
  - Many independent commands
  - Many build targets
  - Many supported architectures
  - Many source files



Manually compiling each file and linking is NOT scalable for large software projects or large teams

- Building manually can
  - Cause Consistency issues
  - Waste development time

Large chance for human error

Example Compile in KDS [S2a]

- Project for a KL25z Platform with a Cortex-M0+with containing 2 Source files and some startup Files

```
12:27:47 **** Build of configuration Debug for project project2 ****
make all
Building file: ../Sources/main.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Sources/main.d" -MT"Sources/main.o" -c -o
"Sources/main.o" "../Sources/main.c"
Finished building: ../Sources/main.c

Building file: ../Sources/memory.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Sources/memory.d" -MT"Sources/memory.o" -c -o "Sources/
memory.o" "../Sources/memory.c"
Finished building: ../Sources/memory.c

Building file: ../Project_Settings/Startup_Code/startup_MKL25Z4.S
Invoking: Cross ARM GNU Assembler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -x assembler-with-cpp -MMD -MP -MF"Project_Settings/Startup_Code/startup_MKL25Z4.d"
-MT"Project_Settings/Startup_Code/startup_MKL25Z4.o" -c -o "Project_Settings/Startup_Code/startup_MKL25Z4.o" "../Project_Settings/Startup_Code/startup_MKL25Z4.S"
Finished building: ../Project_Settings/Startup_Code/startup_MKL25Z4.S

Building file: ../Project_Settings/Startup_Code/system_MKL25Z4.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Project_Settings/Startup_Code/system_MKL25Z4.d" -
MT"Project_Settings/Startup_Code/system_MKL25Z4.o" -c -o "Project_Settings/Startup_Code/system_MKL25Z4.o" "../Project_Settings/Startup_Code/system_MKL25Z4.c"
Finished building: ../Project_Settings/Startup_Code/system_MKL25Z4.c

Building target: project2.elf
Invoking: Cross ARM C++ Linker
arm-none-eabi-g++ -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -T "MKL25Z128xxx4_flash.ld" -Xlinker --gc-sections -L"C:/classes/boulder/ECEN5013/Spring2016/kds_wksp/project2/Project_Settings/
Linker_Files" -Wl,-Map,"project2.map" -specs=nano.specs -specs=nosys.specs -o "project2.elf"  ./Sources/main.o ./Sources/memory.o ./Project_Settings/Startup_Code/startup_MKL25Z4.o ./Project_Settings/Startup_Code/system_MKL25Z4.o
Finished building target: project2.elf

12:27:49 Build Finished (took 1s.659ms)
```

Example Compile in KDS [S2b]

- Project for a KL25z Platform with a Cortex-M0+with containing 2 Source files and some startup Files

```
12:27:47 **** Build of configuration Debug for project project2 ****
make all
Building file: ../Sources/main.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Sources/main.d" -MT"Sources/main.o" -c -o
"Sources/main.o" "../Sources/main.c"
Finished building: ../Sources/main.c

Building file: ../Sources/memory.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Sources/memory.d" -MT"Sources/memory.o"  -c -o
"Sources/memory.o" "../Sources/memory.c"
Finished building: ../Sources/memory.c

Building file: ../Project_Settings/Startup_Code/startup_MKL25Z4.S
Invoking: Cross ARM GNU Assembler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -x assembler-with-cpp -MMD -MP -MF"Project_Settings/Startup_Code/startup_MKL25Z4.d"
-MT"Project_Settings/Startup_Code/startup_MKL25Z4.o" -c -o "Project_Settings/Startup_Code/startup_MKL25Z4.o" "../Project_Settings/Startup_Code/startup_MKL25Z4.S"
Finished building: ../Project_Settings/Startup_Code/startup_MKL25Z4.S

Building file: ../Project_Settings/Startup_Code/system_MKL25Z4.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -DTIMER_INTERRUPT=0 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Project_Settings/Startup_Code/system_MKL25Z4.d" -
MT"Project_Settings/Startup_Code/system_MKL25Z4.o" -c -o "Project_Settings/Startup_Code/system_MKL25Z4.o" "../Project_Settings/Startup_Code/system_MKL25Z4.c"
Finished building: ../Project_Settings/Startup_Code/system_MKL25Z4.c

Building target: project2.elf
Invoking: Cross ARM C++ Linker
arm-none-eabi-g++ -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections  -g3 -T "MKL25Z128xxx4_flash.ld" -Xlinker --gc-sections
-L"C:/classes/boulder/ECEN5013/Spring2016/kds_wksp/project2/Project_Settings/Linker_Files" -Wl,-Map,"project2.map" -specs=nano.specs -specs=nosys.specs -o "project2.elf"  ./Sources/main.o ./Sources/memory.o
./Project_Settings/Startup_Code/startup_MKL25Z4.o ./Project_Settings/Startup_Code/system_MKL25Z4.o
Finished building target: project2.elf

12:27:49 Build Finished (took 1s.659ms)
```

Example Compile in KDS [S3]

- Each compile, assemble and link command are
  - More then 100 Characters
  - More then 10 Flags

Need something to simplify this!

Building file: ../Sources/main.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -g3 -I"../Sources" -I"../Includes" -std=c99 -MMD -MP -MF"Sources/main.d" -MT"Sources/main.o" -c -o "Sources/main.o" "../Sources/main.c"

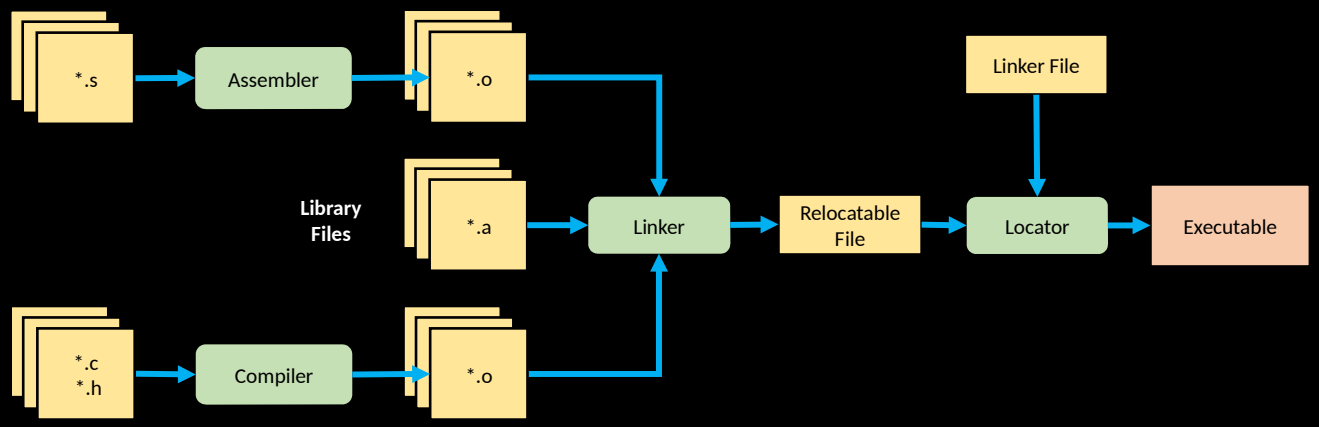
Building file: ../Project_Settings/Startup_Code/startup_MKL25Z4.S
Invoking: Cross ARM GNU Assembler
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -g3 -x assembler-with-cpp -MMD -MP -MF"Project_Settings/Startup_Code/startup_MKL25Z4.d" -MT"Project_Settings/Startup_Code/startup_MKL25Z4.o" -c -o "Project_Settings/Startup_Code/startup_MKL25Z4.o" "../Project_Settings/Startup_Code/startup_MKL25Z4.S"

Building target: project.elf
Invoking: Cross ARM C++ Linker
arm-none-eabi-g++ -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -g3 -T "MKL25Z128xxx4_flash.ld" -Xlinker --gc-sections -L"C:/coursera/kds_wksp/project2/Project_Settings/Linker_Files" -Wl,-Map,"project.map" -specs=nano.specs -specs=nosys.specs -o “project.elf" ./Sources/main.o ./Sources/memory.o ./Sources/ports.o ./Sources/timer.o ./Project_Settings/Startup_Code/startup_MKL25Z4.o ./Project_Settings/Startup_Code/system_MKL25Z4.o

Build Management Software [S4]

- Build Management Software (or Build Automation) provides a simple and consistent method for producing a target executable

- Automated the process of
  - Preprocessing
  - Assembling
  - Compiling
  - Linking
  - Relocating

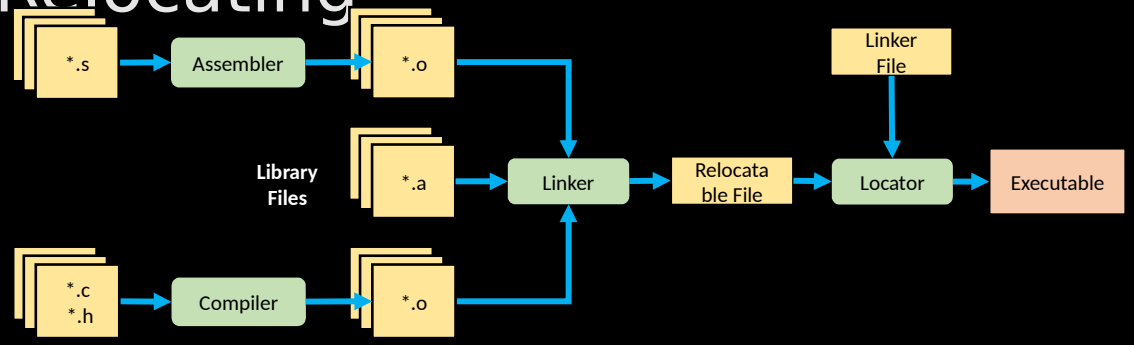


GNU Compiler Collection [S5]

- GNU Toolset performs all operations using **make**
  - Preprocessing
  - Assembling
  - Compiling
  - Linking
  - Relocating

```
alex@ubuntu14: ~
alex@ubuntu14:~$ which make
/usr/bin/make
alex@ubuntu14:~$ make -v
GNU Make 3.81
Copyright (C) 2006  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.

This program built for i686-pc-linux-gnu
alex@ubuntu14:~$
```



Name	Symbol	ARM Executable
Assembler	as	arm-none-eabi-as
Compiler	gcc	arm-none-eabi-gcc
Linker	ld	arm-none-eabi-ld
Make	make	make

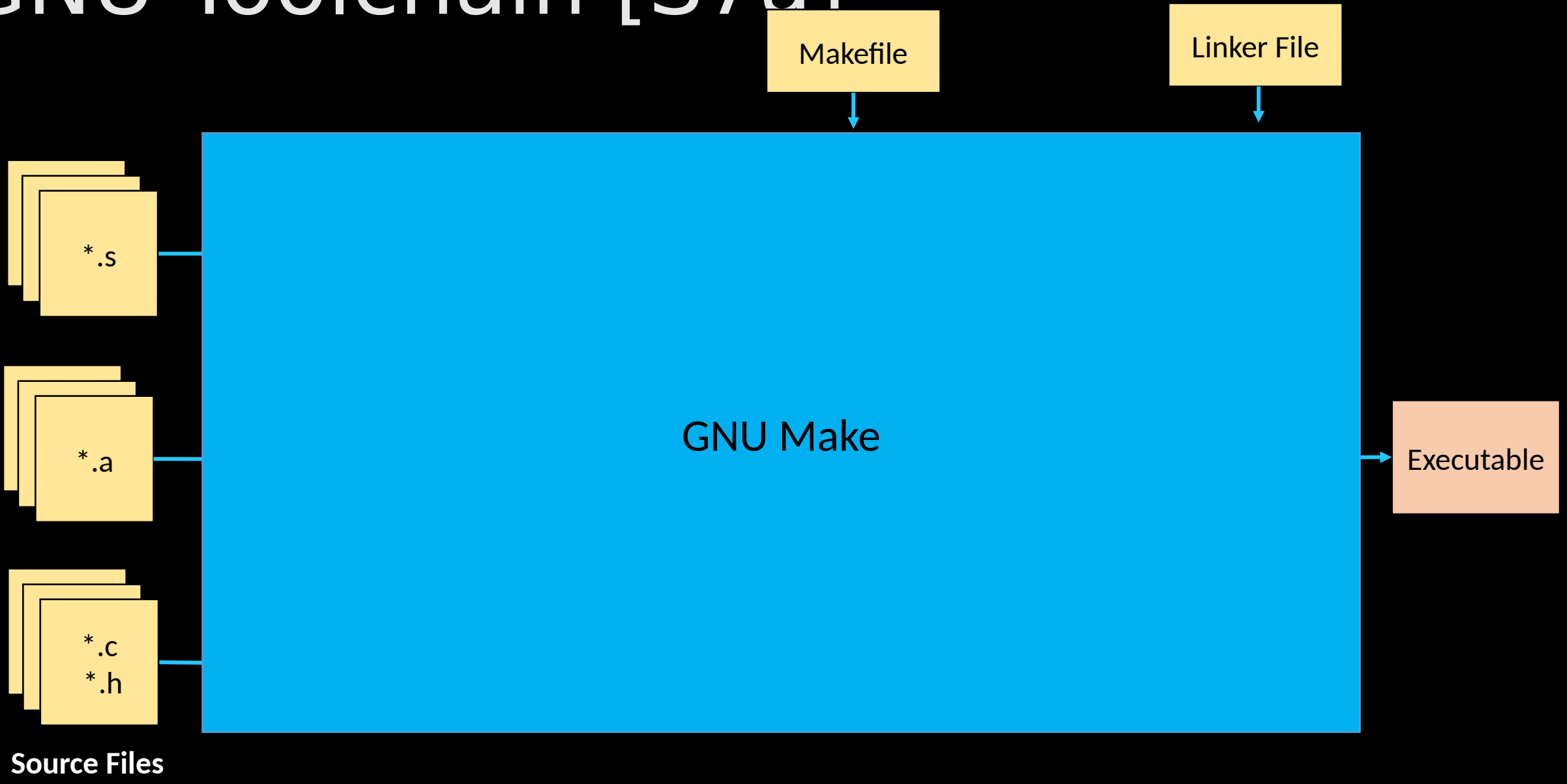
GNU Make [S6]

- GNU Make
  - “Tool that controls the generation of executables and other non-source files of a program from the program’s source files.”[2]

- GNU = GNU’s Not Unix
  - A collection of software, a **Toolset**
  - Contains GCC = GNUs Compiler Collection

- Make is our Build Management Software
  - Determines what files need to be compiled/recompiled in a project given a **makefile**
  - Helps generate build dependencies and other files
  - Widely used and free

GNU Toolchain [S7a1]



Makefiles [S8]

- One or more files used to tell **make** how to build a particular project
  - Invoked from the command line
- Makefiles have build **targets** or build **rules**
  - These are **recipes** for how to build a particular executable or non-source file
- Executables can have **dependencies**
  - Requirements needed for a particular recipe
  - These can be auto-generated from make
- Can define variables/constants to use during compilation
  - Compiler Instance
  - Compiler/Linker Options
  - Architecture to build for

Compile Time Switch

Makefile Rules/Targets [S9]

- Many multiple rules can be executed for any given instance of make that is run
  - Specify target you wish to execute when invoking make

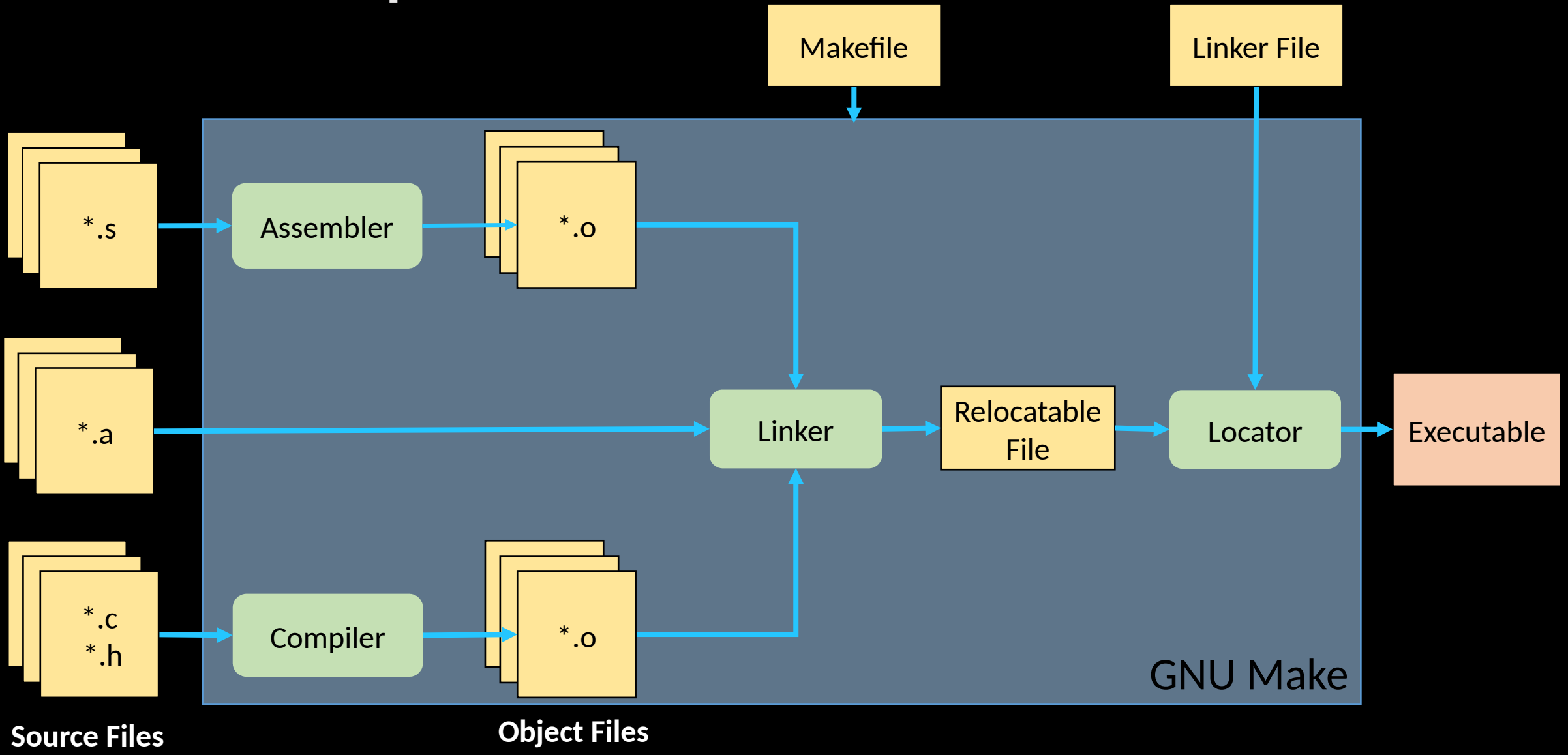
```
$ make main.o
$ make all
$ make clean
```

Example Makefile Targets

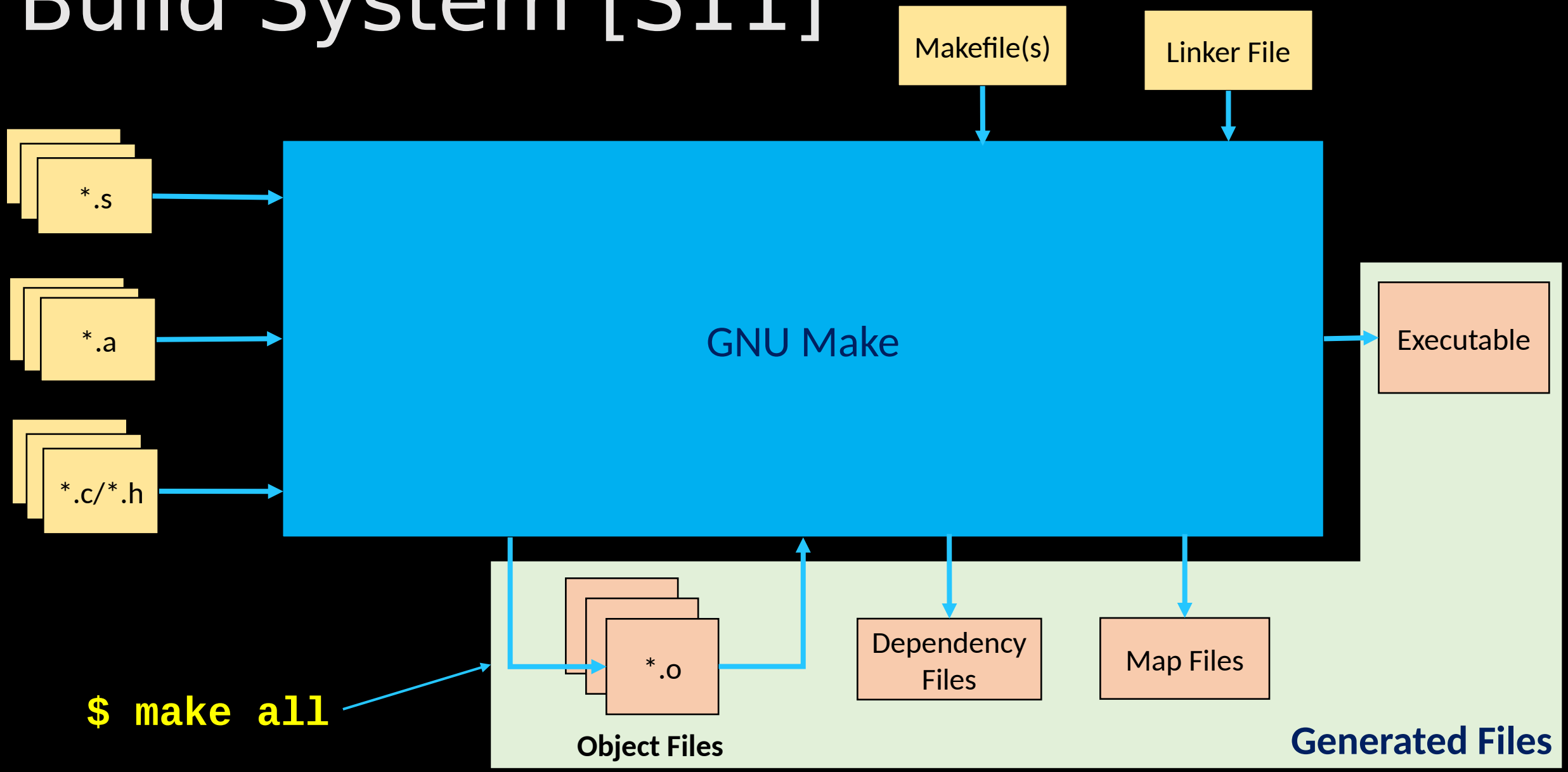
- If no target is specified, defaults to first defined target in the makefile

```
$ make
```

GNU Compiler Collection [S10]



Build System [S11]

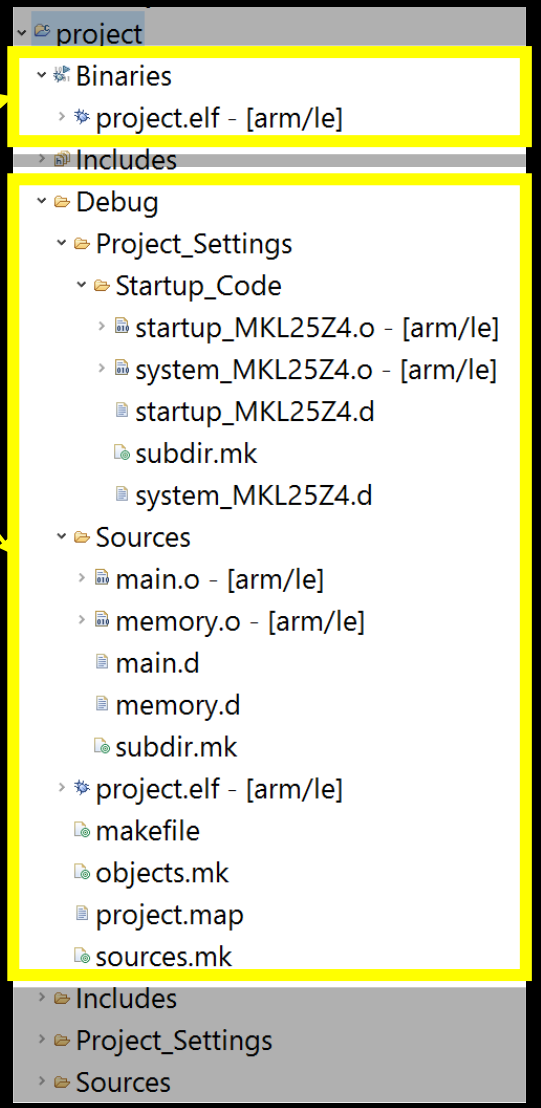


IDE and Make Autogeneration

- Your IDE typically will list Makefiles, output files, and executables in the project explorer

- IDE will dynamically create Makefiles through the use of file auto-generation
  - Software teams DO NOT use IDE autogenerated build systems, they create their own

Kinetis Design Studio Make Build System



Banner that gets inserted at the top of all of your auto-generated files

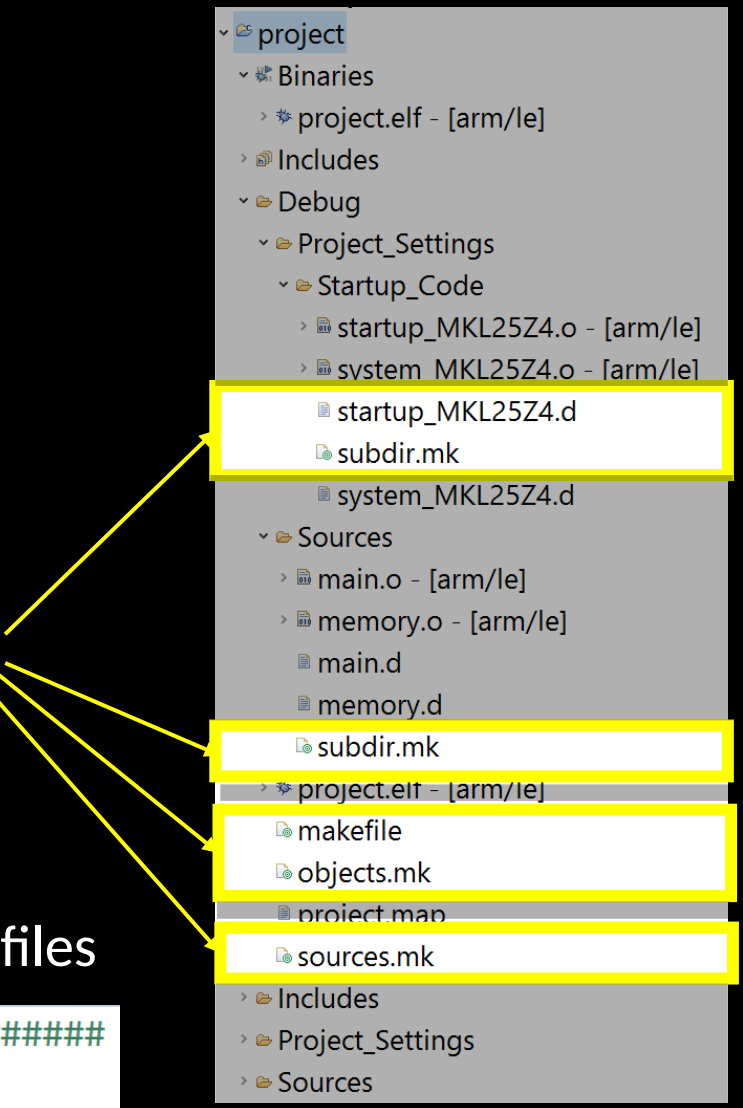
```
################################################################################
# Automatically-generated file. Do not edit!
################################################################################
```

IDE and Make Autogeneration

- Your IDE typically will list Makefiles, output files, and executables in the project explorer

- IDE will dynamically create Makefiles through the use of file auto-generation
  - Software teams DO NOT use IDE autogenerated build systems, they create their own

Kinetis Design Studio Make Build System



Banner that gets inserted at the top of all of your auto-generated files

```
################################################################################
# Automatically-generated file. Do not edit!
################################################################################
```