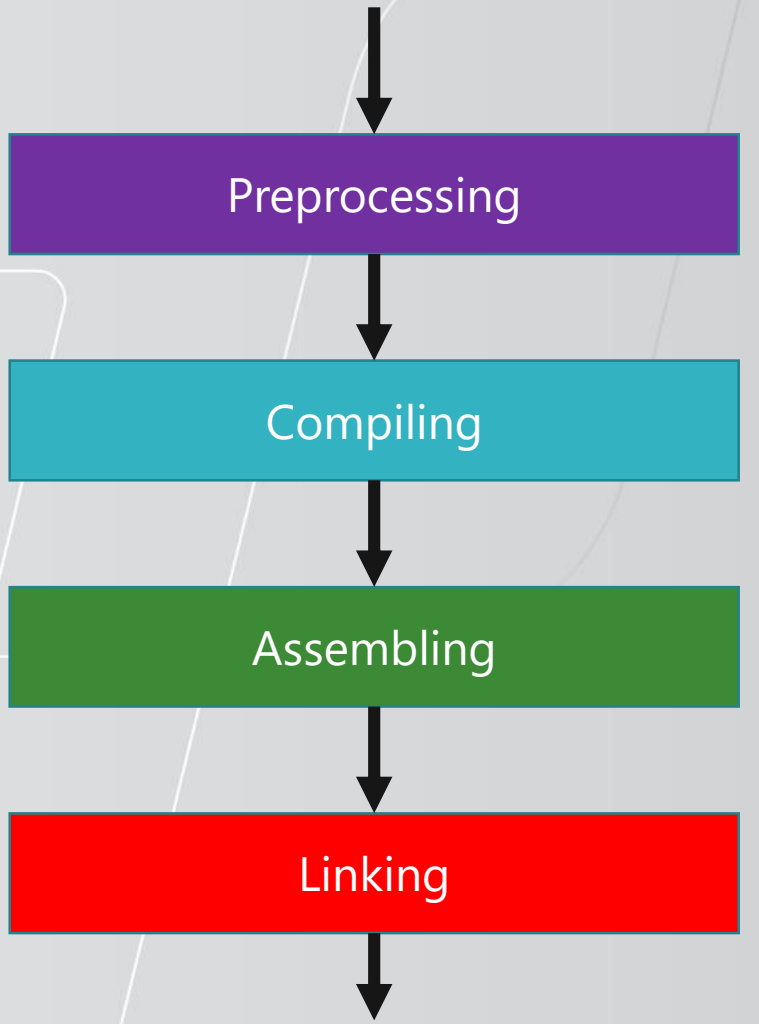
Compilation Process



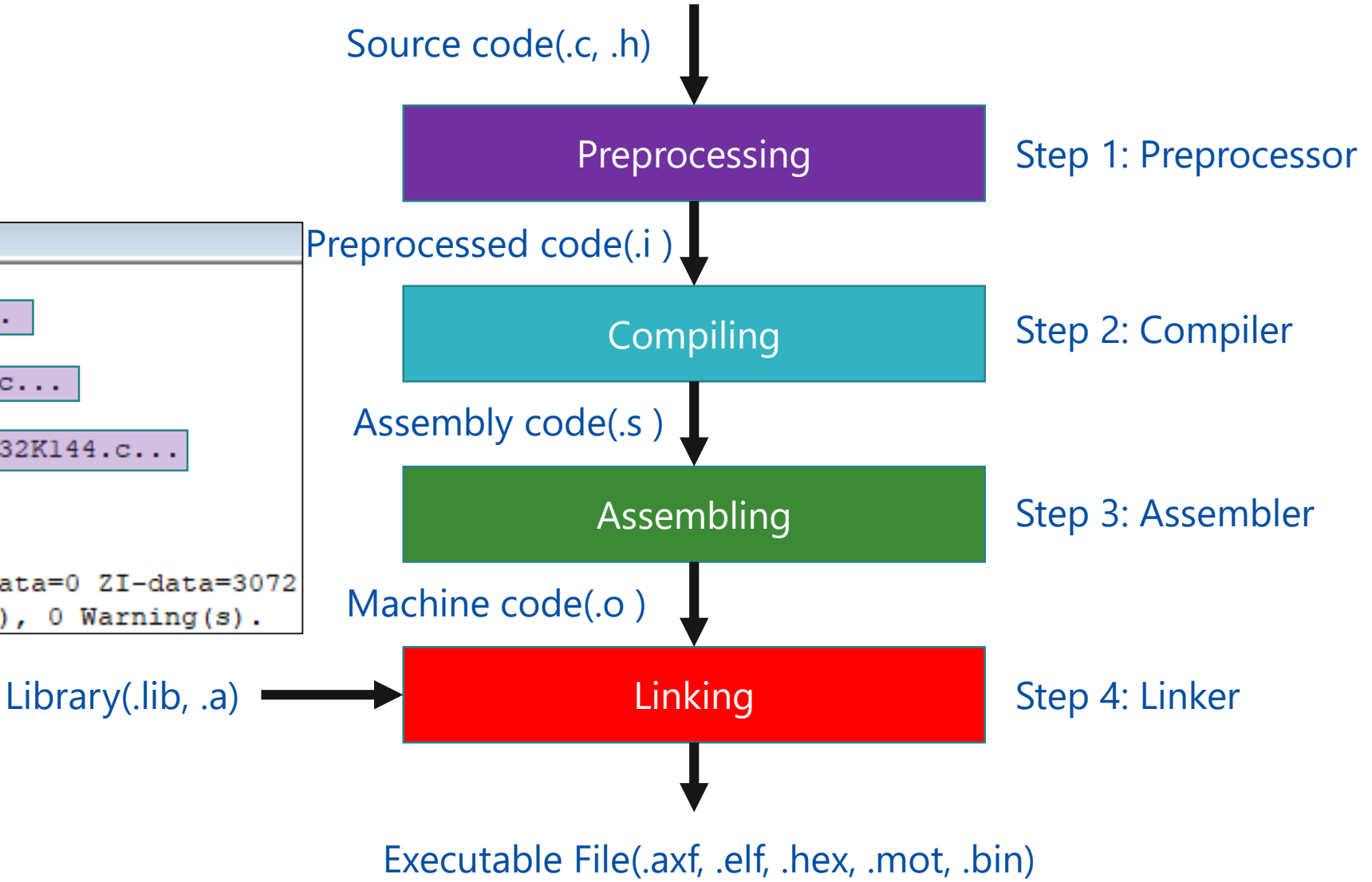
17 Oct 2023

OUTLINE

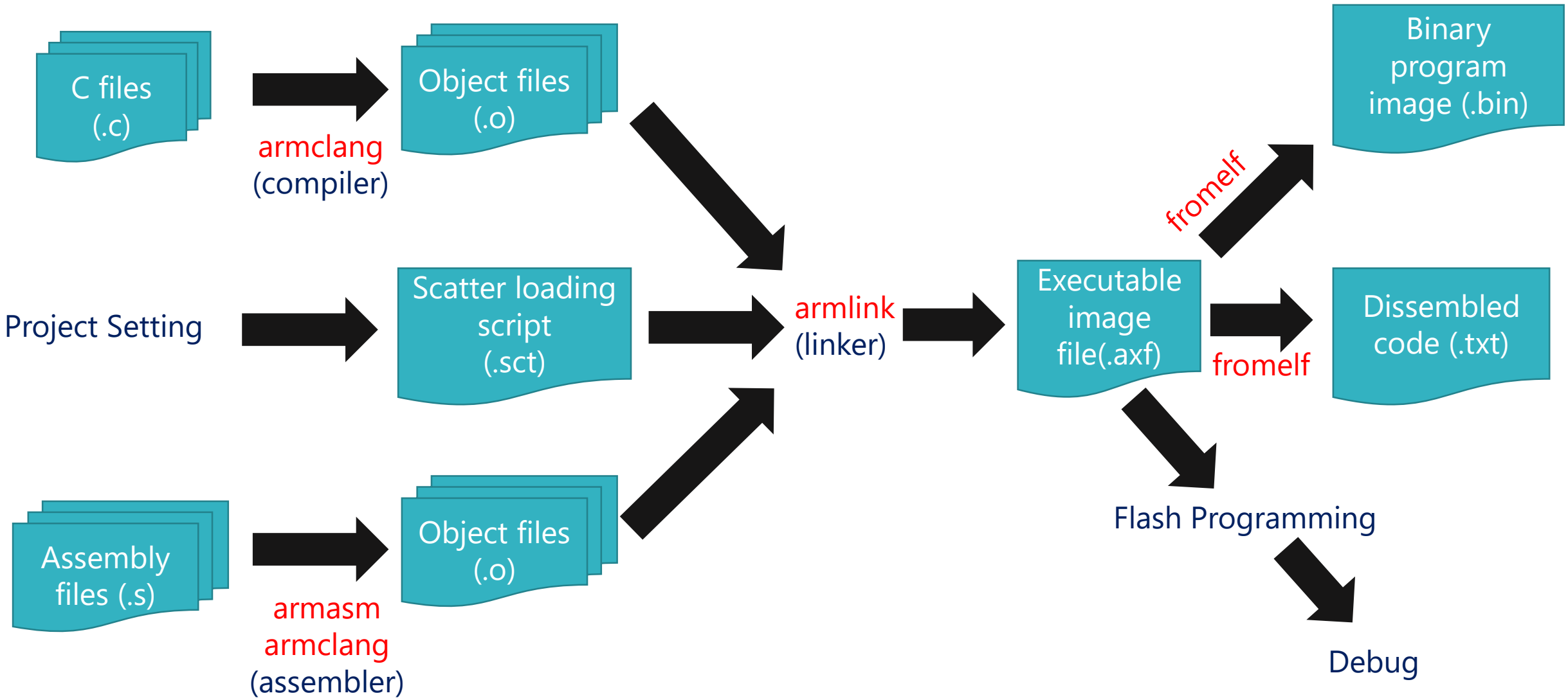
1. Compilation Process
2. Step 1: Preprocessor
3. Step 2: Compiler
4. Step 3: Assembler
5. Step 4: Linker
6. ARM image conversion utility

1. Compilation Process

```
Build Output
Rebuild target 'Target 1'
creating preprocessor file for main.c...
compiling main.c...
creating preprocessor file for startup.c...
compiling startup.c...
creating preprocessor file for system_S32K144.c...
compiling system_S32K144.c...
assembling startup_S32K144.S...
linking...
Program Size: Code=556 RO-data=580 RW-data=0 ZI-data=3072
".\Objects\Day08_Clock.axf" - 0 Error(s), 0 Warning(s).
```



1. Compilation Process



Step 1: Preprocessor

armclang
--target=arm-arm-none-eabi
-mcpu=cortex-m4
-E
-I<Path folder contain header file(option)>
<source file(.c)>
-o
<file output(.i)>

armclang --target=arm-arm-none-eabi -mcpu=cortex-m4 **-E** main.c -o main.i

Step 2: Compiler

armclang
--target=arm-arm-none-eabi
-mcpu=cortex-m4
-s
<Preprocessed code (.i)>
-o
<File output(.s)>

armclang --target=arm-arm-none-eabi -mcpu=cortex-m4 **-S** main.i -o main.s

Step 2: Compiler

armclang
--target=arm-arm-none-eabi
-mcpu=cortex-m4
-c
<Preprocessed code (.i) or source file(.c) >
-o
<file output(.o)>

armclang --target=arm-arm-none-eabi -mcpu=cortex-m4 **-c** main.i -o main.o

armclang --target=arm-arm-none-eabi -mcpu=cortex-m4 **-c** main.c -o main1.o

Step 3: Assembler

armclang
--target=arm-arm-none-eabi
-mcpu=cortex-m4
-masm=auto
-c
<Assembly code (.s)>
-o
<File output(.o)>

armclang --target=arm-arm-none-eabi -mcpu=cortex-m4 -masm=auto **-c** startup_S32K144.S -o st.o

Step 4: Linker

armlink
--scatter=<Scatter File.sct>
--cpu=cortex-m4
--info sizes
--symbols
--info totals
--info unused
……..
--map --list=<image.map>
<Object files (*.o)>
-o
<File output(.axf)>

armlink --scatter=S32K144.sct --cpu=cortex-m4 --info sizes --symbols --info totals --info unused --map --list=S32K144.map main.o startup.o -o out.axf

ARM image conversion utility

fromelf [Flags] input_file

Flags for Text Information

-v	verbose
-a	print data addresses (For images built with debug)
-c	disassemble code
-d	print contents of data section
-e	print exception tables
-g	print debug tables
-r	print relocation information
-s	print symbol table
-t	print string table
-y	print dynamic segment contents
-z	print code and data size information

fromelf –c main.o

fromelf –c main.o >dumobject.txt


Thank you