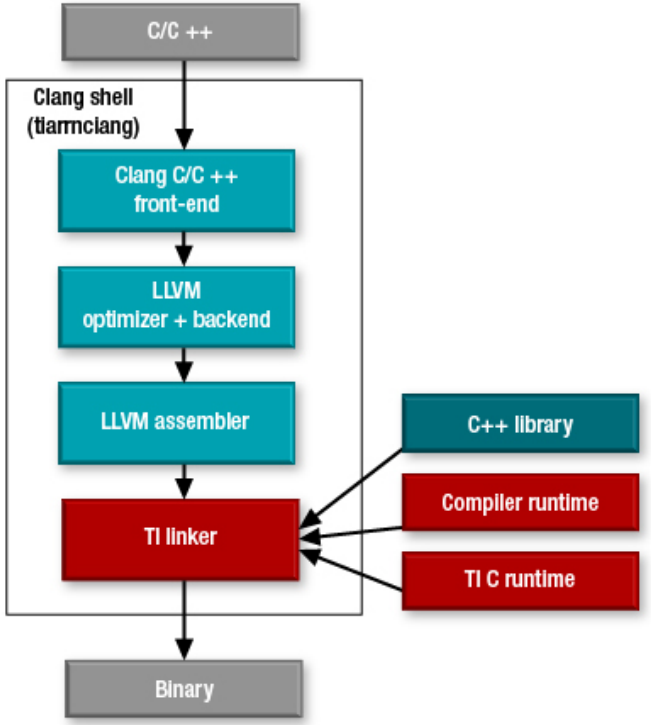
Introduction to Linker

Prashant Shivhare

Compilation: Source to Binary

- Compilation Process
  - Source File (.c,.cpp), Preprocessor, Compiler, Assembler, Linker
- Linker
  - Links multiple object files into one final executable file.
    - Object files: *.o, *.a, *.obj
    - Executable files: *.out, *.elf



TI ARM CLANG Compiler

- **tiarmclang**
  - C/C++ compiler & integrated assembler.
  - Invoke linker by default
  - Options
    - *-E* : Stop after pre-processing.
    - *-S* : Stop after producing assembly file.
    - *-c* : Stop after producing object file. Do not invoke linker.
    - *-Wl,<opt-list>* : Options passed to linker.

- **tiarmlnk**
  - Actual TI ARM Linker
  - Usually invoked via *tiarmclang* only so that object library search path & runtime libraries are implicitly included.

TI ARM Linker (tiarmlnk)

- **Entry point**
  - The address of the first instruction to be run.
  - Default entry point: *_c_int00*
  - *-e* <*symbol*>: Tells linker to set <*symbol*> as the entry point.

- **Map file**
  - Include information helpful for debugging.
    - Layout of final executable file
    - Module summary, symbols, entry point, etc.
    - *-m=<file name>*: Produces the information in map file named *<file name>.*

Linker Command File

- **Linker options**
  - *-e<symbol>*
  - *--stack_size*
  - *--heap_size*
- **Symbols**
  - Can define symbols that can be used in source files.
  - Resolved at link time
- **Directives**
  - *SECTIONS* : Control the input & output sections
  - *MEMORY* : Defines the target memory

```
1
2   --stack_size=16384
3   --heap_size=32768
4
5
6   SECTIONS
7   {
8       .vectors:{} palign(8) > M4F_VECS
9       .text:   {} palign(8) > M4F_IRAM
10      .bss:    {} palign(8) > M4F_DRAM
11      .data:   {} palign(8) > M4F_DRAM
12      .rodata: {} palign(8) > M4F_DRAM
13      .sysmem: {} palign(8) > M4F_IRAM
14      .stack:  {} palign(8) > M4F_IRAM
15  }
16
17  MEMORY
18  {
19      M4F_VECS : ORIGIN = 0x00000000 , LENGTH = 0x00000200
20      M4F_IRAM : ORIGIN = 0x00000200 , LENGTH = 0x0002FE00
21      M4F_DRAM : ORIGIN = 0x00030000 , LENGTH = 0x00010000
22  }
23
```

Sections

Section Name	Use Case
.text	Used for program code.
.bss	Used for uninitialized global variables.
.data	Used for initialized non-const global variables.
.const	Used for initialized const objects.
.rodata	Used for string constants.
.stack	Used for the function call stack.
.sysmem	Used for the dynamic memory allocation pool.

Executable File

- Permanent storage
  - Non-volatile storage
    - Flash, eMMC, etc.
  - Store anywhere in this type of storage
- Load
  - Stage before running.
  - Load the executable at the load addresses mostly in temporary storages.
  - May load in slow & large memories.
- Run
  - After loading, start running following the run addresses.
  - May need to copy code in case load & run address differ.
  - Run from fast & small memories

Sample Program

```
tiarmclang -mcpu=cortex-m4 -o hello_world.out hello_world.c "-Wl,-emain,-m=hello_world.map"
```

- Run the above command to produce the *.out* file.
- Options
  - *-mcpu* : Target processor
  - *-o* : Name of the output produced
  - *-Wl* : Pass options to linker
    - *-emain* : Set entry point to *main*
    - *-m* : Produce map file

THANK YOU ☺