Arm/AArch64 Embedded Development with LLD

What's New?

Amilendra Kodithuwakku (Arm Limited)

EuroLLVM 2023

arm

A lightning talk on recent additions to LLD for embedded development on Arm.

1. Cortex®-M Security Extensions (CMSE)
2. Big Endian support

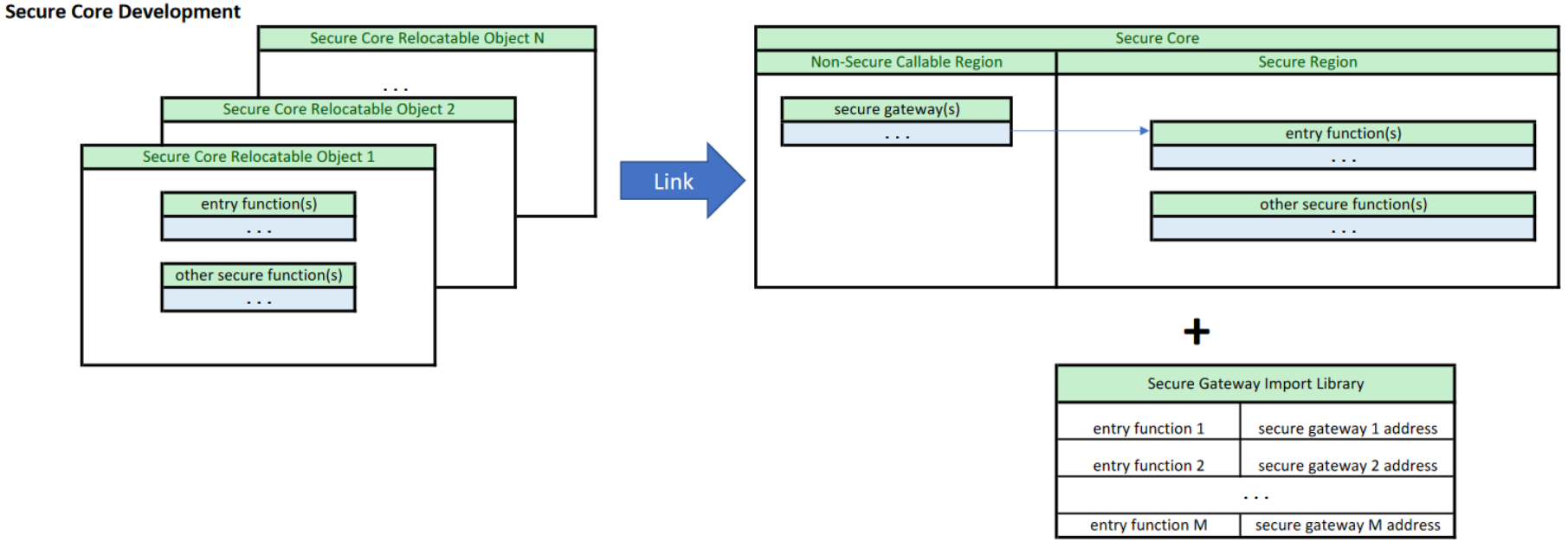
Cortex®-M Security Extensions (CMSE)

- A.K.A Armv8-M Security Extensions or Arm TrustZone for Armv8-M
- Under CMSE, physical memory is divided into Secure and Non-Secure regions.
- A Secure region has a part called a Non-Secure Callable (NSC) region that can be accessed by Non-Secure memory.

Cortex®-M Security Extensions (CMSE)

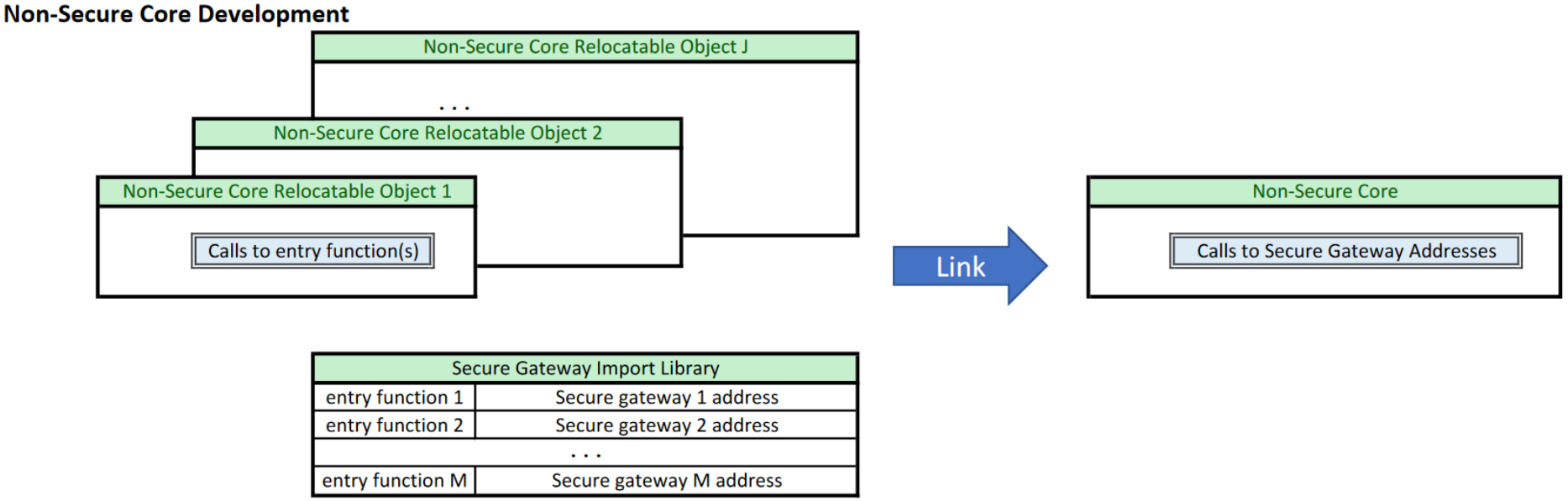
- A CMSE application generally has two parts;
  - a secure core
  - a non-secure core
- The secure core resides in a Secure memory region
- The non-secure code resides in a Non-secure memory region
- The secure and non-secure cores are developed independent of each other

CMSE application development flow



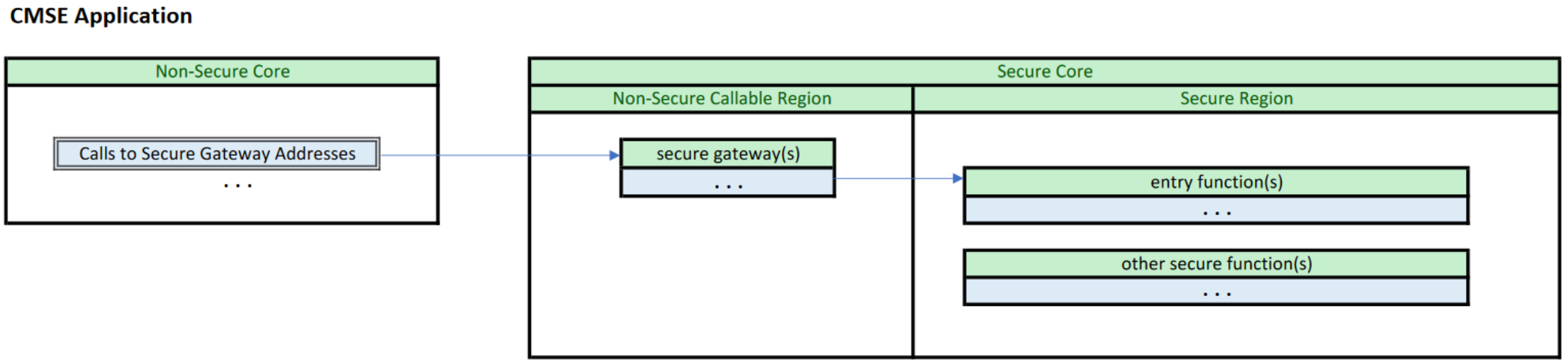
- Functions in the Secure core meant to be accessed by the Non-secure core are called Entry Functions.
- The linker creates a Secure Gateway (SG) for each Entry Function.
- Entry functions reside in the Secure memory region
- Secure Gateways reside in the Non-Secure Callable region.
- The linker also outputs a **Secure Gateway Import Library (import library),** which is a relocatable file that maps Entry Functions to their corresponding SG addresses.

CMSE application development flow



- Secure core developers share the import library with non-secure core developers.
- Using information in the import library, the linker relocates all calls to the entry functions in the non-secure core to go through their corresponding secure gateways.

CMSE application development flow



Big Endian Support for Arm

- Endianness is a simple concept, but as a colleague at Arm likes to put it, “it is the most complicated simple thing in the world”

- There are 3 independent things that can be big or little endian:
  - ELF file metadata (Symbols, Relocations etc)
  - Instructions
  - Data

- AArch64 has only 1 big endian configuration.
  - ELF file metadata (Symbols, Relocations etc) is big-endian
  - Instructions are little-endian
  - Data is little-endian

- Arm has more complicated big-endian configurations.

The BE-32 (word-invariant) configuration

- Early Arm CPUs (pre Arm-V6) supports a big-endian configuration called BE-32
  - ELF file metadata (Symbols, Relocations etc) is big-endian
  - Instructions are big-endian
  - Data is big-endian
- Works only for word-aligned accesses

The BE-8 (byte-invariant) configuration

- Arm-V6 introduced unaligned access which made BE-32 insufficient.
- BE-8 was introduced as a result.
  - ELF file metadata (Symbols, Relocations etc) is big-endian
  - Instructions are big-endian in relocatable objects **but little-endian in executables**
  - Data is big-endian
- Note : BE-8 is same as BE-32 except for instructions in executables being little-endian.
- To avoid an ABI break, the linker changes the endianness of instructions when creating BE8 executables.

BE-32 support in LLD

- LLD was developed concentrating more on Linux development, so the Arm backend implementation assumed little-endian.
- For embedded development this assumption is not true.
- Adding BE-32 support meant removing assumptions of little-endian.
- Although BE-32 is needed only for compatibility with legacy systems, the decision to support BE-32 was made because that made adding BE-8 support easier

BE8 support in LLD

- At a high level the only difference between BE-32 and BE-8 is that for BE-8
  - The linker should support the `--be8` option which instructs the linker to generate BE-8 executables
  - The linker sets the flag `EF_ARM_BE8` in the ELF header of BE-8 executables
  - The linker reverses the endianness of instructions (but not data)
- This is still WIP.

References

- CMSE
  - Armv8-M Architecture Reference Manual
    - https://developer.arm.com/documentation/ddi0553/latest/
  - Requirements for the Toolchain
    - https://developer.arm.com/documentation/ecm0359818/latest
  - LLD Support (WIP)
    - https://reviews.llvm.org/D139092

- Big-Endian
  - Word/Byte invariance
    - https://exchangetuts.com/types-of-endianness-1639978363148945
  - `EF_ARM_BE8` in the ELF header of BE-8 executables
    - https://github.com/ARM-software/abi-aa/blob/main/aaelf32/aaelf32.rst#52elf-header
  - LLD BE32 Support (Merged)
    - https://reviews.llvm.org/D140201
    - https://reviews.llvm.org/D140202

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה