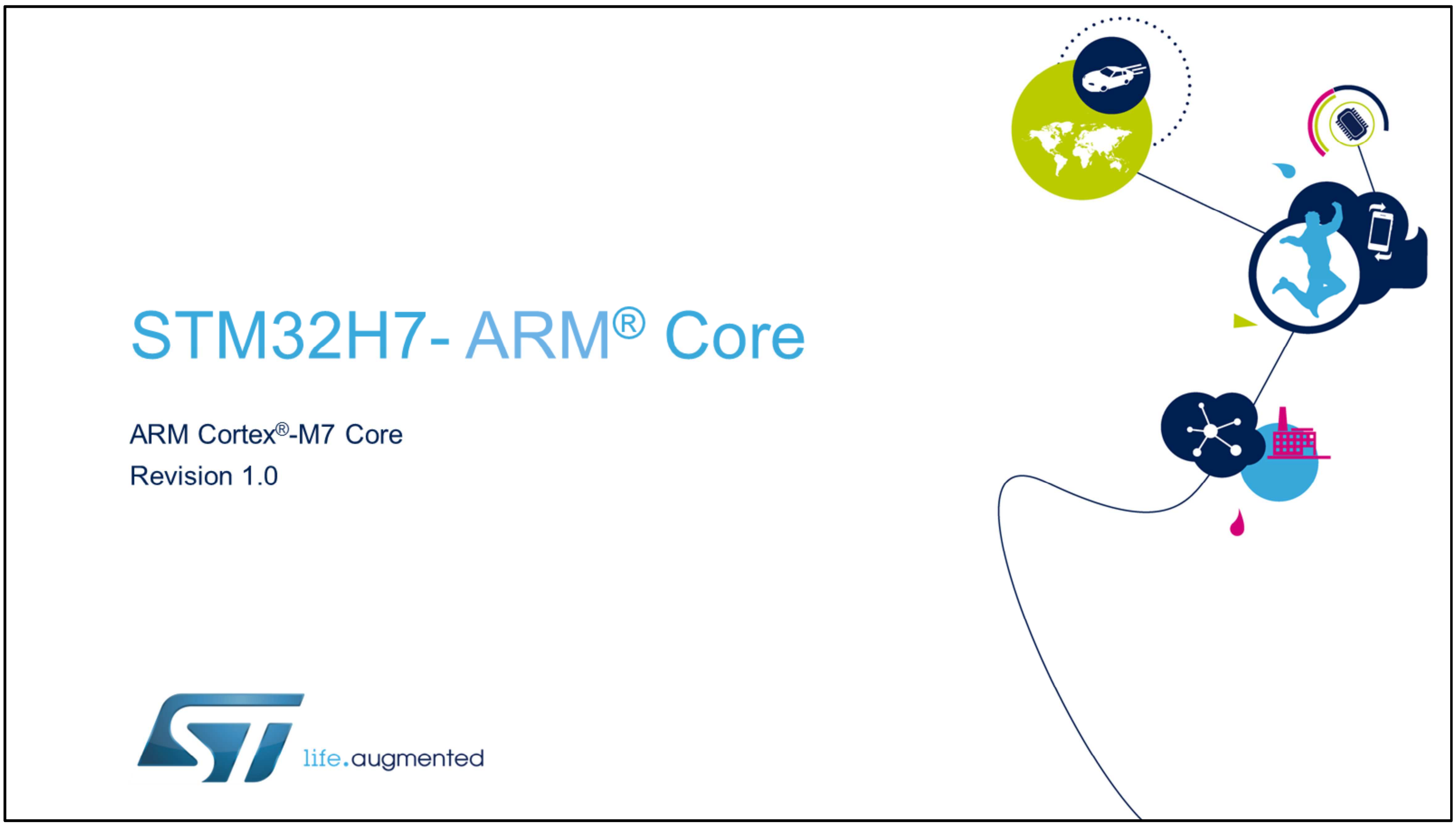


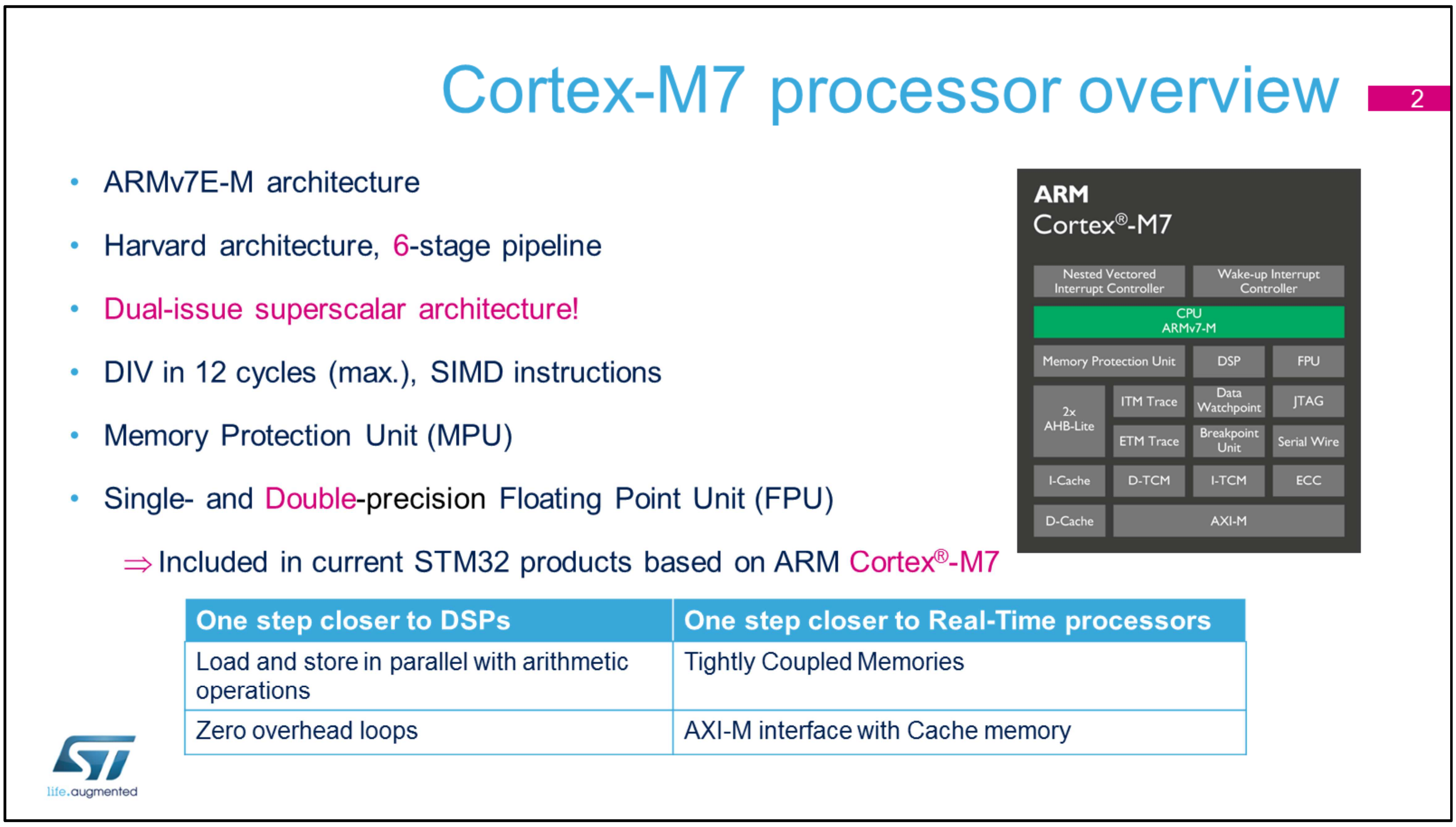
STM32H7- ARM® Core

ARM Cortex®-M7 Core

Revision 1.0

life.augmented

Hello, and welcome to this presentation of the ARM® Cortex®-M7 core which is embedded in all products of the STM32H7 microcontroller family.

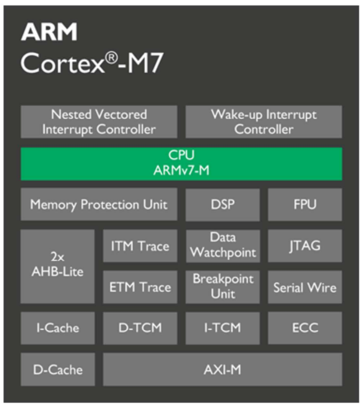


Cortex-M7 processor overview

2

- ARMv7E-M architecture
- Harvard architecture, 6-stage pipeline
- Dual-issue superscalar architecture!
- DIV in 12 cycles (max.), SIMD instructions
- Memory Protection Unit (MPU)
- Single- and Double-precision Floating Point Unit (FPU)

⇒ Included in current STM32 products based on ARM Cortex®-M7



One step closer to DSPs	One step closer to Real-Time processors
Load and store in parallel with arithmetic operations	Tightly Coupled Memories
Zero overhead loops	AXI-M interface with Cache memory

life.augmented

The Cortex®-M7 core is part of the ARM Cortex®-M group of 32-bit RISC cores. It implements the ARMv7E-M architecture.

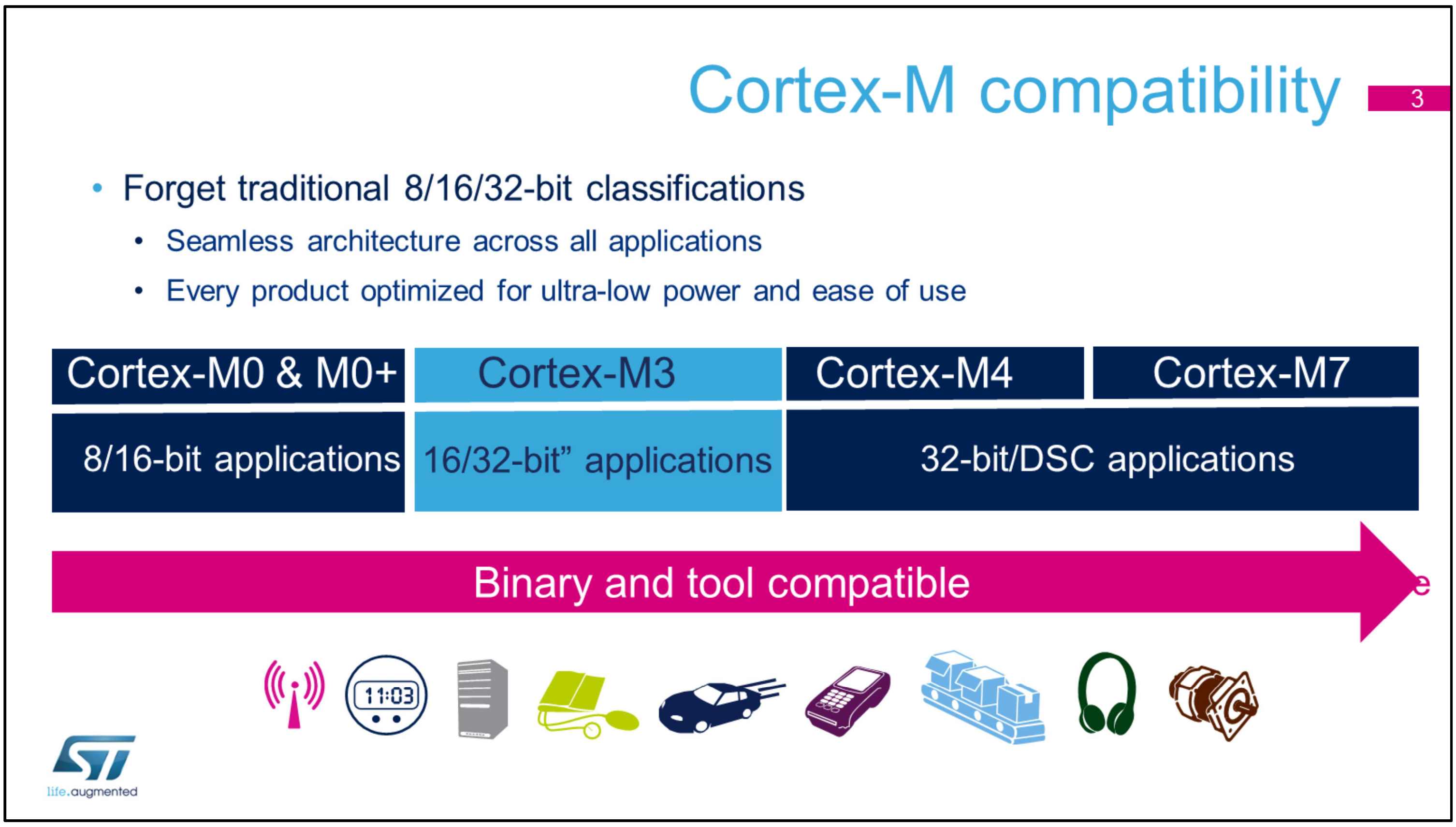
It features a 6-stage pipeline and in-order dual-issue superscalar with single- and double-precision floating point unit and SIMD support.

The performance of the Cortex®-M7 core is much closer to that of a digital signal processor than the Cortex®-M4 core.

It can execute load and store operations in parallel with arithmetic operations with zero overhead on loops.

The Cortex®-M7 core directly interfaces with Tightly

Coupled Memories or TCM for very low interrupt latency, resulting in a more deterministic execution.



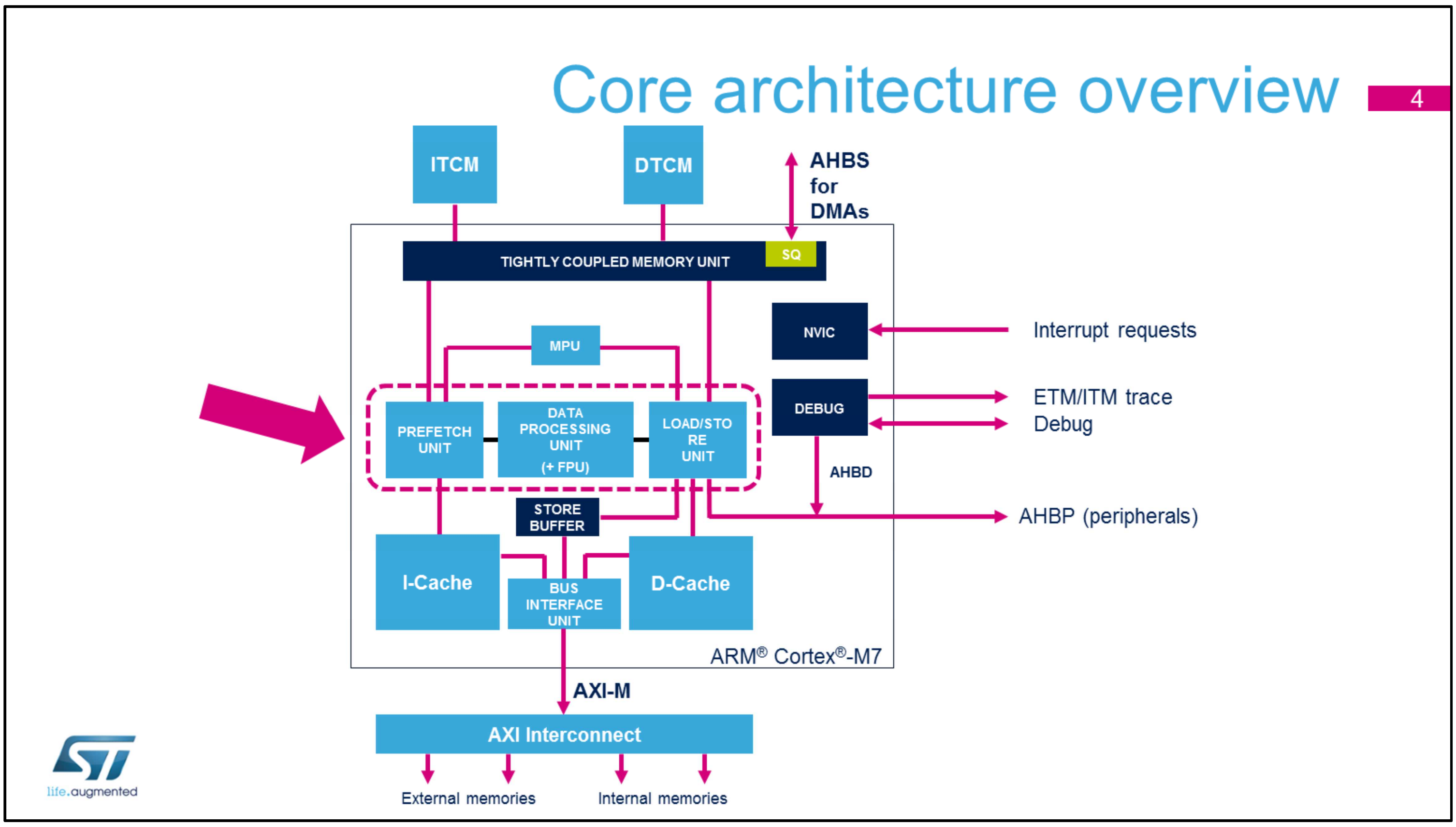
Cortex-M compatibility

- Forget traditional 8/16/32-bit classifications
  - Seamless architecture across all applications
  - Every product optimized for ultra-low power and ease of use

Cortex-M0 & M0+	Cortex-M3	Cortex-M4	Cortex-M7
8/16-bit applications	16/32-bit" applications	32-bit/DSC applications	

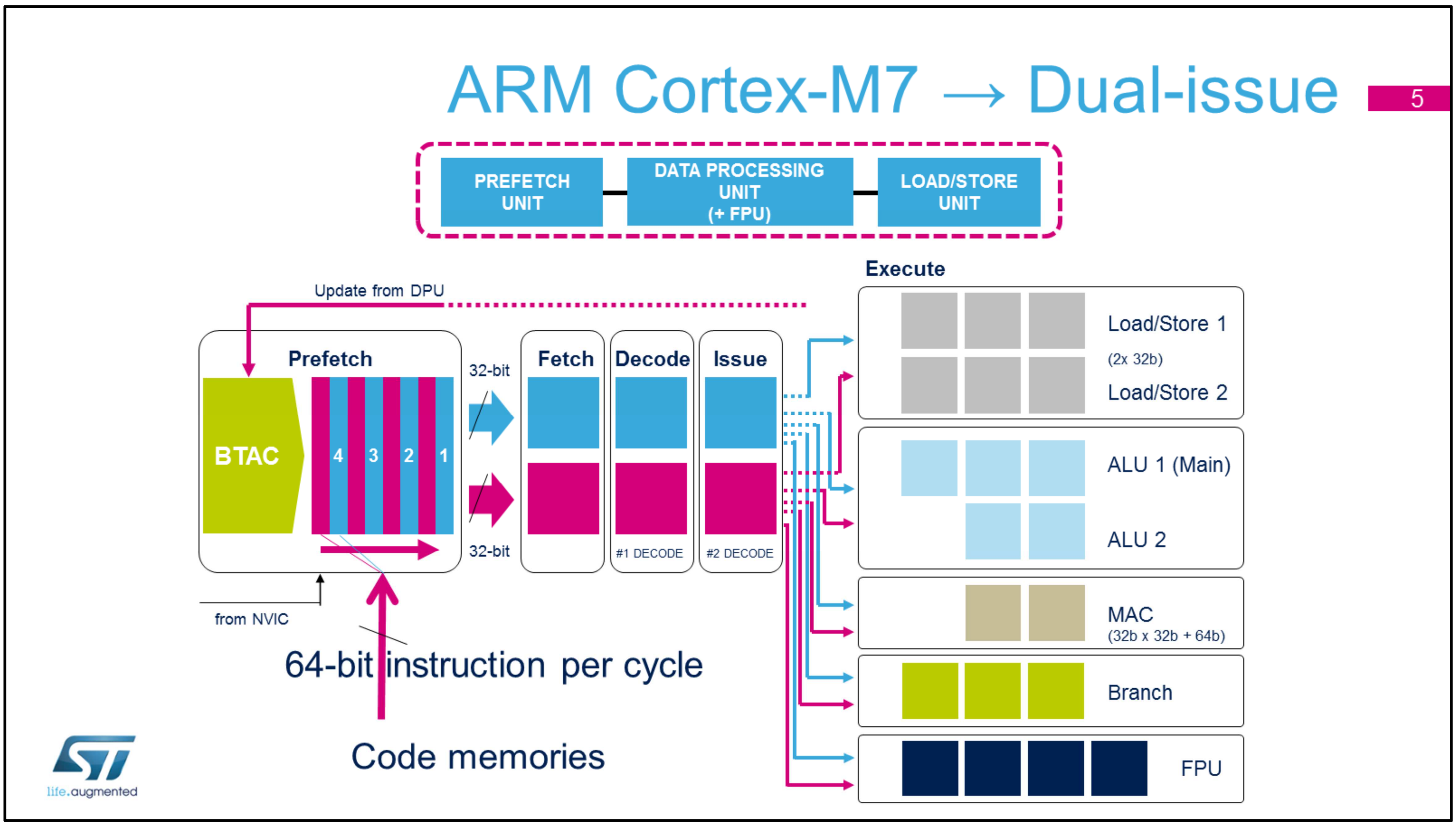
Binary and tool compatible
11:03
life.augmented

STM32H7 microcontrollers integrate an ARM® Cortex®-M7 core in order to benefit from the improved performance of the Cortex®-M processors architecture and particularly of the high level of performance in low-power modes.



Core architecture overview

The Cortex®-M7 core delivers more performance than Cortex®-M4 core thanks to an enhanced architecture bringing increased processing capabilities.
To enable this performance improvement, let’s look at the three units which are responsible for executing instructions starting by the prefetch unit (PFU), and then the data processing unit (DPU) in conjunction with the load store unit (LSU).



ARM Cortex-M7 → Dual-issue

The prefetch unit (PFU) provides one 64-bit instruction per cycle to the **D**ata **P**rocessing **U**nit (DPU).
It includes:

- a buffer of 4 entries of 64 bits each to enable fetching ahead of the DPU
- and **B**ranch **T**arget **A**ddress **C**ache (BTAC) for single cycle branch prediction

The Cortex®-M7 core has a 6-stage dual-issue pipeline for efficient operation. It brings the ability to process 2 instructions in parallel if certain criteria are fulfilled.

The data processing unit (DPU) is split into several pipes:

- Two ALUs, with one ALU capable of executing SIMD operations,
- Single MAC pipeline with one MAC per cycle capability,

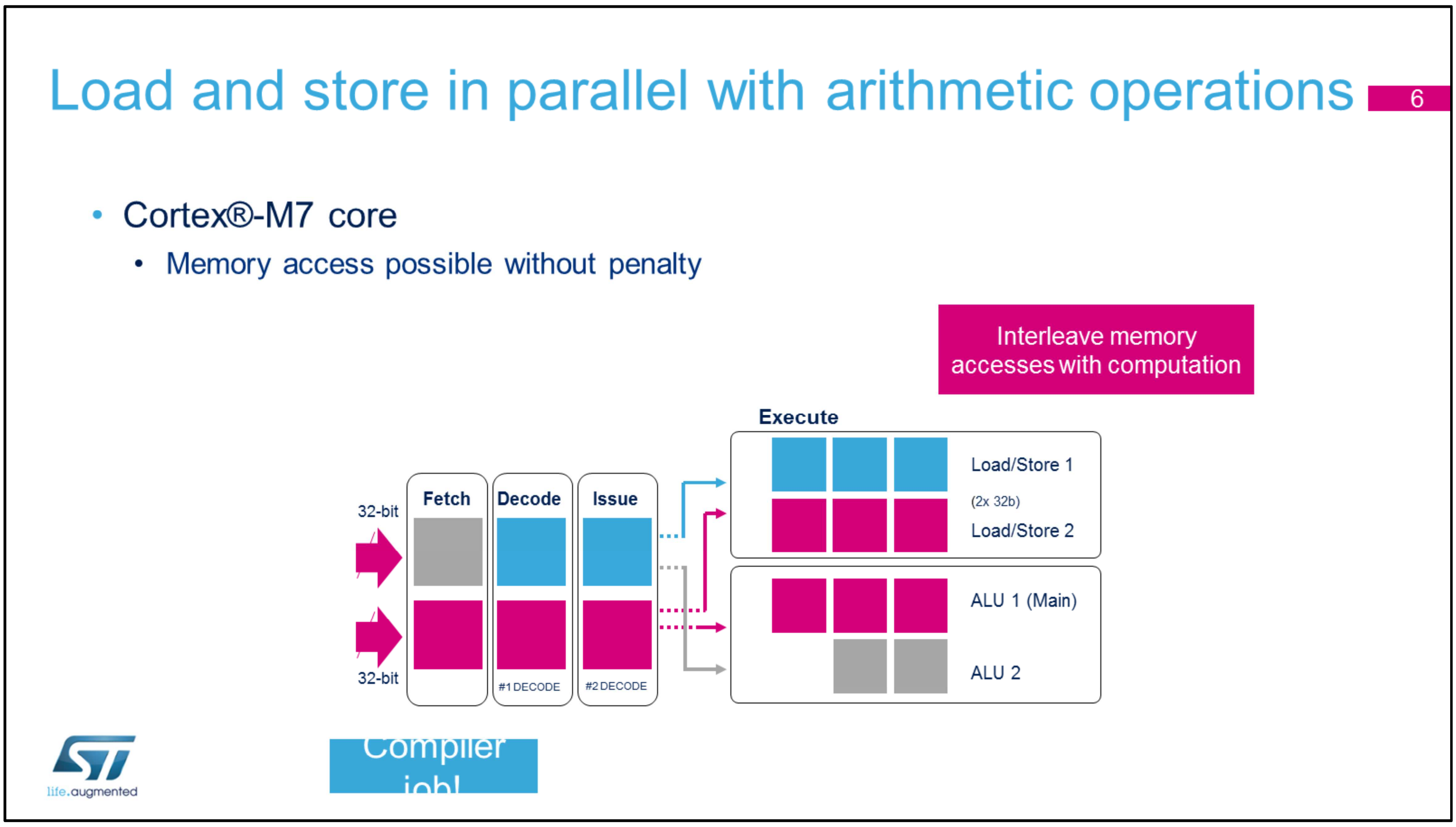
- One floating point pipe supporting single and double precision operation.

When an instruction reaches the Issue stage, it is split into micro-operations and based on the needed operation and registers used. It is then issued to the appropriate blocks further in the processing pipe.

Forwarding of flags from the DPU to the PFU allows early resolution of direct branches in the decoder and first execution stages of pipeline.

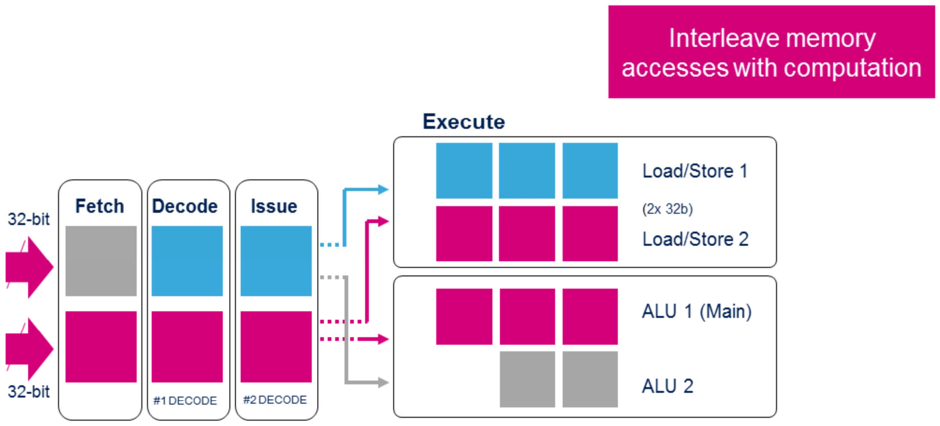
The Load Store Unit (LSU) provides either dual 32-bit load channels or a single 64-bit store channel with store buffering to increase store throughput.

The compiler hides the complexity of the core pipeline and optimizes code to take advantage of this architecture.



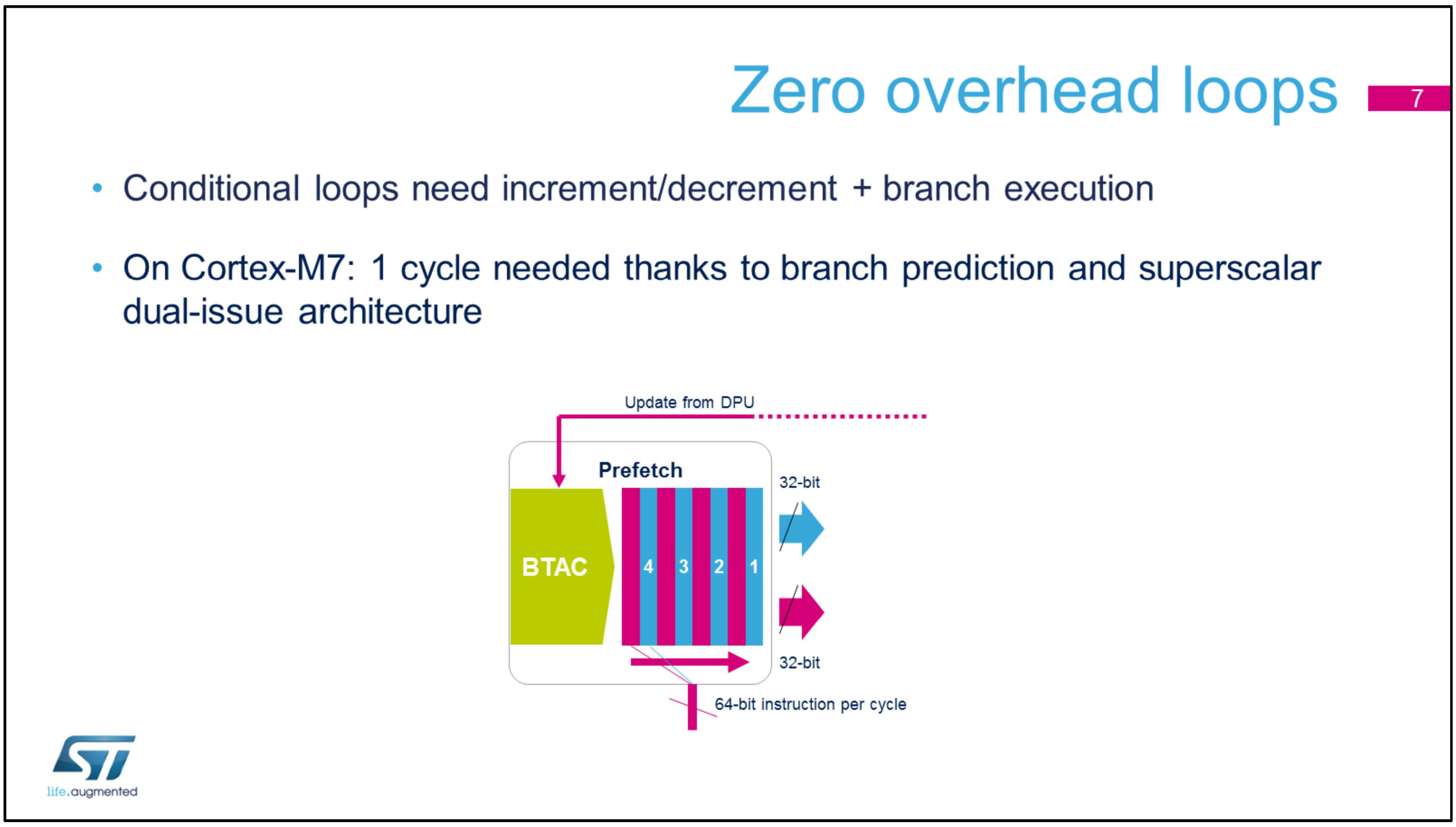
Load and store in parallel with arithmetic operations

- Cortex®-M7 core
  - Memory access possible without penalty



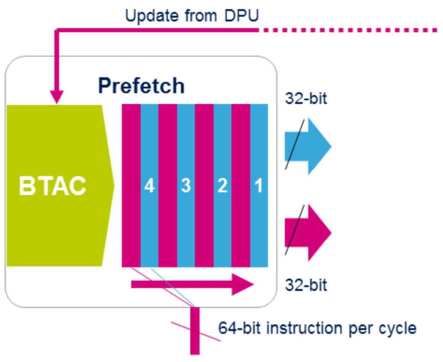
Compiler job!

Compared to the Cortex®-M4 core, the most important advantage is that code can read dual 32-bit values (double load with one instruction) and in parallel process the previous two data on the MAC pipe.
The Cortex®-M7 core is more efficient with long sequences of computations.



Zero overhead loops

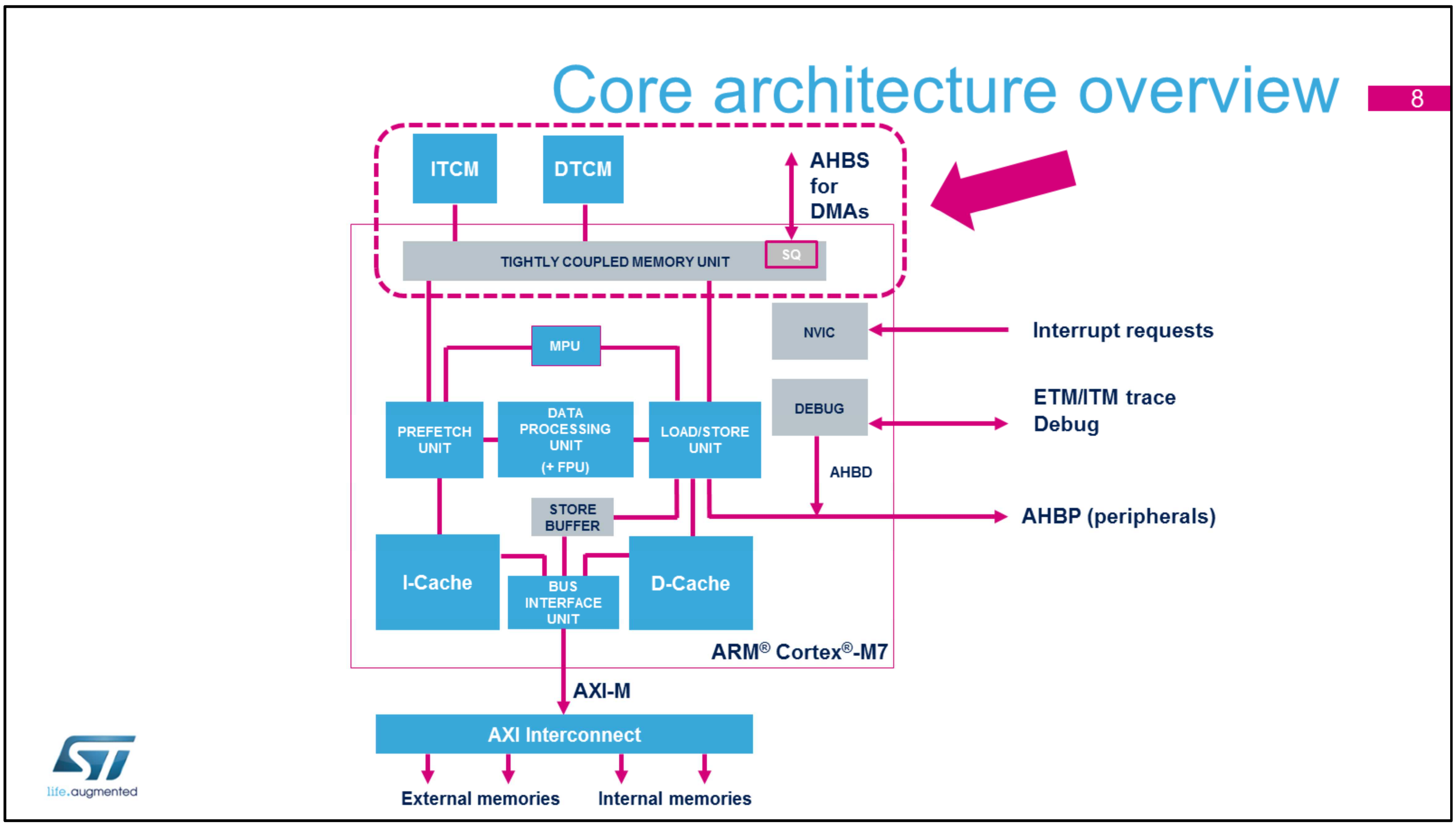
- Conditional loops need increment/decrement + branch execution
- On Cortex-M7: 1 cycle needed thanks to branch prediction and superscalar dual-issue architecture



life.augmented

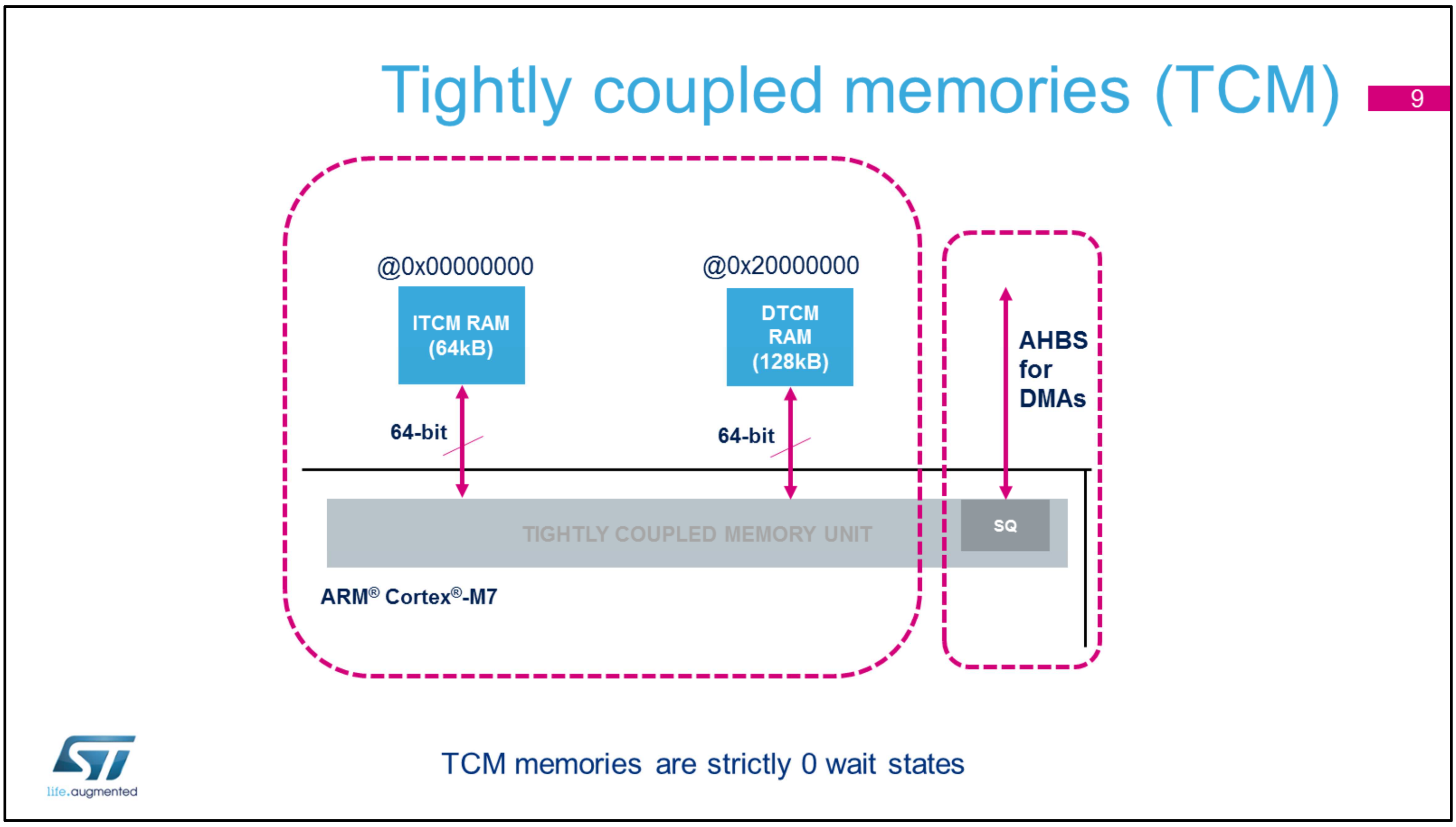
As a branch can also be dual issued, it can be executed in parallel with computation.

Branch Target Address Cache or BTAC predicts whether the branch can be taken or not and reacts accordingly. It remembers the conditions and, based on the processing, it predicts the next address to fetch.



Core architecture overview

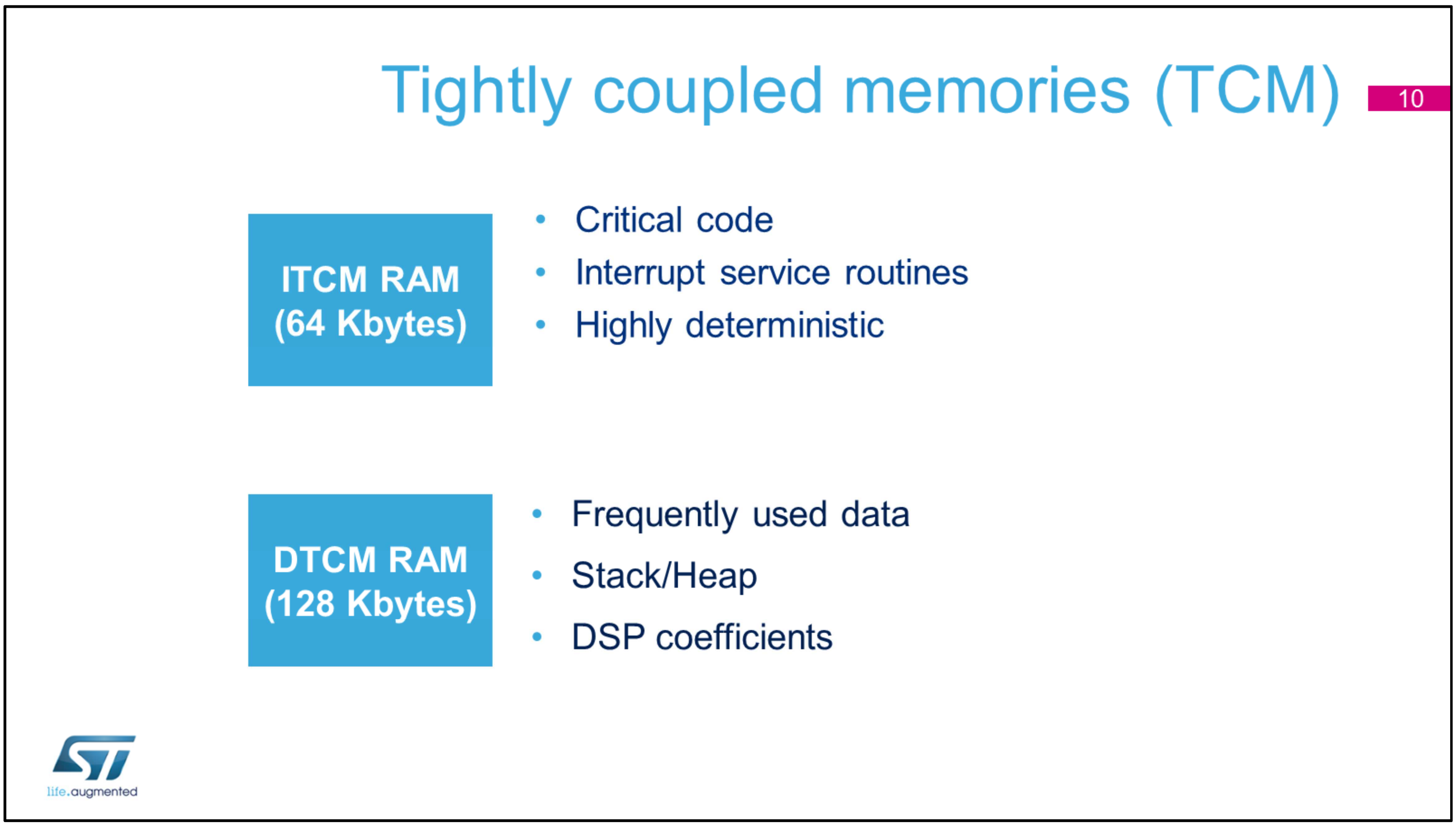
Tightly-Coupled Memories (TCM) are dedicated memories directly connected to the processor, but not through a bus, so avoiding arbitration and latencies for frequently executed code.
Tightly Coupled Memories for instructions (ITCM) and data (DTCM) allow to statically map important data and instructions to be accessed over these interfaces. This can be the case for the vector table, the interrupt service routines and certain time-critical control loops that are executed often and require low latency and deterministic execution time.
The AHBS provides a mean for DMAs to access any of the tightly coupled RAMs.
STM32H7 devices also implement ECC protection on TCM memories.



Tightly coupled memories (TCM)

TCM memories are strictly 0 wait states

The ITCM RAM has one 64-bit memory interface to satisfy core fetch bandwidth.
STM32H7 microcontrollers enable access to 64-Kbyte SRAM over the ITCM interface.
The DTCM RAM has two 32-bit memory interfaces to ensure more parallelism on request. Software can use up to 128 Kbytes of SRAM for critical data.
ITCM enables 12 clock cycles interrupt latency, which is achieved when code is placed in ITCM and data in DTCM.
The AHBS is a 32-bit AMBA3 AHB-Lite Slave interface. It provides system access (as for example DMAs) to the ITCM and DTCM. The AHBS supports simultaneous system and processor access requests.



Tightly coupled memories (TCM)

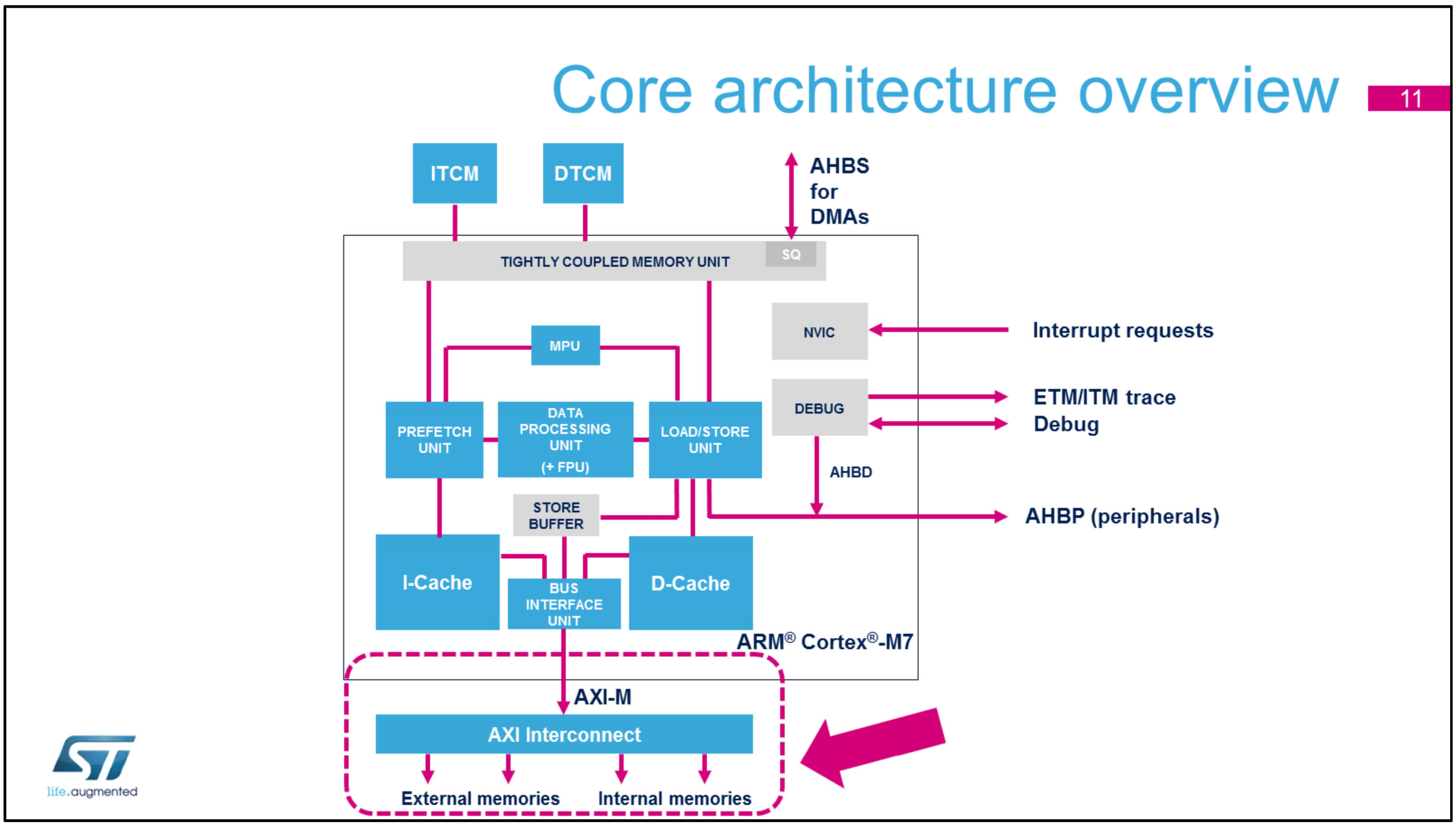
ITCM RAM (64 Kbytes)

- Critical code
- Interrupt service routines
- Highly deterministic

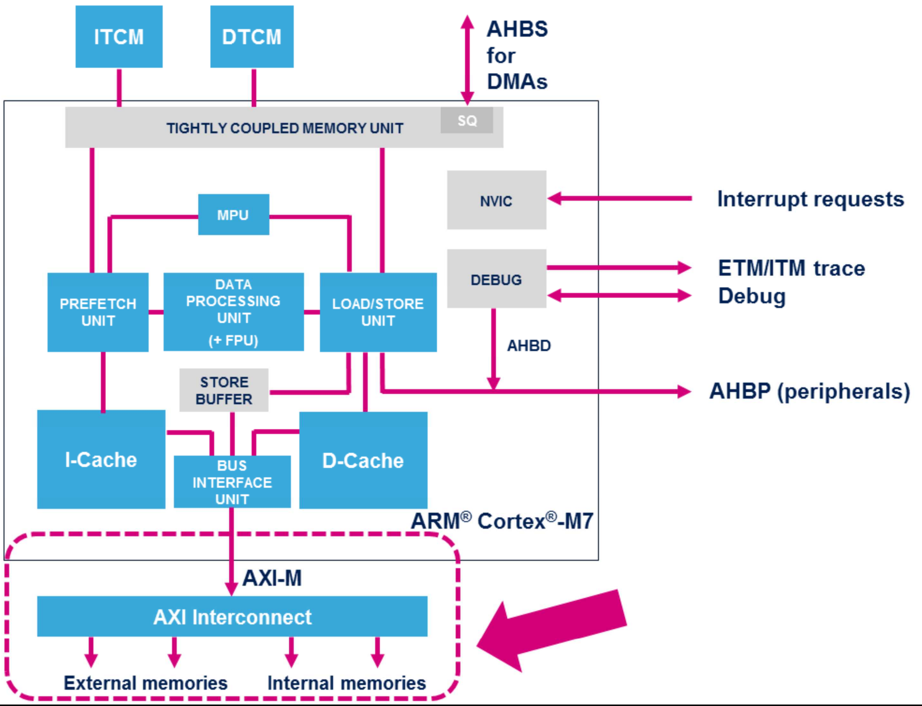
DTCM RAM (128 Kbytes)

- Frequently used data
- Stack/Heap
- DSP coefficients

Here are some situations where you might want to use the ITCM RAM and DTCM RAM.



Core architecture overview

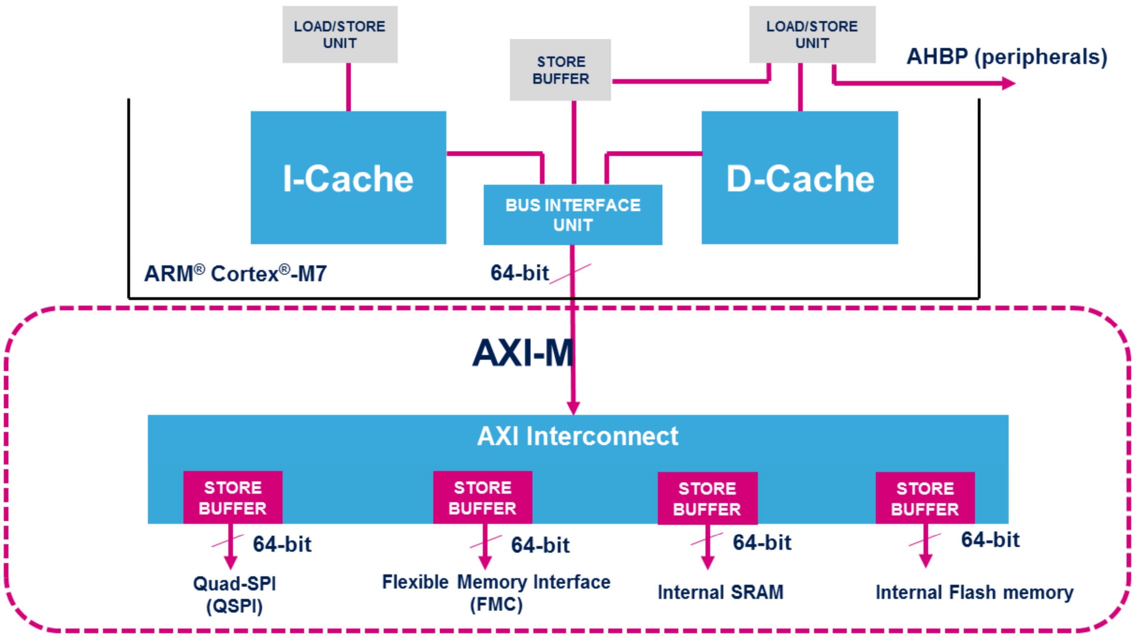


Compared to the Cortex®-M4 core where several AHB buses are needed for parallel transactions with memory system, the Cortex®-M7 core integrates a single AXI master bus.

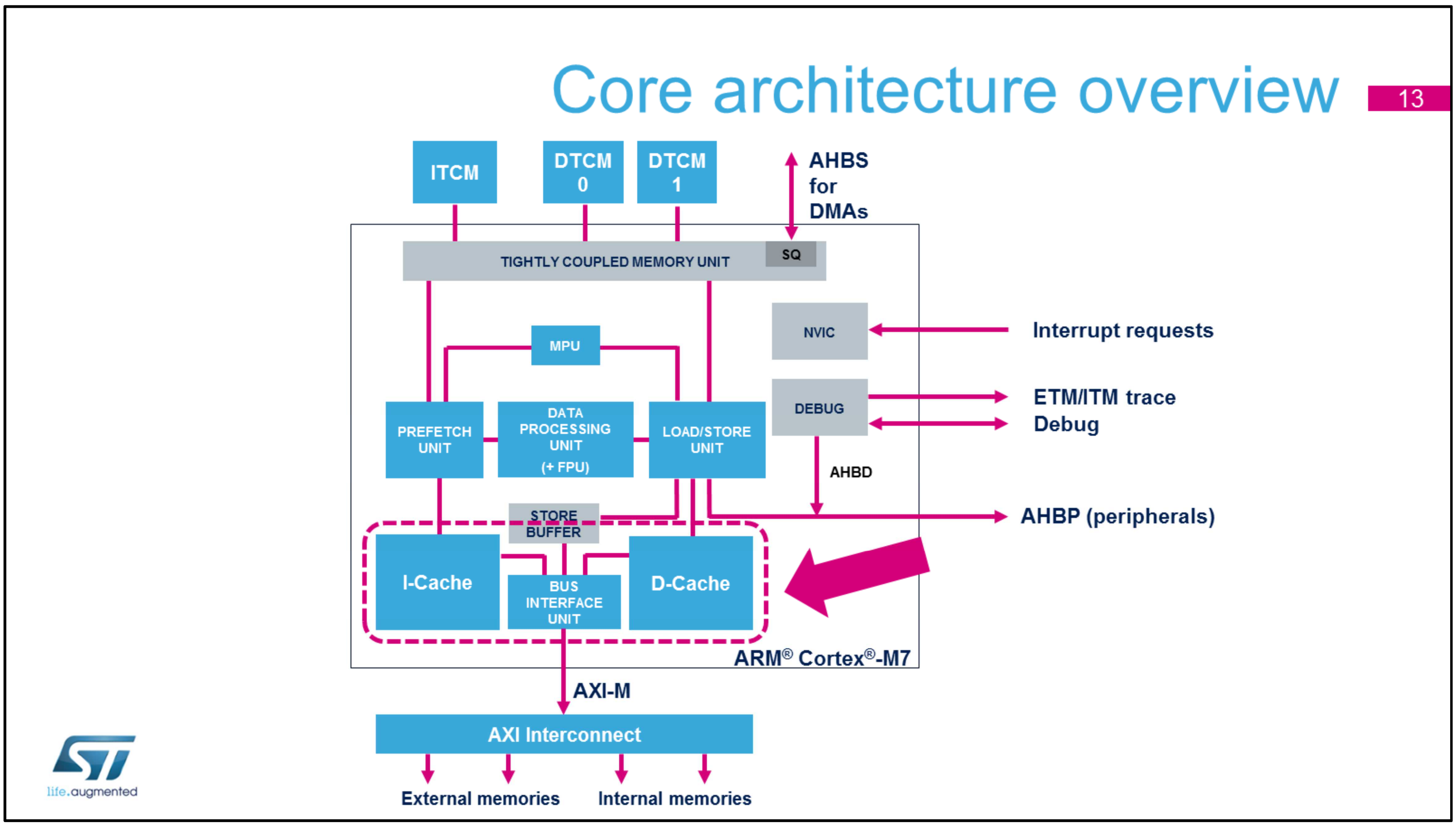
The AXI Master (AXI-M) interface is part of the Bus Interface Unit or BIU. It is a 64-bit wide AXI interface that connects the CPU to internal and external memories. It can be used for:

- Instruction fetches
- Data cache line fills and evictions
- Non-cacheable, normal-type memory data accesses
- Device and strongly-ordered type data accesses

AXI-M interface



STM32H7 microcontrollers integrate an AXI bus matrix to take advantage of the Cortex®-M7 AXIM interface.
The AXI de-correlates the access request from the data phase, so the request is independent from its corresponding data. If the memory has latencies, this separation makes the bus available to perform a new request if no functional relationship exists between the two requests (example Instruction Fetch & Data Fetch are performed in parallel).

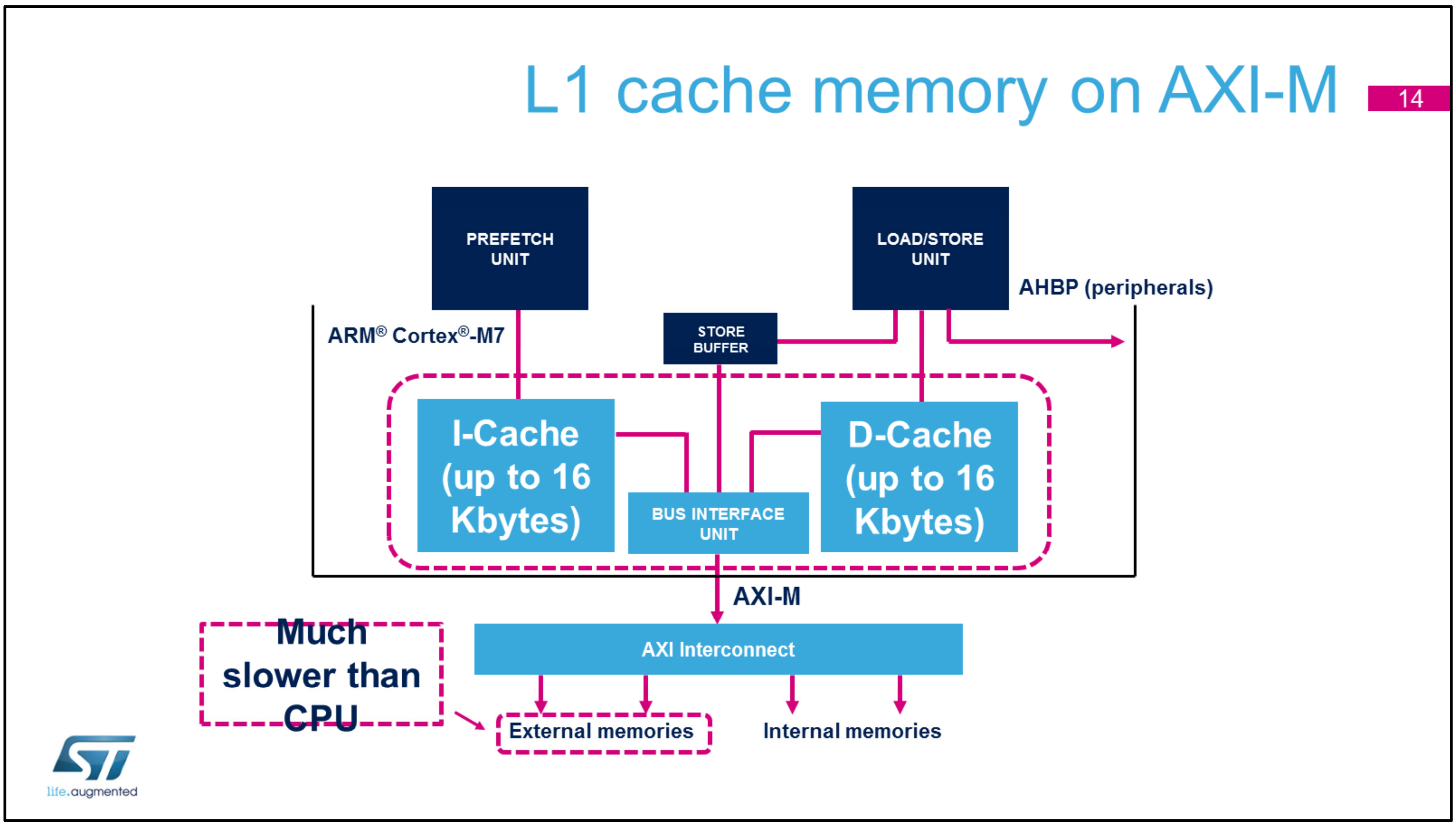


Core architecture overview

In addition to the AXIM interface, the Cortex®-M7 core integrates optional Instruction and Data caches for efficient memory access.

If the cache is enabled, any access that is not for a TCM or the AHBP interface is managed by the appropriate cache controller. In case of a cache hit, data is fetched or written to cache RAMs if the cache-ability criteria is fulfilled.

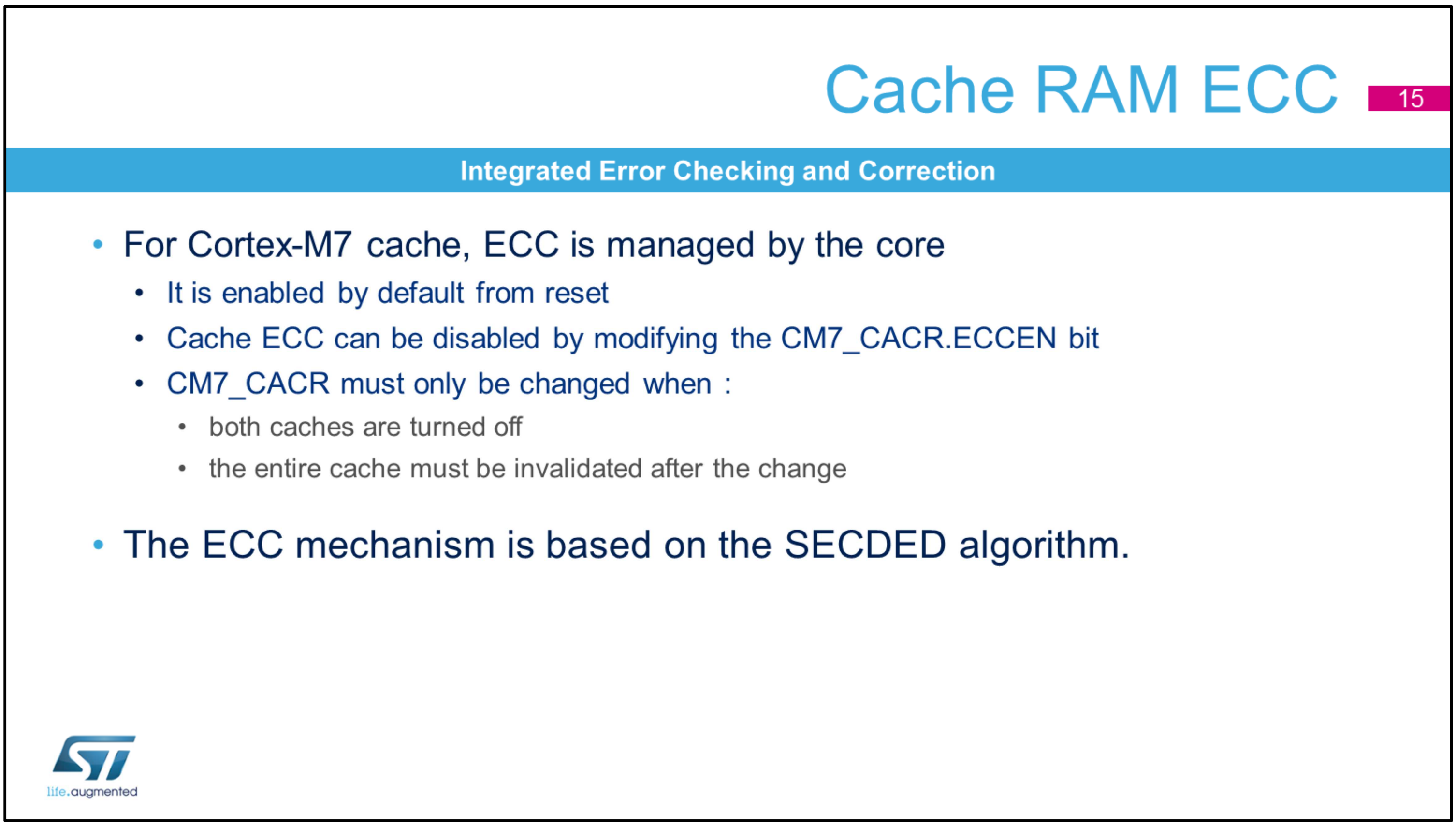
When the cache is disabled or non-cacheable or shared memory attributes are set, the accesses are performed directly to the memory using the AXIM interface.



L1 cache memory on AXI-M

The STM32H7 microcontroller implements 16 Kbyte for instruction and data caches with ECC protection on cache RAMs. The L1 cache on STM32H7 offers fast access to frequently used code and data from the next level of lower-speed memories such as external memories.
Both caches use a line-length of 256 bits (32 bytes) using a four-way set-associative scheme for data cache, and two-way set-associative scheme for the instruction cache.
A set is a group of contiguous lines aligned with an appropriate boundary (64 bytes for 2 way, 128 bytes for 4 way). Sets allow for a faster search for an address to determine if it is cached or not. The cached instructions or data are fetched from external memory using the AXIM interface.
No hardware coherency is supported, software needs to

manage cache maintenance by invalidating and cleaning cache lines before use. Or an easier way to maintain data coherency is to mark regions as “shared”. It prevents these regions from being cached in D-Cache but this will result in lower performance since all accesses go to next level memory.



Cache RAM ECC

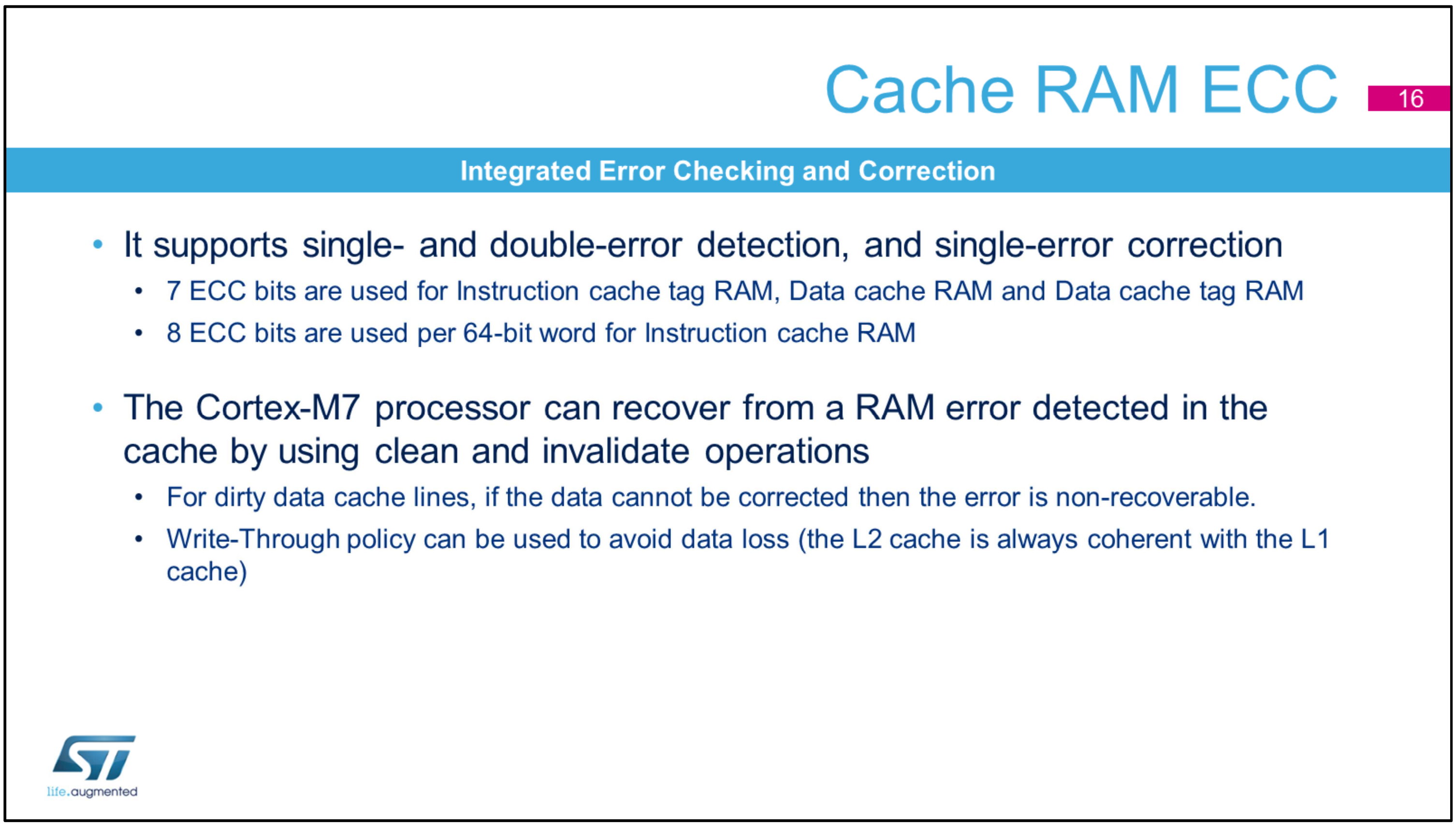
15

Integrated Error Checking and Correction

- For Cortex-M7 cache, ECC is managed by the core
  - It is enabled by default from reset
  - Cache ECC can be disabled by modifying the CM7_CACR.ECCEN bit
  - CM7_CACR must only be changed when :
    - both caches are turned off
    - the entire cache must be invalidated after the change
- The ECC mechanism is based on the SECDED algorithm.

life.augmented

On STM32H7 devices, the Cache RAM ECC protection uses the SECDED algorithm. Cache RAM protection is managed by the core, and is enabled by default after a reset.
Cache configurations must be changed only when the caches are turned OFF. A cache flush must be performed after changing the ECC protection settings.

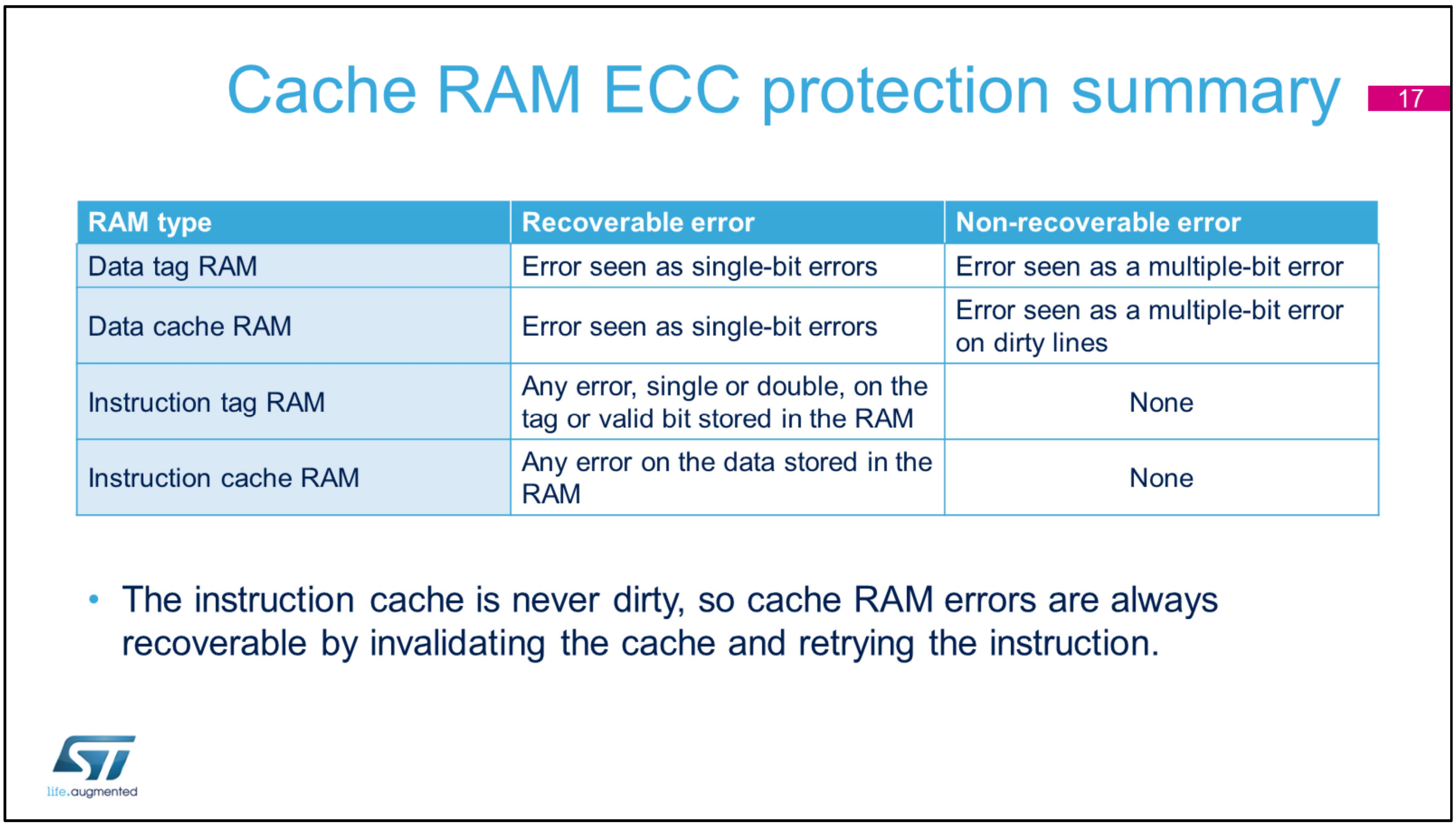


Cache RAM ECC

Integrated Error Checking and Correction

- It supports single- and double-error detection, and single-error correction
  - 7 ECC bits are used for Instruction cache tag RAM, Data cache RAM and Data cache tag RAM
  - 8 ECC bits are used per 64-bit word for Instruction cache RAM
- The Cortex-M7 processor can recover from a RAM error detected in the cache by using clean and invalidate operations
  - For dirty data cache lines, if the data cannot be corrected then the error is non-recoverable.
  - Write-Through policy can be used to avoid data loss (the L2 cache is always coherent with the L1 cache)

The cache RAMs implement 7 ECC bits for instruction tag, data tag and data. 8-bit ECC code is used for instruction cache RAM.
The ECC protection allows the Cortex-M7 to recover from a RAM error detected in runtime. The recovery uses a cache clean and invalidate mechanism.
For dirty data cache lines, if the data cannot be corrected then the error is non-recoverable. The Write-Through policy can be used to avoid data loss (the L2 cache is always coherent with the L1 cache).



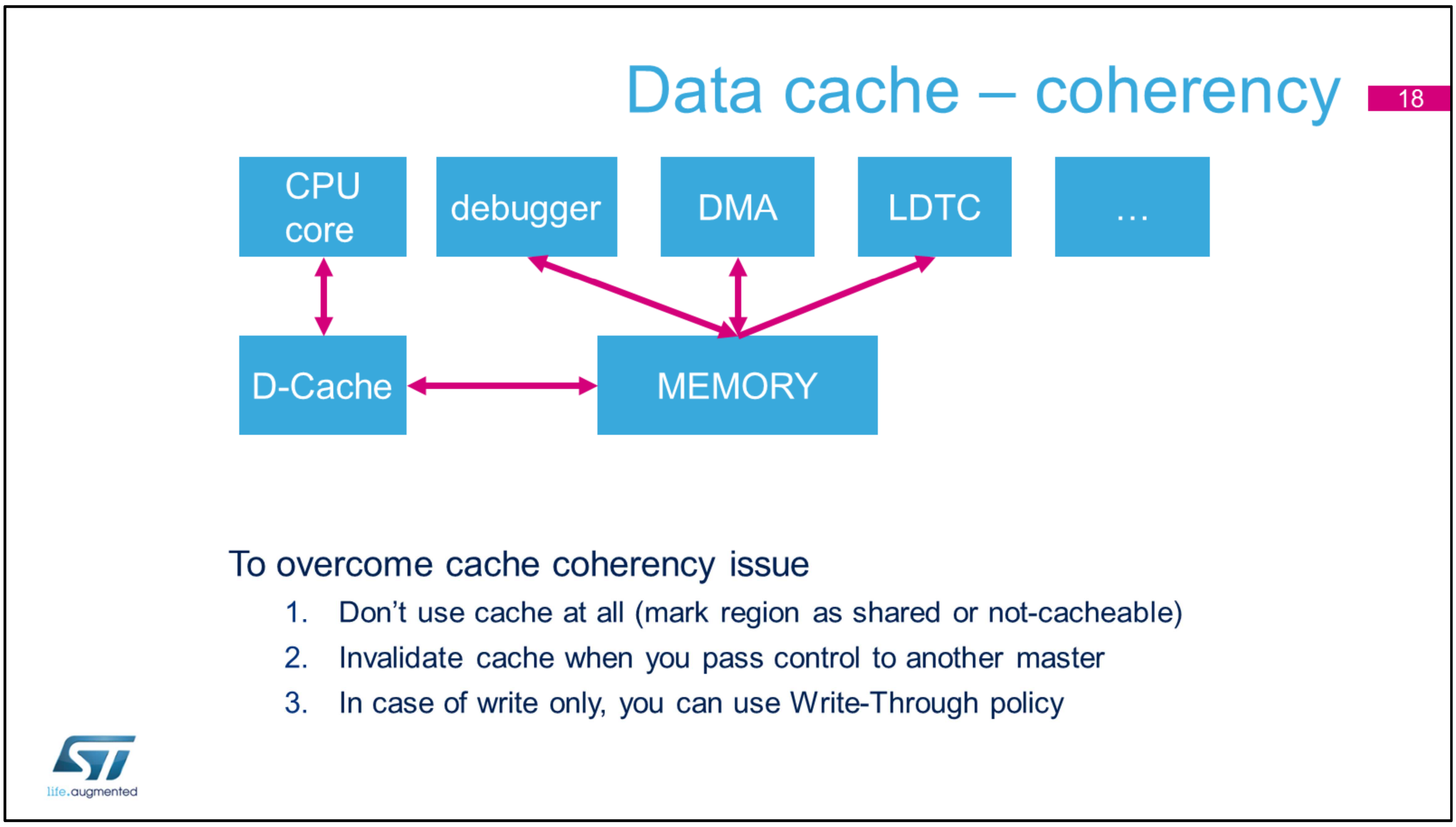
Cache RAM ECC protection summary

RAM type	Recoverable error	Non-recoverable error
Data tag RAM	Error seen as single-bit errors	Error seen as a multiple-bit error
Data cache RAM	Error seen as single-bit errors	Error seen as a multiple-bit error on dirty lines
Instruction tag RAM	Any error, single or double, on the tag or valid bit stored in the RAM	None
Instruction cache RAM	Any error on the data stored in the RAM	None

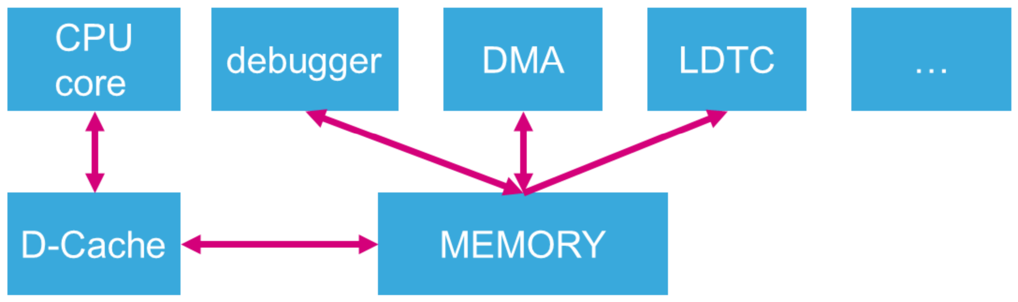
- The instruction cache is never dirty, so cache RAM errors are always recoverable by invalidating the cache and retrying the instruction.

life.augmented

This is a summary of cache RAM ECC protection and recoverable errors.



Data cache – coherency



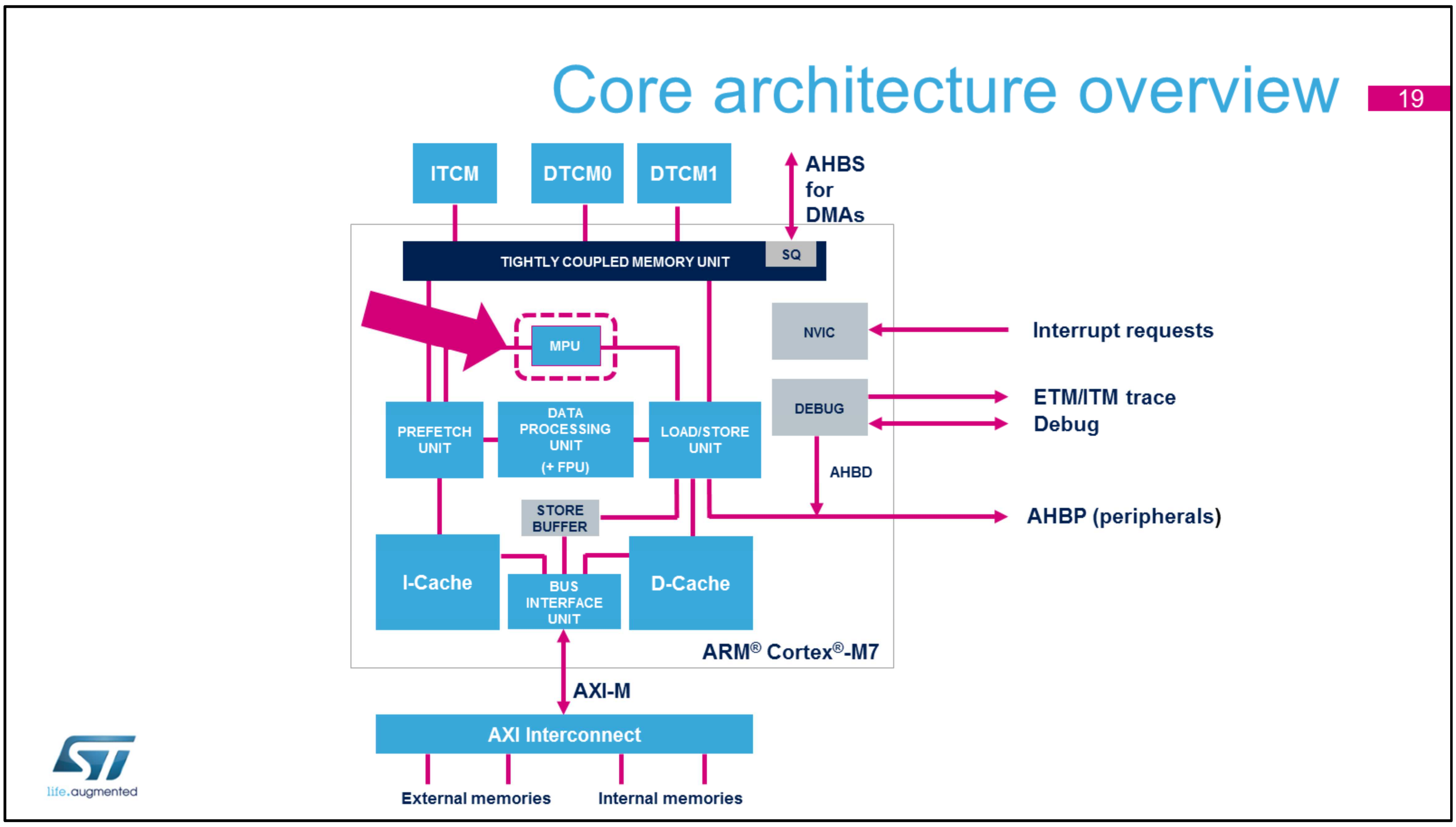
To overcome cache coherency issue

1. Don’t use cache at all (mark region as shared or not-cacheable)
2. Invalidate cache when you pass control to another master
3. In case of write only, you can use Write-Through policy

life.augmented

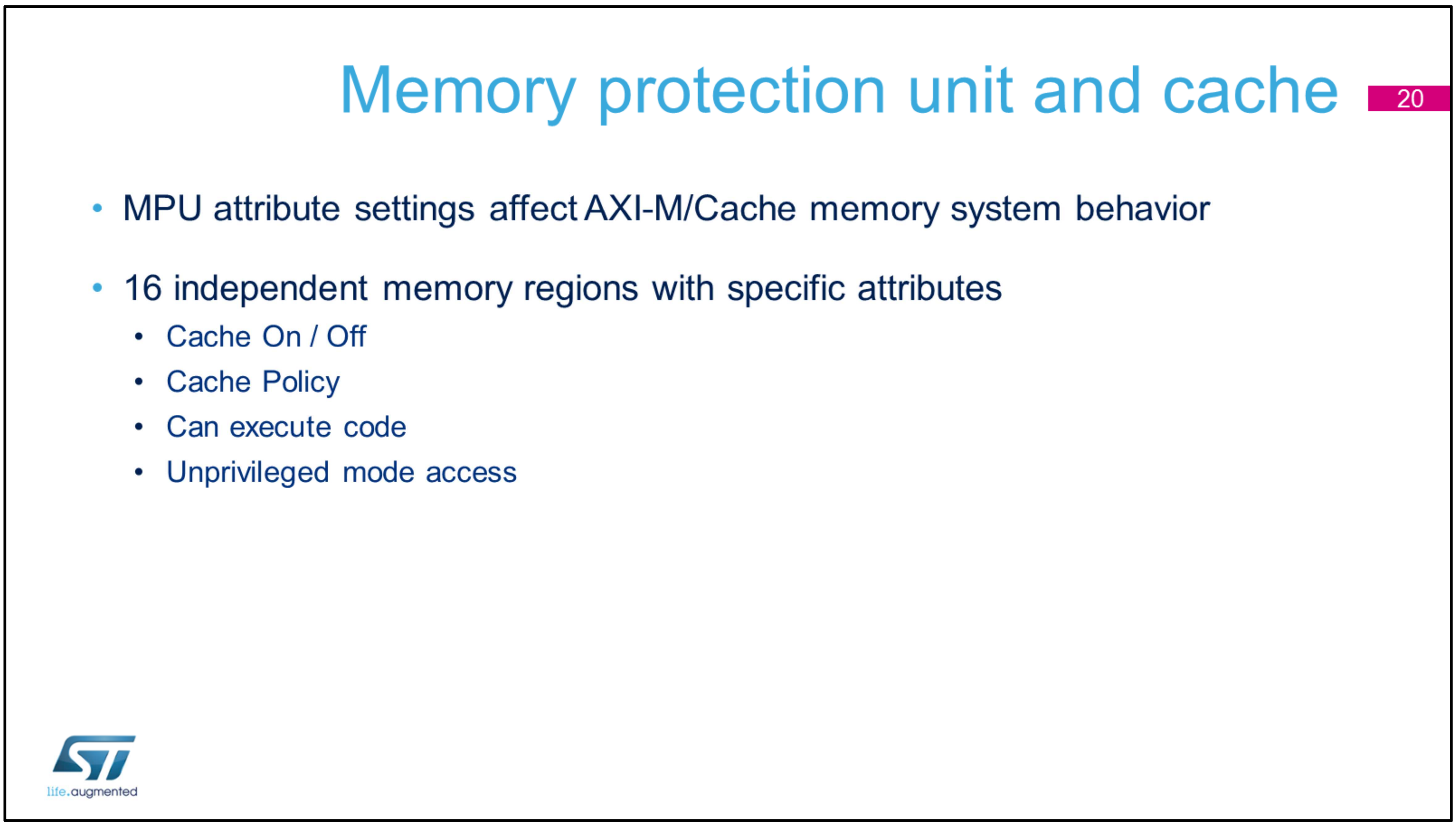
Three solutions exist to overcome cache coherency issue when different masters (for example, DMAs) share the same memory buffers as the core:

- The first solution is to mark regions as “shared” to prevent these regions from being cached in D-Cache
- The second solution is to clean or invalidate cache when software passes or gets control over memory buffers. There is CMSIS functions to do all steps to clean and/or invalidate caches
- The third solution is to use a write-through policy for write-only memory buffers where the CPU is the data producer.



Core architecture overview

In the Cortex®-M7 core, the Memory Protection Unit (MPU) is used to configure the behavior of the cache controllers and the AXIM interface, enforce access rules and separate processes.



Memory protection unit and cache

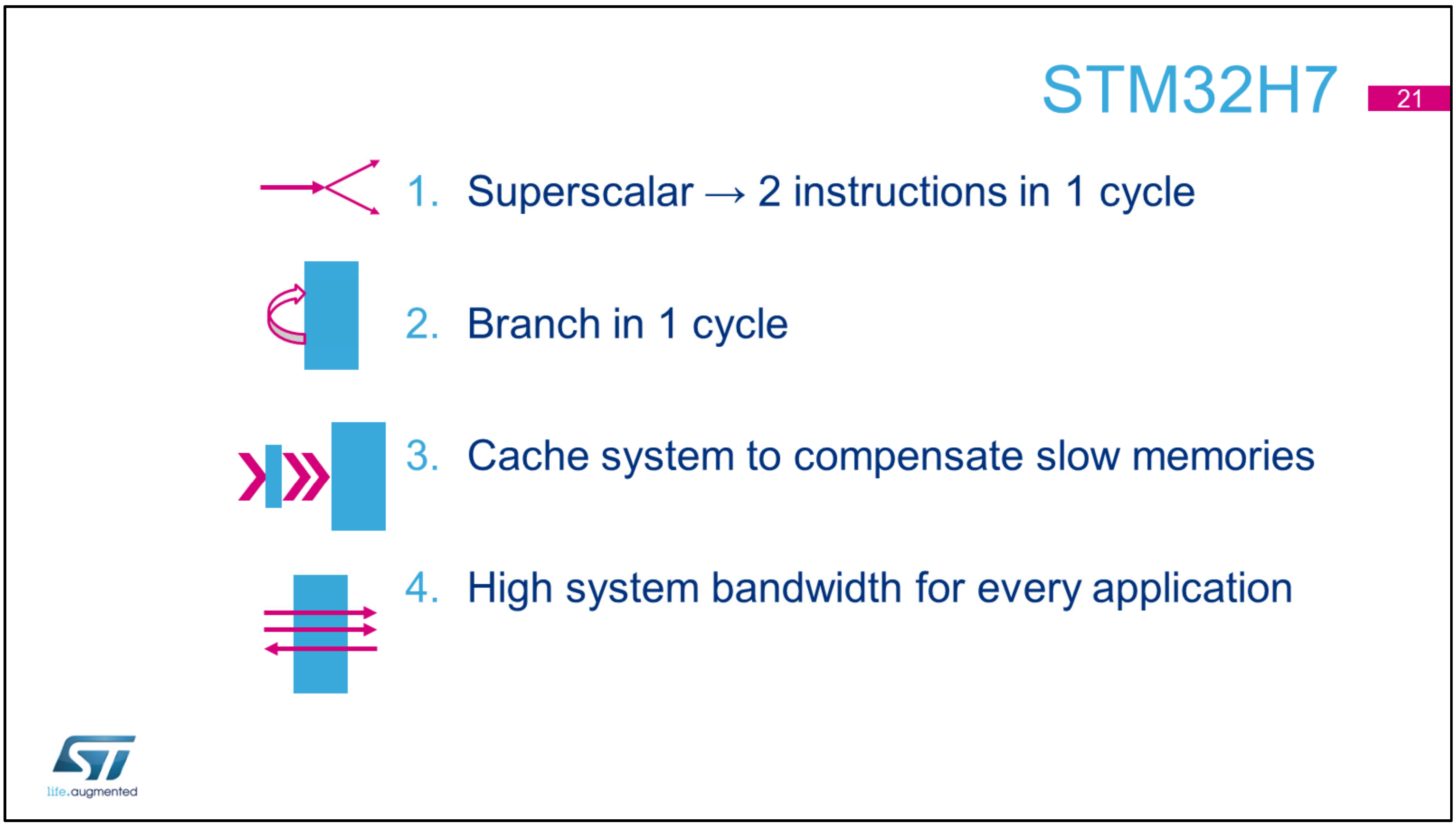
20

- MPU attribute settings affect AXI-M/Cache memory system behavior
- 16 independent memory regions with specific attributes
  - Cache On / Off
  - Cache Policy
  - Can execute code
  - Unprivileged mode access

life.augmented

The Memory Protection Unit (MPU) in STM32H7 microcontroller offers support for 16 independent memory regions, with independent configurable attributes for:

- access permission: for read/write permissions in privileged/unprivileged modes,
- execution permission: executable region or region prohibited for instruction fetch, and
- cache policy.

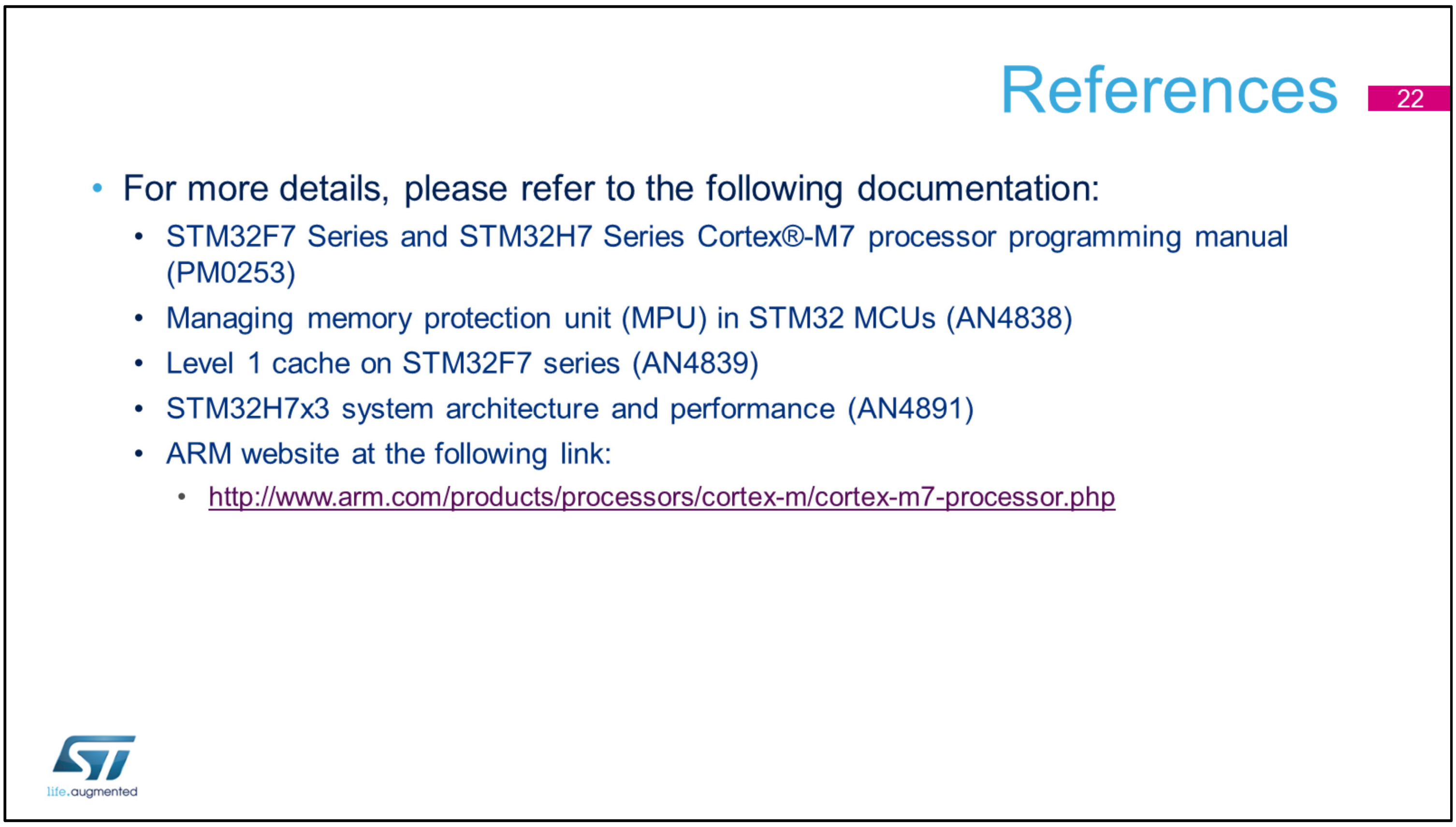


STM32H7

1. Superscalar → 2 instructions in 1 cycle
2. Branch in 1 cycle
3. Cache system to compensate slow memories
4. High system bandwidth for every application

life.augmented

The STM32H7 microcontrollers design benefits from the new features of the Cortex®-M7 core such as memory interfaces, the cache system that compensates slow memories and superscalar architecture to offer high processing bandwidth for every application while keeping good responsiveness.



References

- For more details, please refer to the following documentation:
  - STM32F7 Series and STM32H7 Series Cortex®-M7 processor programming manual (PM0253)
  - Managing memory protection unit (MPU) in STM32 MCUs (AN4838)
  - Level 1 cache on STM32F7 series (AN4839)
  - STM32H7x3 system architecture and performance (AN4891)
  - ARM website at the following link:
    - http://www.arm.com/products/processors/cortex-m/cortex-m7-processor.php

life.augmented

For more details, please refer to these application notes and the Cortex®-M7 programming manual available on www.st.com website.
Also visit the ARM website in which you will find more information about the Cortex®-M7 core.