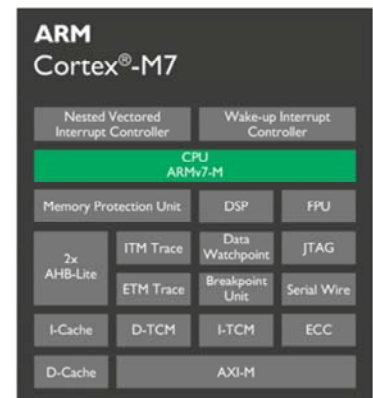# STM32F7- ARM® Core

ARM Cortex®-M7 Core

Revision 1.0

Hello, and welcome to this presentation of the ARM®
Cortex®-M7 core which is embedded in all products of
the STM32F7 microcontroller family.

Cortex-M7 processor overview

- **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**
  - ⟹ For STM32F74x and STM32F75x, Single-precision only

ARM Cortex®-M7

Nested Vectored Interrupt Controller		Wake-up Interrupt Controller	
CPU ARMv7-M			
Memory Protection Unit		DSP	FPU
2x AHB-Lite	ITM Trace	Data Watchpoint	JTAG
	ETM Trace	Breakpoint Unit	Serial Wire
I-Cache	D-TCM	I-TCM	ECC
D-Cache	AXI-M		

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

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.
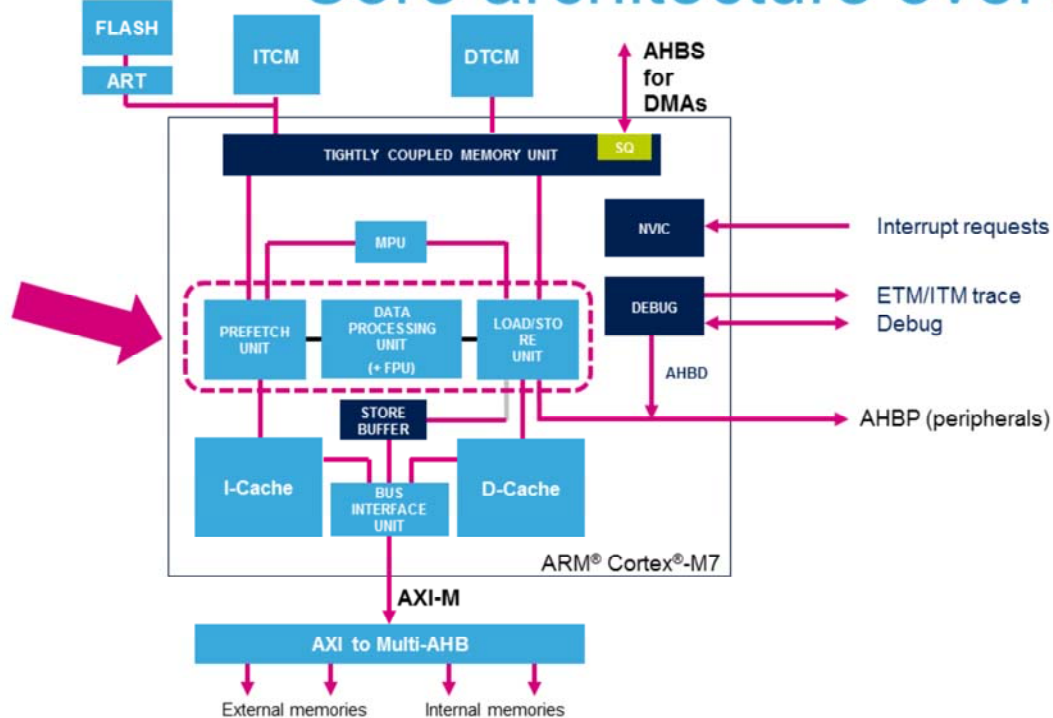
STM32F7 microcontrollers integrate an ARM® Cortex®-M7 core in order to benefit from the improved performance of the Cortex®-M processors architecture and particularly of the high level of performance in low-power modes.

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
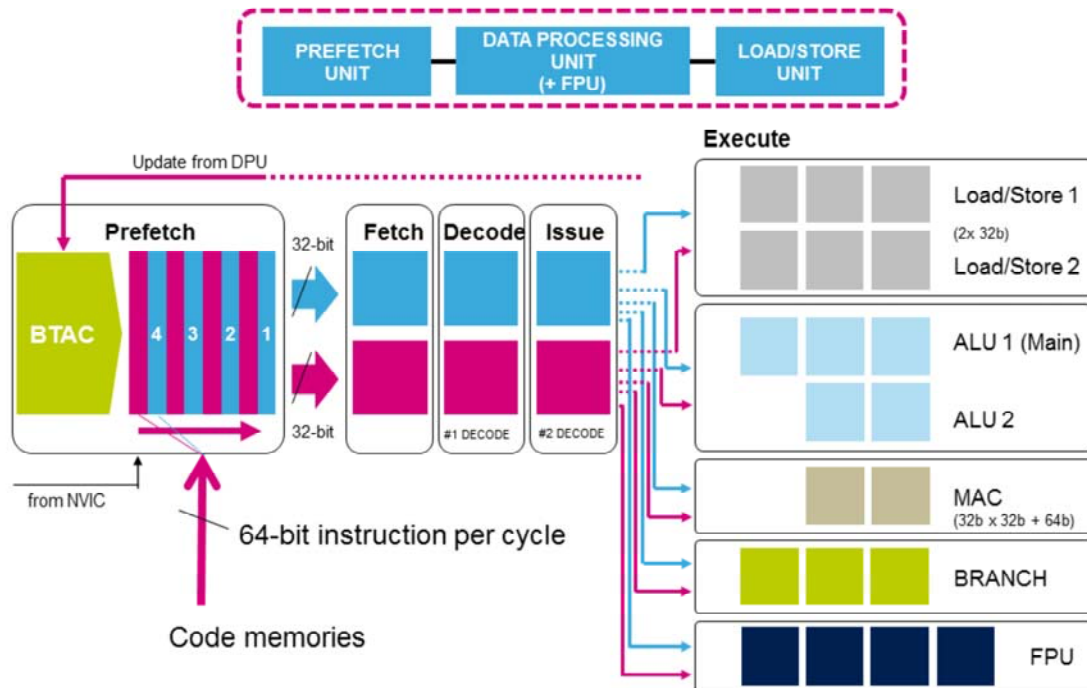
The prefetch unit (PFU) provides one 64-bit instruction per cycle to the **D**ata **P**rocessing **U**nit (DPU).
It includes:
- a buffer of 4 entries of 64 bits each to enable fetching ahead of the DPU
- and **B**ranch **T**arget **A**ddress **C**ache (BTAC) for single cycle branch prediction

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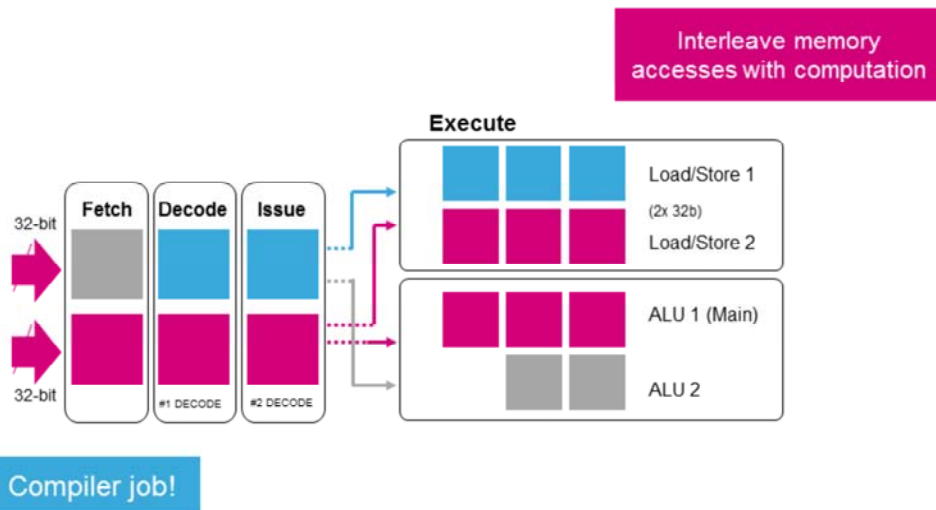
The Load Store Unit (LSU) provides either dual 32-bit load channels or a single 64-bit store channel with store buffering to increase store throughput .

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

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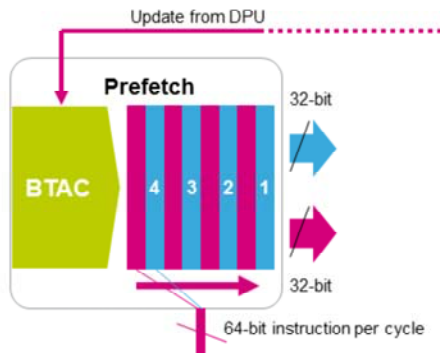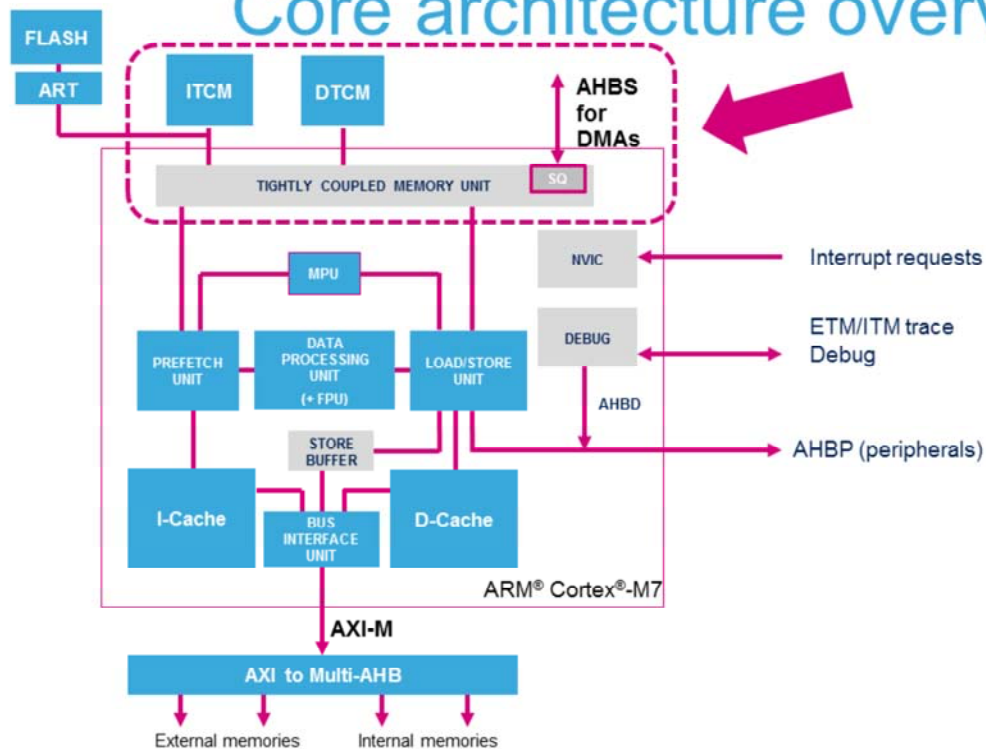
As a branch can also be dual issued, it can be executed in parallel with computation.

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.
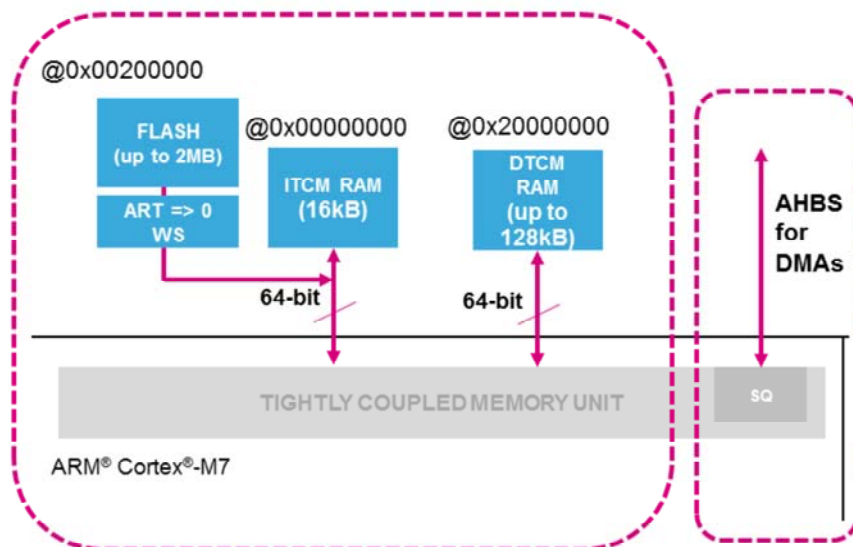
Tightly-Coupled Memories (TCM) are dedicated memories directly connected to the processor, but not through a bus, so avoiding arbitration and latencies for frequently executed code.

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.

Tightly-coupled memories (TCM)

@0x00200000

FLASH (up to 2MB)

ART => 0 WS

@0x00000000

ITCM RAM (16kB)

@0x20000000

DTCM RAM (up to 128kB)

64-bit

64-bit

AHBS for DMAs

TIGHTLY COUPLED MEMORY UNIT

SQ

ARM® Cortex®-M7

TCM memories are strictly 0 wait states

The ITCM RAM has one 64-bit memory interface to satisfy core fetch bandwidth.
STM32F7 microcontrollers enable access to Flash memory devices on the ITCM interface and include an integrated ART Flash accelerator for best performance. A 16-Kbyte SRAM is also accessible over the ITCM bus.
The DTCM RAM has two 32-bit memory interfaces to ensure more parallelism on request. Depending on the STM32F7 device, 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 ["AMBA" "3"] 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 (16 Kbytes)
- Critical code
- Interrupt service routines
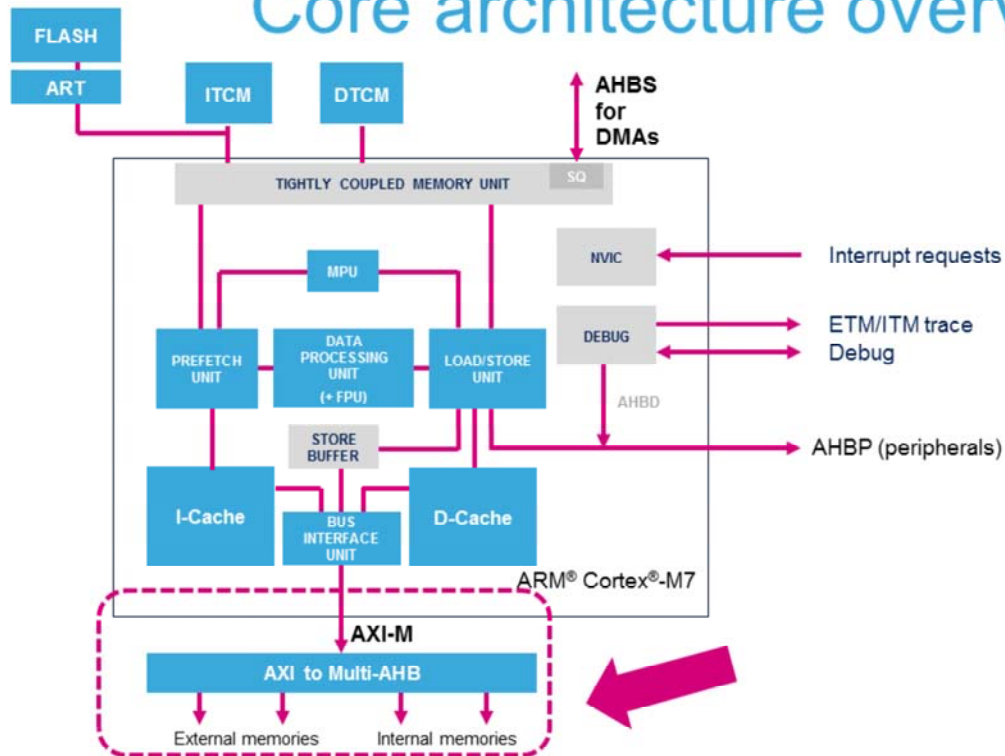- Highly deterministic

DTCM RAM (up to 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 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.
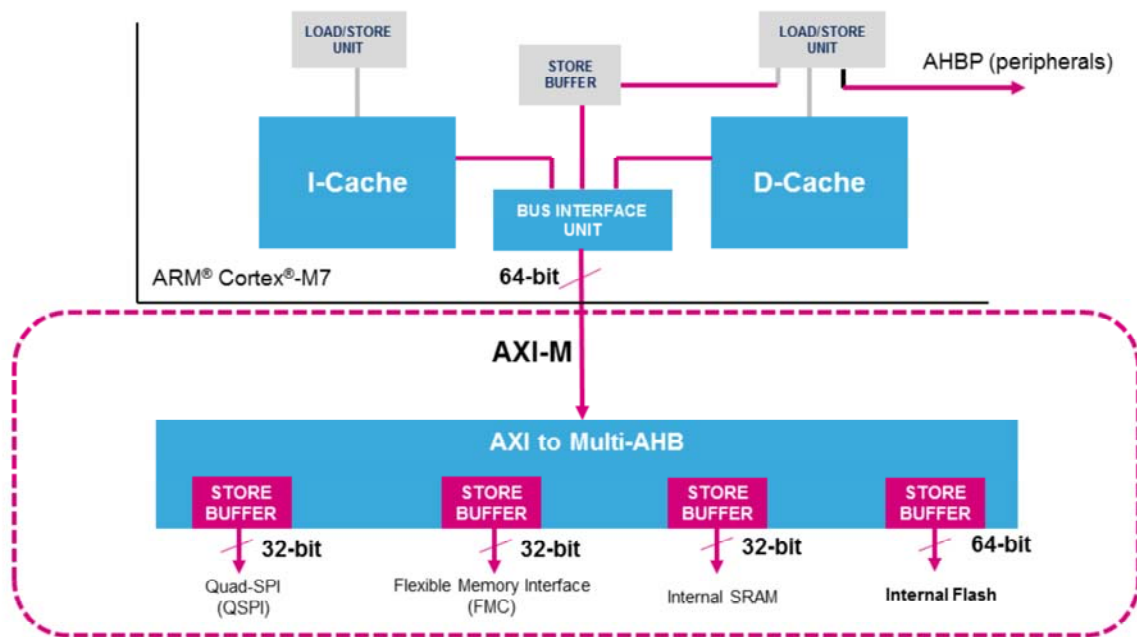
STM32F7 microcontrollers integrate an AXI-to-Multi-AHB bridge to take advantage of the Cortex®-M7 AXIM interface.
The AXI de-correlates the access request from the data phase, so the request is independent from its corresponding data. If the memory has latencies, this separation makes the bus available to perform a new request if no functional relationship exists between the two requests (example Instruction Fetch & Data Fetch are performed in parallel).

The STM32F7 AXI master runs at the same frequency as the core, and with the AXI-to-Multi-AHB bridge running at the same frequency, it optimizes memory system performance even in case of latencies on external memories.

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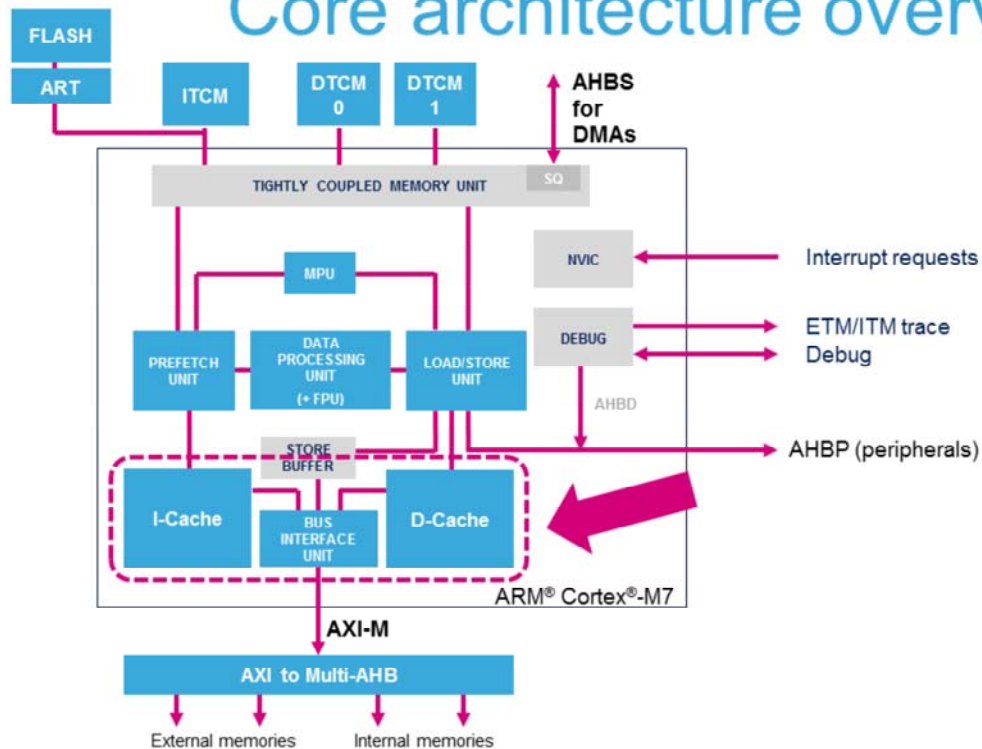
In addition to the AXIM interface, the Cortex®-M7 core integrates optional Instruction and Data caches for efficient memory access.

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 cache-ability criteria is fulfilled. When cache is disabled or non-cacheable or shared memory attributes are set, the accesses are performed directly to the memory using the AXIM interface.
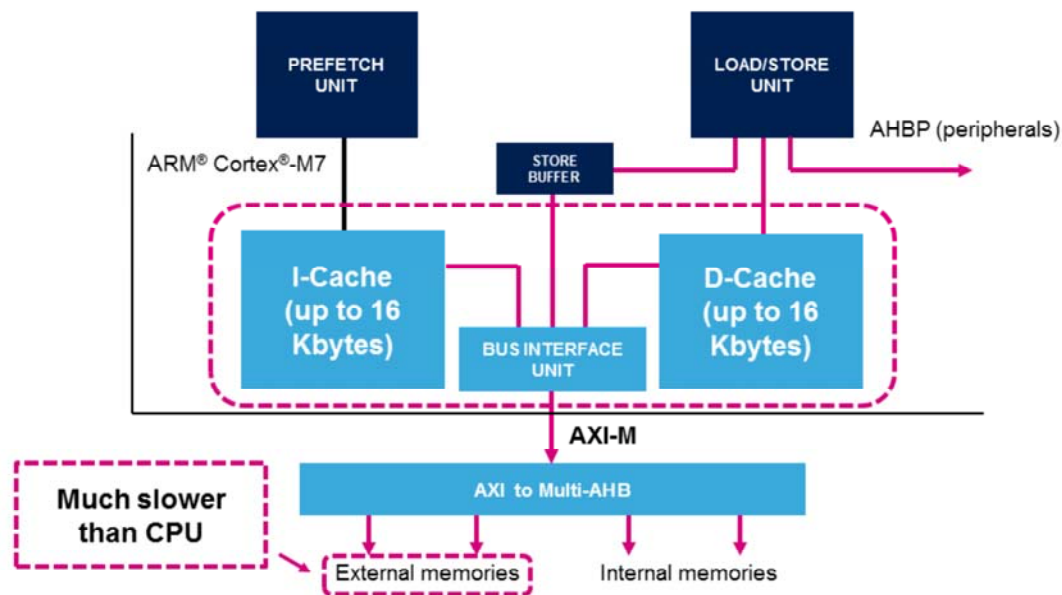
Depending on STM32F7 microcontroller, the instruction and data cache sizes vary from 4Kbytes to 16Kbytes. The L1 cache on STM32F7 offers fast access to frequently used code and data from the next level of lower-speed memories such as external memories.

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.

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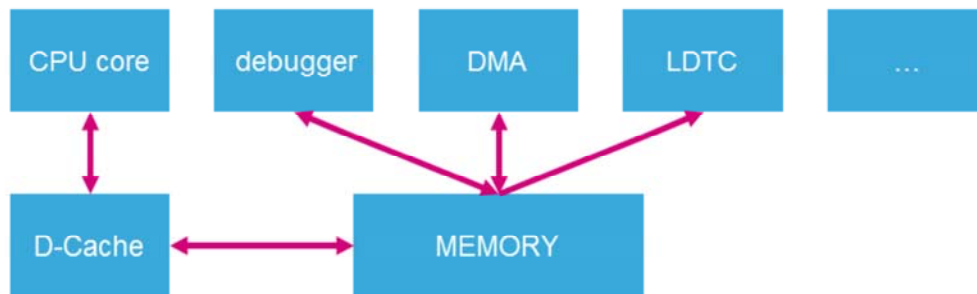
To overcome cache coherency issue
1. Don't use cache at all (mark region as shared or not-cacheable)
2. Invalidate cache when you pass control to another master
3. In case of write only, you can use Write-Through policy

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 buffer where the CPU is the data producer.

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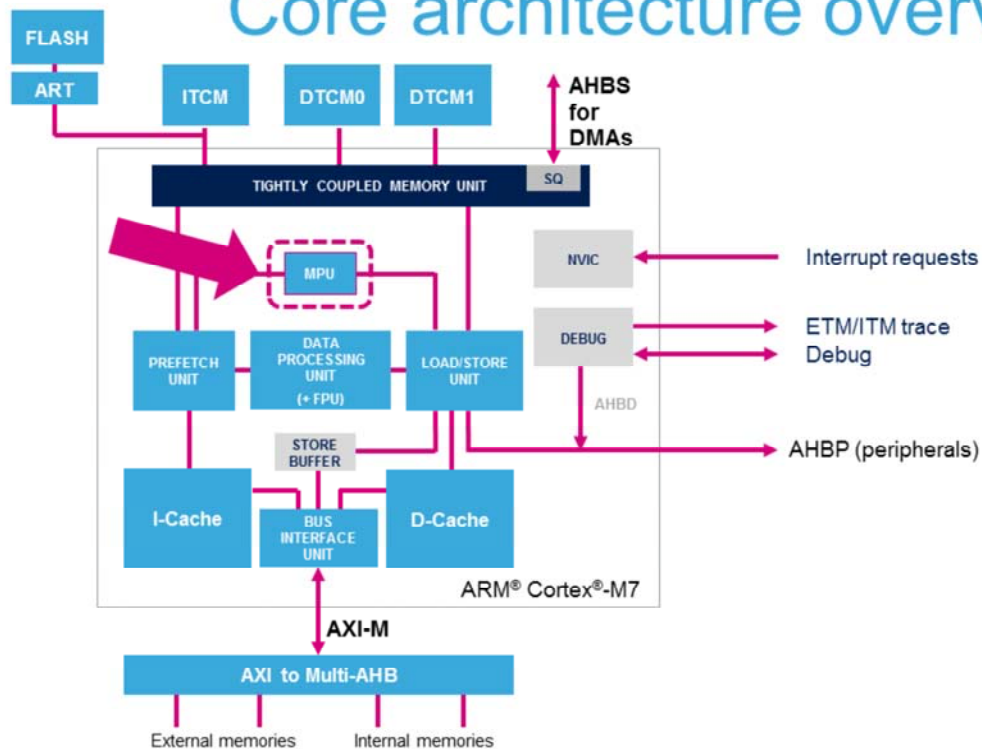
In the Cortex®-M7 core, the Memory Protection Unit (MPU) is used to configure the behavior of the cache controllers and the AXIM interface, enforce access rules and separate processes.
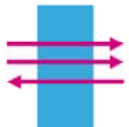
Memory protection unit and cache

- MPU attributes settings **affect** AXI-M/Cache memory system **behavior**

- **8 independent memory regions**
    - Cache On / Off
    - Cache Policy
    - Can execute code?
    - Unprivileged mode access?

The MPU in STM32F7 microcontroller offers support for eight independent memory regions, with independent configurable attributes for:
- access permission: allowed or not read/write/in privileged/or in unprivileged mode,
- execution permission: executable region or region prohibited for instruction fetch,
- and cache policy.

The STM32F7's 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.

- For more details, please refer to the following documentation:
  - STM32F7 Series Cortex®-M7 processor programming manual (PM0253)
  - Managing memory protection unit (MPU) in STM32 MCUs (AN4838)
  - Level 1 cache on STM32F7 series (AN4839)
  - STM32F7 series system architecture and performance (AN4667)
  - ARM website at the following link:
    - http://www.arm.com/products/processors/cortex-m/cortex-m7-processor.php

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.