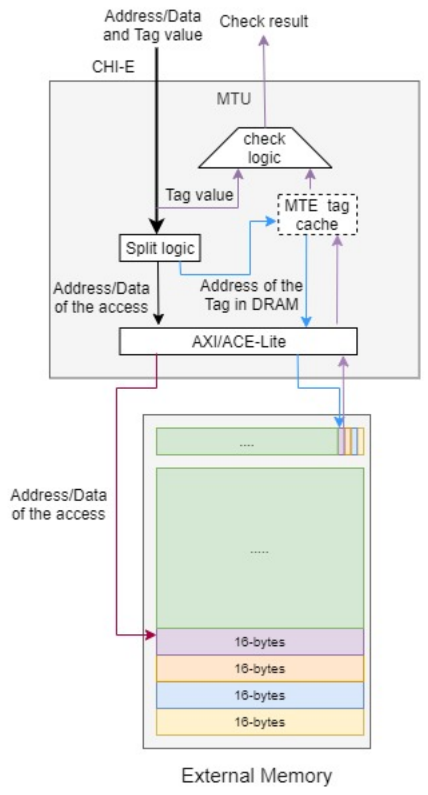
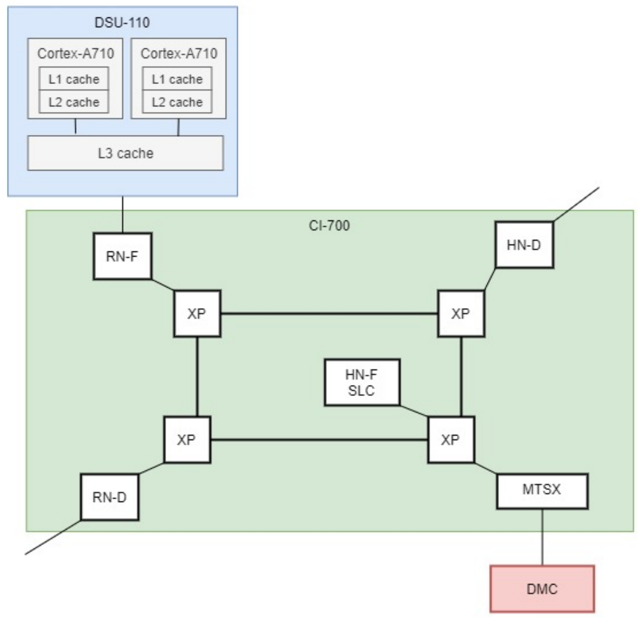
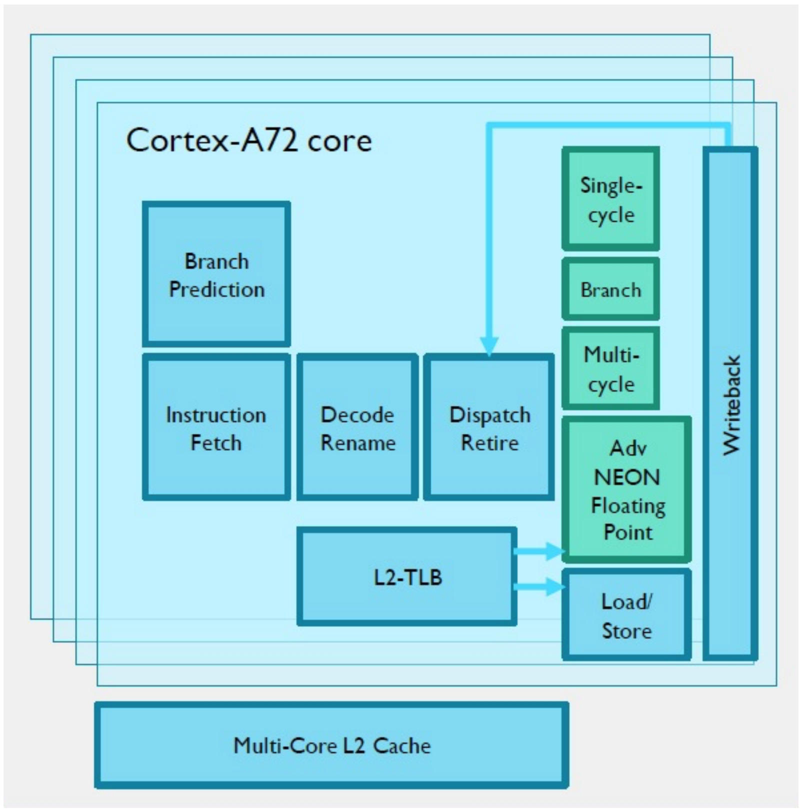
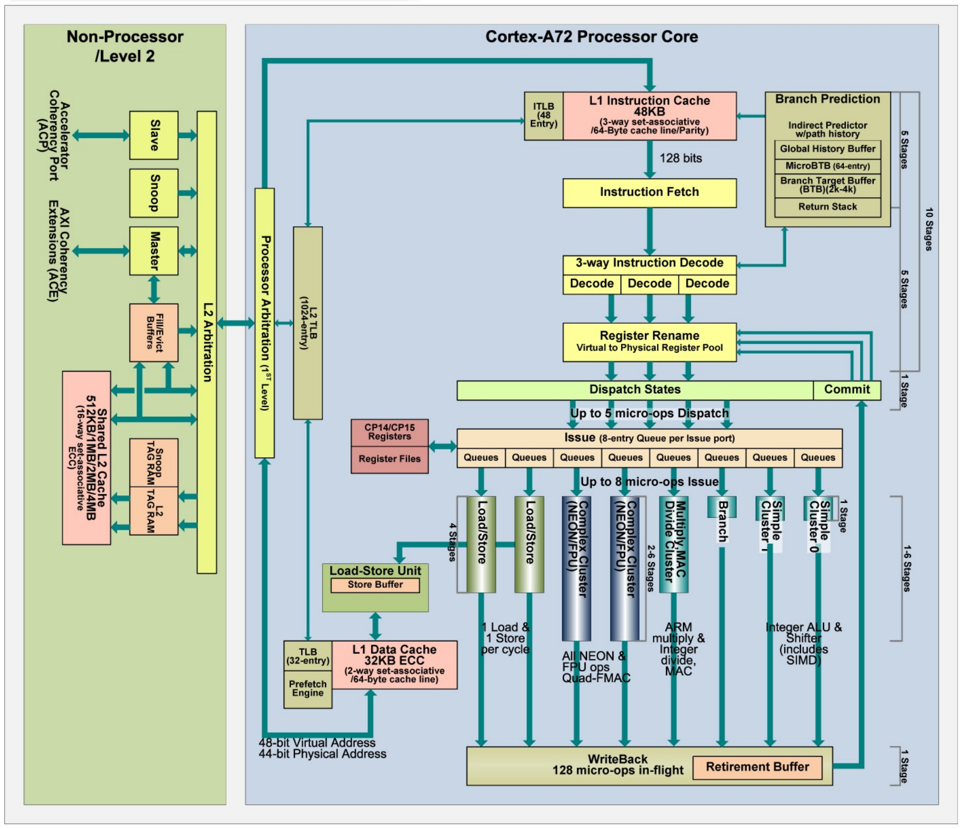
MTE Notes


Address/Data
and Tag value
Check result
CHI-E
MTU
check
logic
Tag value
MTE tag
cache
Split logic
Address/Data
of the access
Address of the
Tag in DRAM
AXI/ACE-Lite
Address/Data
of the access
16-bytes
16-bytes
16-bytes
16-bytes
External Memory


DSU-110
Cortex-A710
L1 cache
L2 cache
Cortex-A710
L1 cache
L2 cache
L3 cache
CI-700
RN-F
HN-D
XP
XP
HN-F
SLC
XP
XP
RN-D
MTSX
DMC



ARM Cortex-A72 Block Diagram


Non-Processor /Level 2
Accelerator Coherency Port (ACP)
AXI Coherency Extensions (ACE)
Slave
Snoop
Master
Fill/Evict Buffers
Snoop TAG RAM
L2 TAG RAM
Shared L2 Cache 512KB/1MB/2MB/4MB (16-way set-associative ECC)
L2 Arbitration
Cortex-A72 Processor Core
Processor Arbitration (1st Level)
L2 TLB (1024-entry)
ITLB (48 Entry)
L1 Instruction Cache 48KB (3-way set-associative /64-Byte cache line/Parity)
128 bits
Instruction Fetch
Branch Prediction
Indirect Predictor w/path history
Global History Buffer
MicroBTB (64-entry)
Branch Target Buffer (BTB)(2k-4k)
Return Stack
5 Stages
10 Stages
5 Stages
1 Stage
3-way Instruction Decode
Decode
Decode
Decode
Register Rename
Virtual to Physical Register Pool
Dispatch States
Commit
Up to 5 micro-ops Dispatch
CP14/CP15 Registers
Register Files
Issue (8-entry Queue per Issue port)
Queues
Up to 8 micro-ops Issue
Load/Store
Load/Store
Complex Cluster (NEON/FPU)
Complex Cluster (NEON/FPU)
Multiply,MAC Divide Cluster
Branch
Simple Cluster 1
Simple Cluster 0
4 Stages
2-6 Stages
1 Stage
1-6 Stages
1 Load & 1 Store per cycle
All NEON & FPU ops Quad-FMAC
ARM multiply & Integer divide, MAC
Integer ALU & Shifter (includes SIMD)
Load-Store Unit
Store Buffer
TLB (32-entry)
Prefetch Engine
L1 Data Cache 32KB ECC (2-way set-associative /64-byte cache line)
48-bit Virtual Address
44-bit Physical Address
WriteBack 128 micro-ops in-flight
Retirement Buffer
1 Stage

Copyright (c) 2015 Hiroshige Goto All rights reserved.

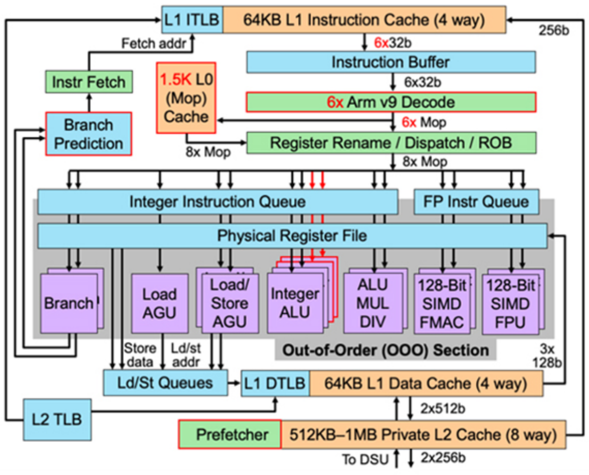


Figure 3. Cortex-X3 microarchitecture

MTE memory tags are stored in the Cortex-A710 L1 data cache, L2 cache, DSU L3 cache and CI-700 HN-F system level cache (SLC). For example, in L1 data cache, the 4-bit MTE tag is stored in the cache line tag ram.

▼ Swap:

▼ MTE Tag Compression for swapped pages 11/13/2023

Currently, when MTE pages are swapped out, the tags are kept in the memory, occupying PAGE_SIZE/32 bytes per page. This is especially problematic for devices that use zram backed in-memory swap, because tags stored uncompressed in the heap effectively reduce the available amount of swap memory.

The RLE-based algorithm suggested by Evgenii Stepanov and implemented in this patch series is able to efficiently compress fixed-size tag buffers, resulting in practical compression ratio of 2x. In many cases it is possible to store the compressed data in 63 bit Xarray values, resulting in no extra memory allocations

▼ Support THP_SWAP on hardware with MTE 12/8/2023

brings up THP_SWAP on ARM64, but it doesn't enable THP_SWP on hardware with MTE as the MTE code works with the assumption tags save/restore is always handling a folio with only one page.

https://github.com/torvalds/linux/blob/master/arch/arm64/mm/mteswap.c

https://github.com/torvalds/linux/blob/master/arch/arm64/mm/copypage.c

▼ Functions

Name	Parameters	What it does
`mte_sync_tags`	`pte_t pte, unsigned int nr_pages`	Synchronizes memory tags. For each page represented by the page table entry `pte`, it checks if the page has been tagged, initializes tags if necessary, and ensures that the tags are visible before updating the page table entry.
`memcmp_pages`	`struct page *page1, struct page *page2`	Compares the content of two pages. If memory tagging extensions (MTE) are supported and enabled, it checks if the pages are identical but tagged differently, in which case it returns a non-zero value to avoid memory merging.
`mte_enable_kernel_sync`	None	Enables MTE synchronization mode at EL1 (exception level 1) for the kernel.

`mte_enable_kernel_async`	None	Enables MTE asynchronous mode at EL1 for the kernel.
`mte_enable_kernel_asymm`	None	Enables MTE asymmetric mode at EL1 for the kernel, if supported by the CPU. If not supported, falls back to synchronous mode.
`mte_check_tfsr_el1`	None	Checks if a tag fault has occurred at EL1 (exception level 1) and reports it if necessary.
`mte_update_sctlr_user`	`struct task_struct *task`	Updates the system control register SCTLR_EL1 for user-mode tasks based on the MTE mode configuration.
`mte_update_gcr_excl`	`struct task_struct *task`	Updates the tagged memory exclusion setting in the system control register GCR_EL1 for user-mode tasks.
`mte_thread_init_user`	None	Initializes MTE-related settings for a user-mode thread.
`mte_thread_switch`	`struct task_struct *next`	Handles MTE-related operations during a thread context switch. Updates the system control registers and checks for asynchronous tag exceptions at EL1.
`mte_cpu_setup`	None	Sets up the CPU for MTE usage, including configuring memory attributes, initializing system registers, and flushing TLB entries.
`mte_suspend_enter`	None	Prepares for entering a system suspend state, including checking for pending tag faults.
`mte_suspend_exit`	None	Restores MTE settings after exiting a system suspend state.
`set_mte_ctrl`	`struct task_struct *task, unsigned long arg`	Sets the MTE mode control for a task based on the specified arguments, and updates the corresponding system control registers accordingly.
`get_mte_ctrl`	`struct task_struct *task`	Retrieves the MTE mode control settings for a task.
`mte_ptrace_copy_tags`	`struct task_struct *child, long request, unsigned long addr, unsigned long data`	Allows copying MTE tags between processes for debugging purposes using `ptrace`.
`mte_tcf_preferred_show`	`struct device *dev, struct device_attribute *attr, char *buf`	Shows the preferred MTE mode (sync, async, or asymm) for a specific CPU device.
`mte_tcf_preferred_store`	`struct device *dev, struct device_attribute *attr, const char *buf, size_t count`	Sets the preferred MTE mode (sync, async, or asymm) for a specific CPU device based on the input string.
`register_mte_tcf_preferred_sysctl`	None	Registers a sysfs attribute to control the preferred MTE mode for each CPU device.
`mte_probe_user_range`	`const char __user *uaddr, size_t size`	Probes a user address range to determine if it's accessible and properly aligned for MTE operations.

▼ Setting Tags:

▼ Checking Tags

- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/include/asm/thr
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64
-

▼ Kernel Code:

- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/kernel/mte.c#L7 ✅
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/include/asm/mt 👎
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/include/asm/mt def.h#L11 👎
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64 👎
- https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64 👎

▼ Bootloader Code:

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64/mt ✅

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/lib/mte.S#L98 ✅

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/kernel/entry.S#L155 👎

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/lib/strlen.S#L11 👎

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/lib/strcmp.S#L15 👎

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/lib/strncmp.S#L15 👎

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/mm/proc.S#L51 👎

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/include/uapi/linux/elf.h#L45 👎

https://android.googlesource.com/platform/system/extras/+/main/mtectrl/mtectrl.cc

Name	Parameters	What it does
mte.S		
`mte_clear_page_tags`	`void *addr`	Clears the tags in a page by repeatedly clearing the tags in memory blocks of a certain size.
`mte_zero_clear_page_tags`	`void *addr`	Zeroes the page and tags simultaneously. If supported by the architecture, it uses the DC GZVA instruction to zero the page. Otherwise, it uses the STZ2G instruction to store zero values into the page and tags.
`mte_copy_page_tags`	`void *dst_addr, void *src_addr`	Copies tags from a source page to a destination page.
`mte_copy_tags_from_user`	`void *to, const void *from, size_t n`	Reads tags from a user buffer and sets the corresponding tags at the given kernel address.
`mte_copy_tags_to_user`	`void *to, const void *from, size_t n`	Gets tags from a kernel address range and writes the tag values to the given user buffer.

`mte_save_page_tags`	`void *page_addr, void *tag_storage`	Saves the tags in a page to a provided storage buffer.
`mte_restore_page_tags`	`void *page_addr, void *tag_storage`	Restores the tags in a page from a provided storage buffer.
mte_helper.S		
mte_insert_random_tag	void *ptr	Inserts a random tag into the given pointer. If the source pointer has a tag, the inserted tag might be the same as the source tag.
mte_insert_new_tag	void *ptr	Inserts a new tag into the given pointer. If the source pointer has a tag, the inserted tag will be different from the source tag.
mte_get_tag_address	void *ptr	Retrieves the tag from the given address.
mte_set_tag_address_range	void *ptr, size_t range	Sets the tag range starting from the given address for the specified range.
mte_clear_tag_address_range	void *ptr, size_t range	Clears the tag range starting from the given address for the specified range.
mte_enable_pstate_tco	None	Enables the PSTATE.TCO (Tag Check Override) field.
mte_disable_pstate_tco	None	Disables the PSTATE.TCO (Tag Check Override) field.
mte_get_pstate_tco	None	Retrieves the value of the PSTATE.TCO (Tag Check Override) field.
mte_common_utils.c		
mte_default_handler	int signum, siginfo_t *si, void *uc	Handles SIGSEGV and SIGBUS signals, determines fault validity, adjusts pc
mte_register_signal	int signal, void (*handler)(int, siginfo_t *, void *)	Registers signal handler for a specific signal
mte_wait_after_trig	None	Yields the CPU to other processes after triggering a fault
mte_insert_tags	void *ptr, size_t size	Inserts memory tags at the specified pointer with the given size
mte_clear_tags	void *ptr, size_t size	Clears memory tags at the specified pointer with the given size
mte_allocate_memory_tag_range	size_t size, int mem_type, int mapping, size_t range_before, size_t range_after	Allocates memory with tagged addresses within a specified range
mte_allocate_memory	size_t size, int mem_type, int mapping, bool tags	Allocates memory with tagged addresses
mte_allocate_file_memory	size_t size, int mem_type, int mapping, bool tags, int fd	Allocates file memory with tagged addresses
mte_allocate_file_memory_tag_range	size_t size, int mem_type, int mapping, size_t range_before, size_t	Allocates file memory with tagged addresses within a specified range

	range_after, int fd	
mte_free_memory_tag_range	void *ptr, size_t size, int mem_type, size_t range_before, size_t range_after	Frees memory with tagged addresses within a specified range
mte_free_memory	void *ptr, size_t size, int mem_type, bool tags	Frees memory with tagged addresses
mte_initialize_current_context	int mode, uintptr_t ptr, ssize_t range	Initializes the context for a memory fault
mte_switch_mode	int mte_option, unsigned long incl_mask	Switches the MTE mode and inclusion mask for memory tagging
mte_default_setup	None	Sets up the default MTE environment
mte_restore_setup	None	Restores the MTE environment to its default state
create_temp_file	None	Creates a temporary file in the tmpfs filesystem

MTE used in other functions:

Makefile:
https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/tools/testing/selftests/arm64/mte/M

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/lib/maple_tree.c#L341

root tree?

dump_tag_range?
https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/arm64/kernel/elfcore.c#L24

hypervisor?
https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/arch/sparc/include/asm/hypervisor

.rst that talks about cpu registers?

https://github.com/torvalds/linux/blob/2668e3ae2ef36d5e7c52f818ad7d90822c037de4/Documentation/arch/arm64/cpu-feature-registers.rst#L180 mte_mem_type

- USE_MALLOC
- USE_MMAP
- USE_MPROTECT