SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

COIMBATORE-35

Accredited by NBA-AICTE and Accredited by NAAC – UGC with A++ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COURSE NAME: 19EEB303 / Microcontroller and its Applications

III YEAR / VI SEMESTER

Unit IV – ARM INSTRUCTION SET

Topic: Memory management unit

Memory management unit

The Memory Management Unit (MMU) in ARM processors translates virtual addresses used by software into physical addresses used by the memory system. It also provides memory protection, allowing for multitasking and the efficient use of memory resources. The MMU works with Translation Lookaside Buffers (TLBs) to speed up address translation and with page tables to map virtual addresses to physical addresses.

Memory management unit

Address Translation:

- The MMU translates virtual addresses (used by programs) into physical addresses (used by the memory system).
- This translation is done using page tables, where each entry in the table maps a virtual address range to a physical address range.
- The MMU uses TLBs (Translation Lookaside Buffers) to cache recent address translations, significantly speeding up the translation process.

Memory management unit

Memory Protection:

•The MMU provides memory protection by controlling access to different memory regions, ensuring that one process cannot interfere with the memory of another process.

•This protection is achieved through access permissions (read, write, execute) associated with each memory region.

Memory management unit

Key Components:

- **Table Walk Unit:**
- This unit reads translation tables from memory to determine the physical address corresponding to a given virtual address.
- **Translation Lookaside Buffers (TLBs):**
- These are caches that store recent address translations, allowing for faster access to physical addresses.
- **Page Tables:**
- These data structures contain the mappings between virtual and physical addresses.

Memory management unit

- **ARM Cortex-M3/M4:**

- These processors may use a memory map that allows built-in peripherals to be accessed using simple memory access instructions, says Mikroe.

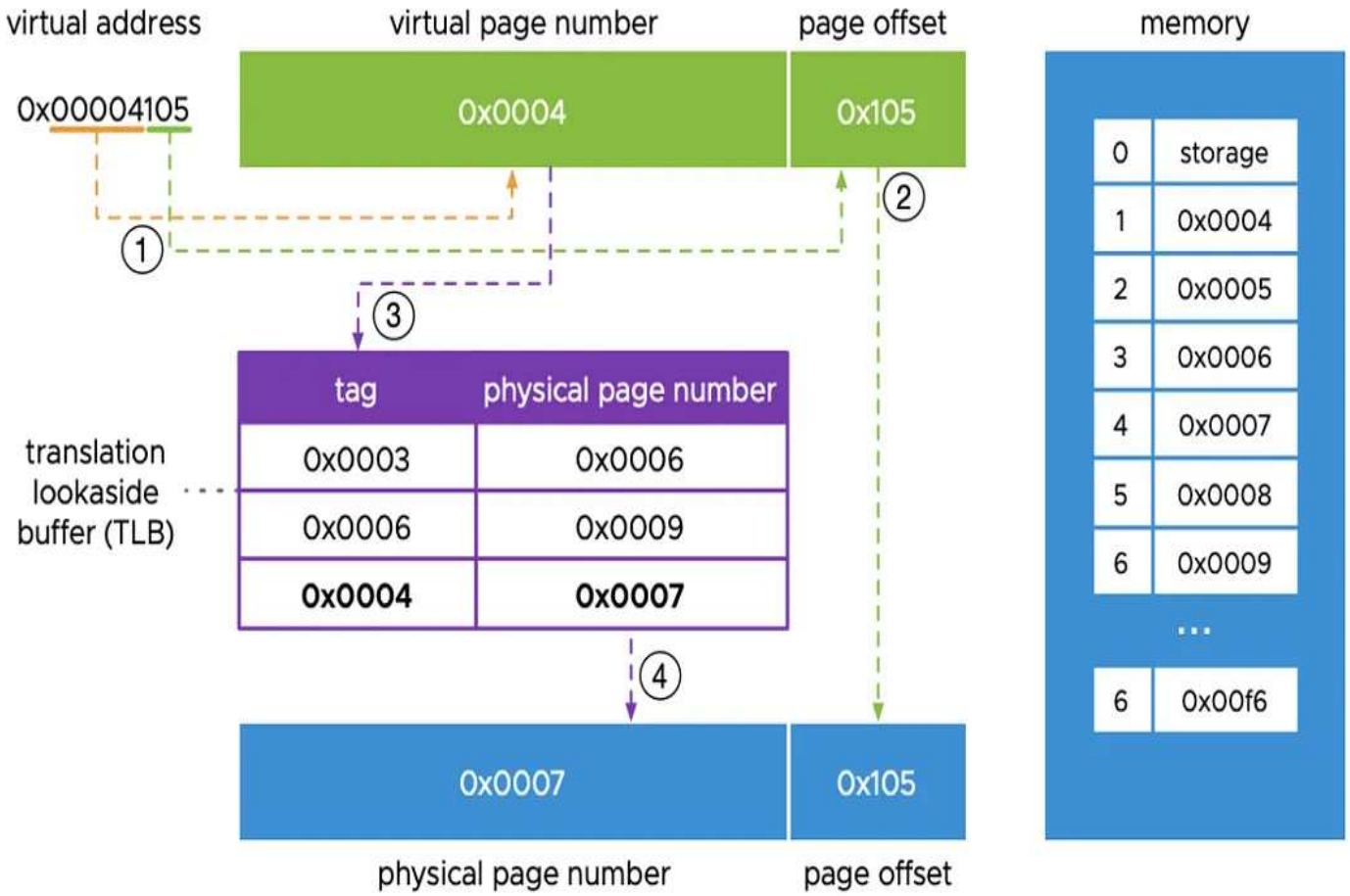
- **ARMv8-A architecture:**

- This architecture supports various page table sizes (4KB, 64KB, 1MB, 16MB) and uses a two-level page table for smaller page sizes.

Memory management unit

MMU or Memory management unit serves to virtualize address space used by running program and this allow to achieve certain level of multi-processing which will not possible without virtualizing the Physical address space, not only that it allows to provide memory space range that Internal Physical doesn't have which allow running as many program as external hard disk can store.

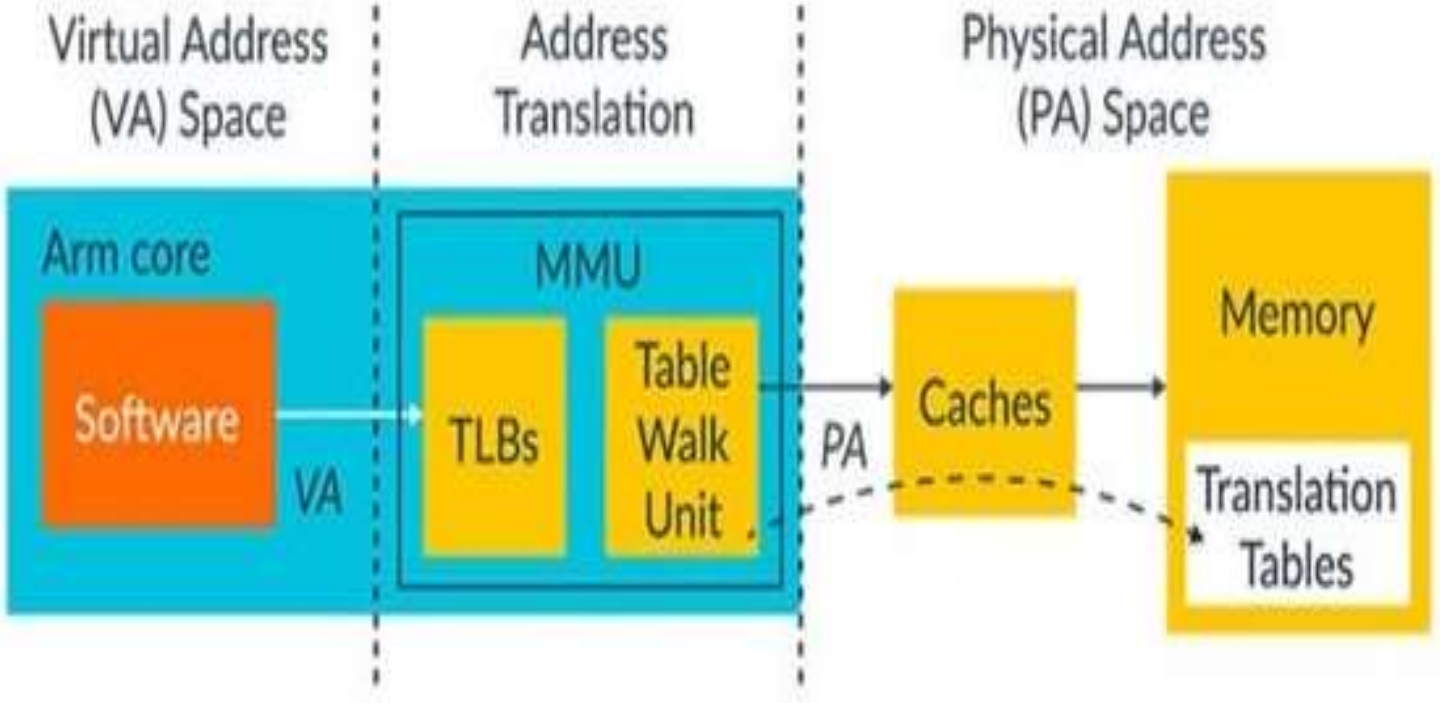
Memory management unit



Memory management unit

•Virtual to Physical Address: when it comes to addressing we have always 2 major issues: we want contiguous address space that can be used by our process and we want as such as we can to fit big number of running process, using only Physical address doesn't help here ad internal Physical address is limited in term of size and also we will have issue of external fragmentation which prevents allocating contiguous address space to single process , and the way to go to solve such issues is by virtualizing the address space used by the process. the Translation of Virtual address to Physical one is done through the MMU but MMU needs certain input and block before performing such translation

Memory management unit



Memory management unit

Above is ARM implementation of MMU, first we need to know that MMU will requires a translation table to be able to do the mapping between Virtual and Physical address, so each VA will use its upper bits to identify the entry within the translation table to pickup then what is left will be used as an offset to be appended to physical address returned by the translation table entry:

VA : 0x01000010 (32 bits address) -> bits[31:24] = 0x01 -> select entry in TTB (Translation Table)

TTB[1] = 0x80000000 (Physical address) MSB part

VA : 0x01000010 -> TTB -> 0x80000000 | 000010 = 0x80000010 (PA)

This is simplified example of course because we may have some extra settings associated with TTB entry and not only 32 bits base Physical Address available (read/write/execute restriction or Memory attribute access like normal/device/strongly ordered in arm)

Memory management unit

To speed up access to TTB a dedicated cache Memory has been added for such purpose which is called Translation Lock-Aside buffer (TLB) and first thing MMU will do is to check the TLB before doing Table Walk-in as this is costly to go all the way to physical memory to get TTB corresponding address.