Memory

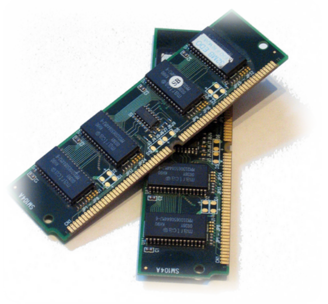
Module 6

Module Syllabus

- Types of memory
- Memory hierarchy
- Static RAM (SRAM)
- Dynamic RAM (DRAM)
- Memory management
- Translation lookaside buffers (TLBs)
- Direct memory access (DMA)

Motivation

- When a processor runs an application, it needs access to its instructions and data.
- These have to be placed in physical storage locations for the processor to easily access.
  - Instructions and data for one program also need isolating from other programs.
- Understanding their designs gives insight into their trade-offs.
- Understanding how they are used gives insight into how hardware can help.


Pair of DRAM modules[1]

DRAM[2]

Computer parts[3]

1. By Douglas Whitaker at English Wikipedia, CC BY-SA 2.5
2. By Petr Kratochvil, CC BY-0
3. By Clipartwiki.com

Volatile vs Non-volatile Memory

- Volatile memory
  - Requires power to retain the data information
  - Usually, faster access speed and less costly
  - Used for temporary data storage, such as CPU cache, internal memory
  - Also known as Random Access Memory (RAM)
- Non-volatile memory
  - No power is required to retain the data information
  - Usually, slower access speed and more costly
  - Used for secondary storage, or long-term persistent storage

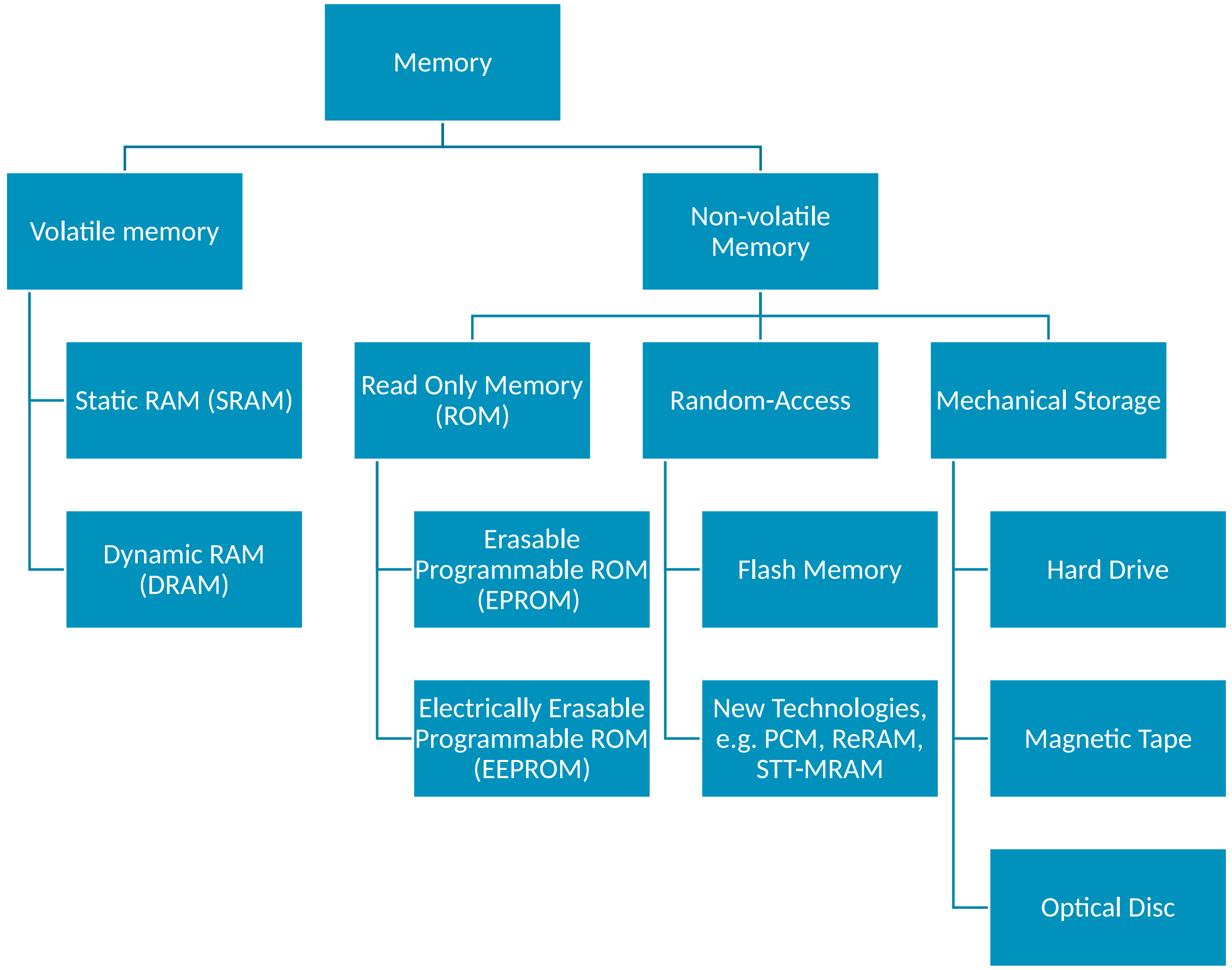
Types of Memory

Volatile memory

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Non-volatile memory

- Read only memory (ROM)
  - Erasable programmable ROM (EPROM)
  - Electrically erasable programmable ROM (EEPROM)
- Non-volatile random-access memory (NVRAM)
  - Flash memory
- Mechanical storage
  - Hard drive, magnetic tape

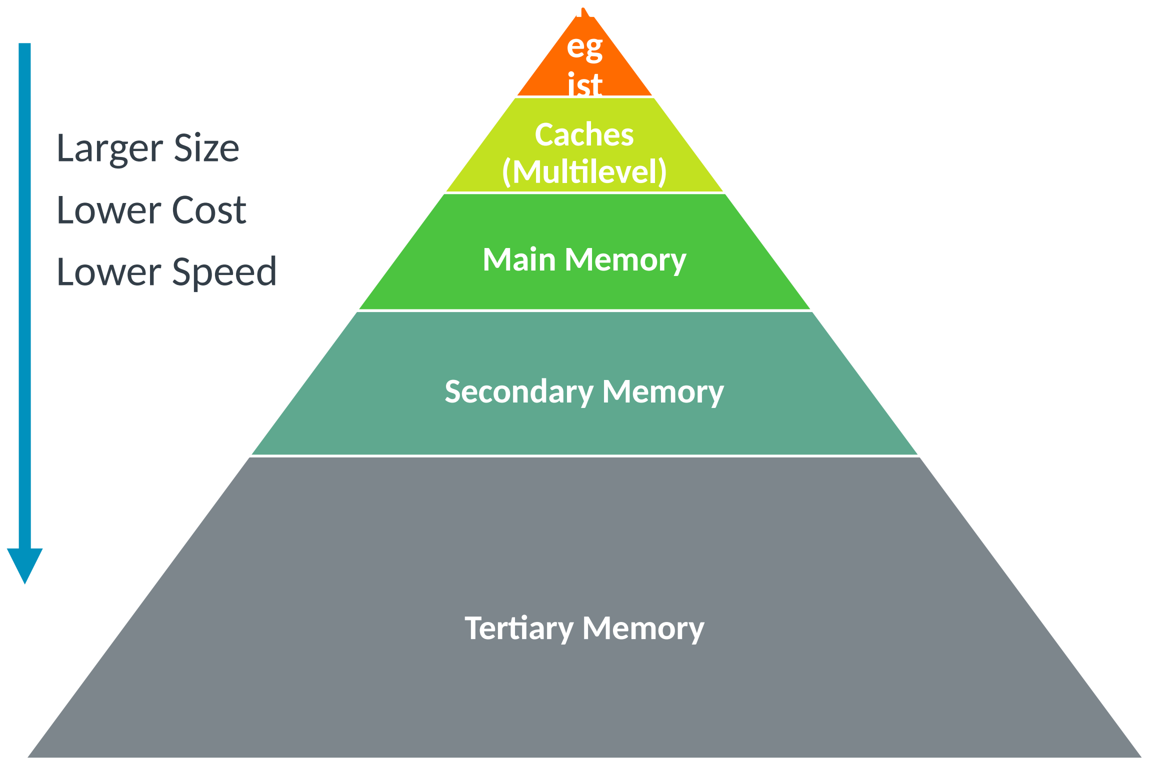


Memory Hierarchy

Memory Types

- Register: usually one CPU cycle to access
- Cache: CPU cache, translation lookaside buffer (TLB)
  - SRAM
- Main memory
  - DRAM
- Secondary memory: hard disk, solid-state drive
- Tertiary memory: tape libraries, cloud

Functional/Organizational View



Volatile Memory

- Although the system contains many types of memory, we'll focus on volatile memory.
  - SRAM, for caches
  - DRAM, for main memory
- These are the types of memory most commonly found in microprocessors.
  - And since they are close to the CPU, their interactions most commonly need to be considered by microarchitects.

Memory Technology Basics

SRAM vs DRAM

SRAM – Static Random Access Memory

- Static – holds data as long as power is maintained
- Requires multiple transistors to retain one bit and has low density compared to DRAM, thus more expensive
- Faster than DRAM
- Used for caches (next module)

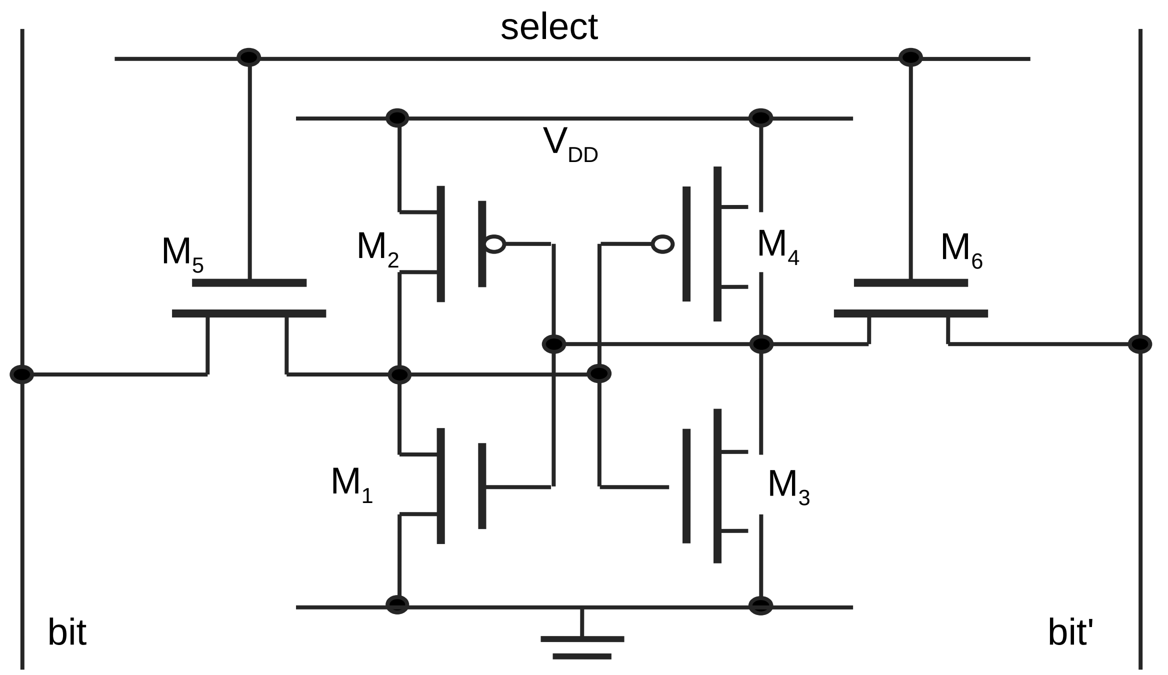
DRAM – Dynamic Random Access Memory

- Dynamic – must be refreshed periodically to hold data
- Requires only one transistor (and one capacitor) to retain one bit of data
- High density, thus cheaper than SRAM
- Used for main memory and sometimes for larger caches

SRAM

SRAM cell

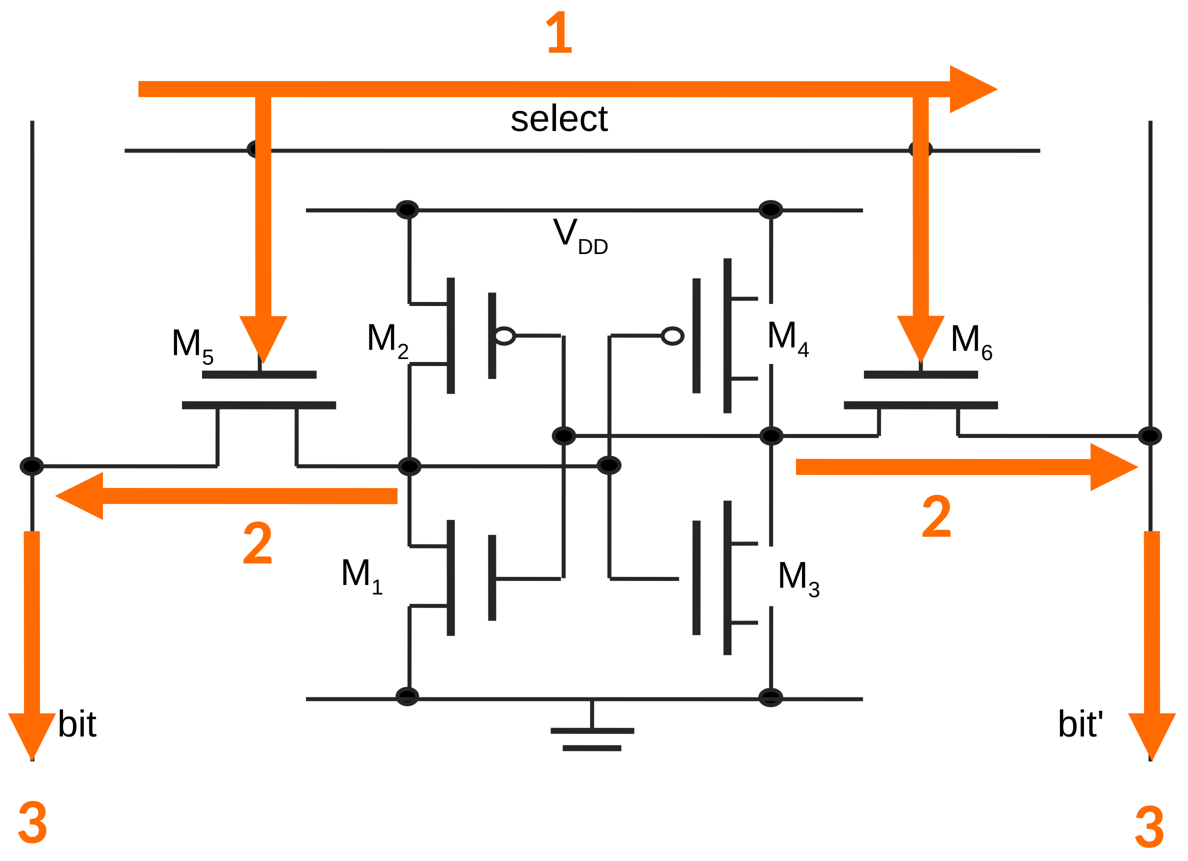
- An SRAM cell is typically made up of six transistors (MOSFETs).
  - A single bit is stored on 4 transistors (M1-M4), which form two inverters that are cross-coupled.
  - Access to the bit is controlled by two access transistors (M5 and M6), which are gated by the word line (select).
  - Data are read in and out through the bit lines.



Accessing SRAM

Read operation

- The address is decoded and the desired cell is then selected, in which case the select line is set to one.
- Depending on the value of the 4 transistors (M1-M4), one of the bit lines (bit or bit') will be charged to 1 and the other will be drained to 0.
- The states of the two bit lines are then read out as 1-bit data.



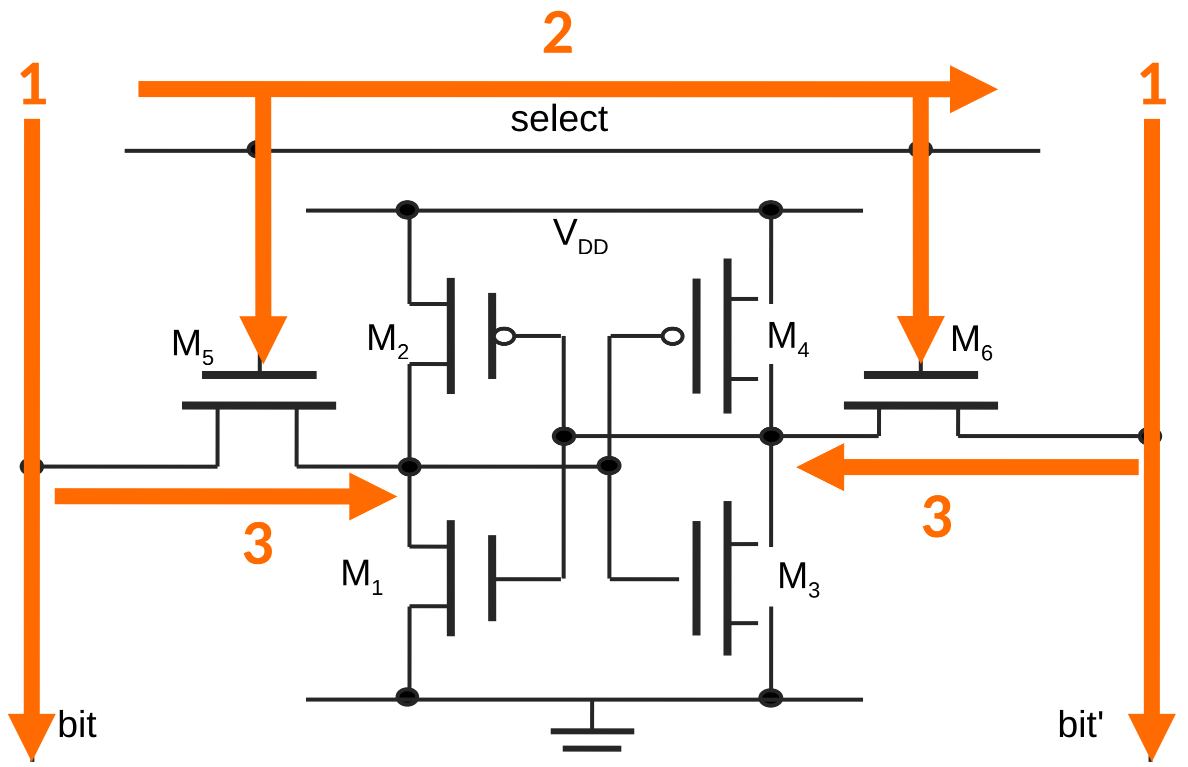
Accessing SRAM

Write operation

The two bit lines are pre-charged to the desired value (e.g., bit = VDD, bit’ = VSS).

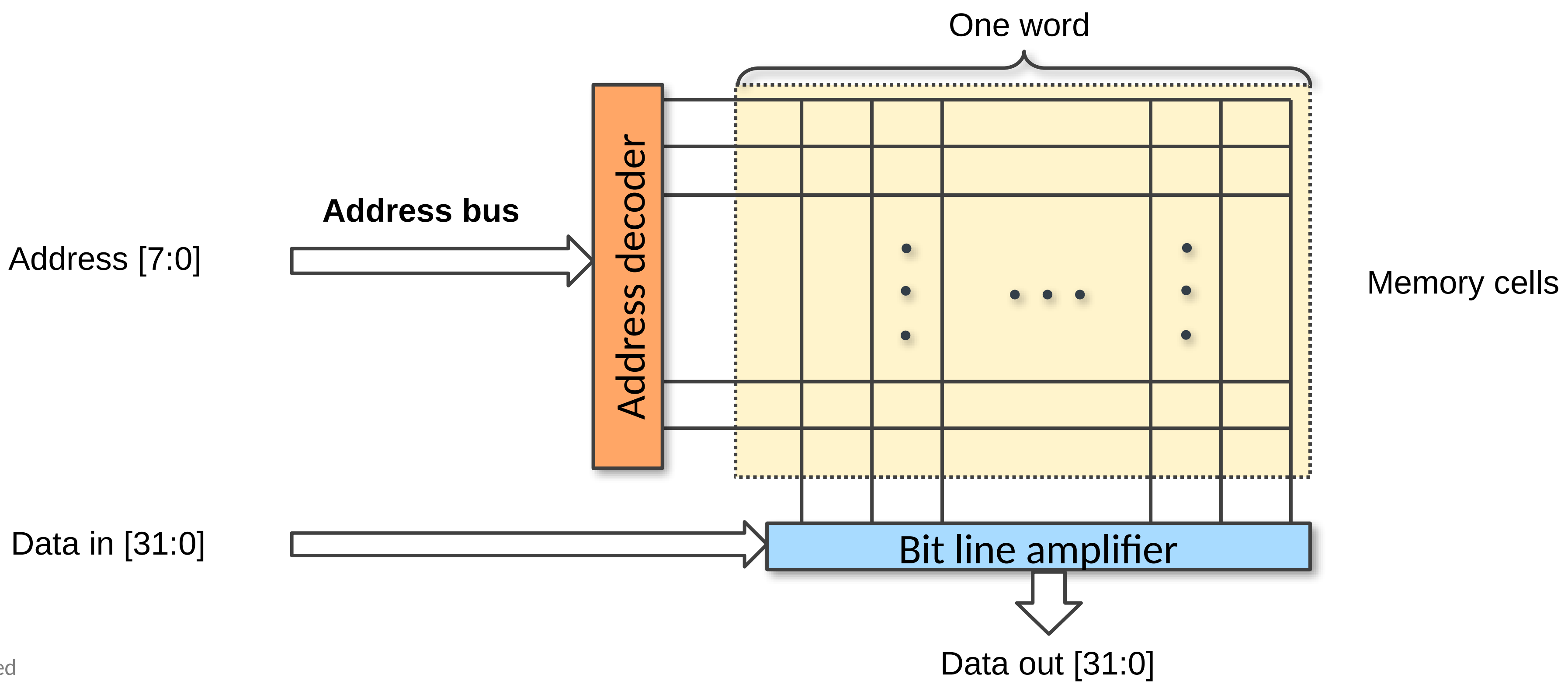
The address is decoded and the desired cell is then selected, in which case the select line is set to one.

The 4 transistors (M1-M4) are then forced to flip their states (either charged or discharged) since the bit lines normally have much higher capacitance than the 4 transistors.



Accessing SRAM

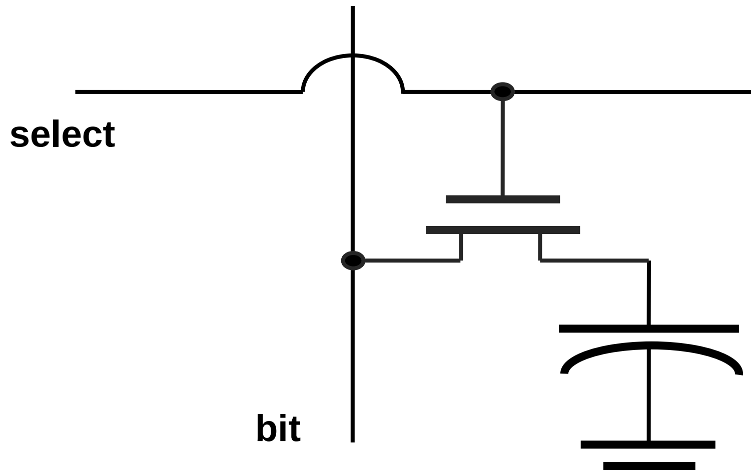
- The SRAM cells are organized into rows, with a whole row accessed at once.
  - For example, a memory architecture with an 8-bit address and 32-bit data is shown below.
- The address decoder uses the address to select a single row, and all its data are read out.



DRAM

DRAM

- A DRAM cell is typically made up of three or even one transistor.
  - A single bit is stored in one capacitor.
  - Access to the bit is controlled by a single access transistor, which is gated by the word line (select).
  - As in SRAM, data are read in and out through the bit line.

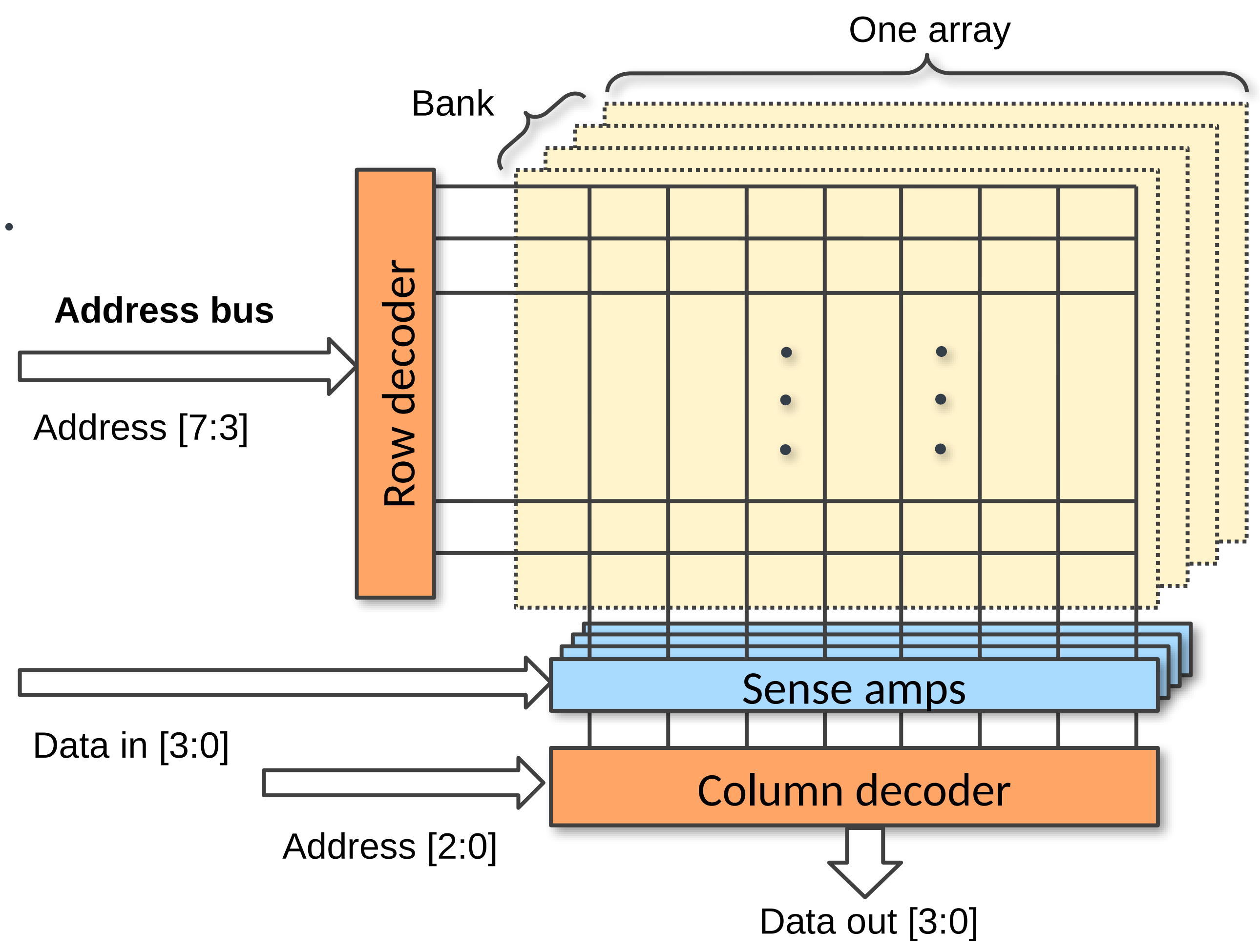


DRAM

- The status of the capacitor (charged or uncharged) indicates the bit state (1 or 0).
- Access is similar to SRAM but.
  - The capacitor is drained on a read and charged (if storing 1) on a write.
  - The cell needs to be refreshed (or recharged) periodically since the capacitor leaks its charge.
    - For example, every 7.8 ms
- DRAM is higher density than SRAM.
  - Therefore less expensive
- DRAM can be categorized according to its synchronization and data rate.
  - Most DRAM is now synchronous (SDRAM), so it has a clock, rather than asynchronous.
  - Double data rate (DDR) DRAM transfers data on both the rising and falling clock edges.

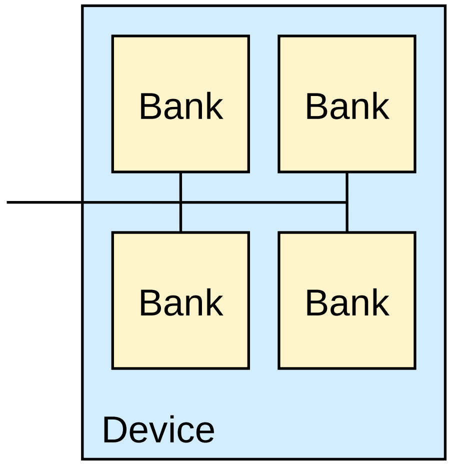
DRAM Organization

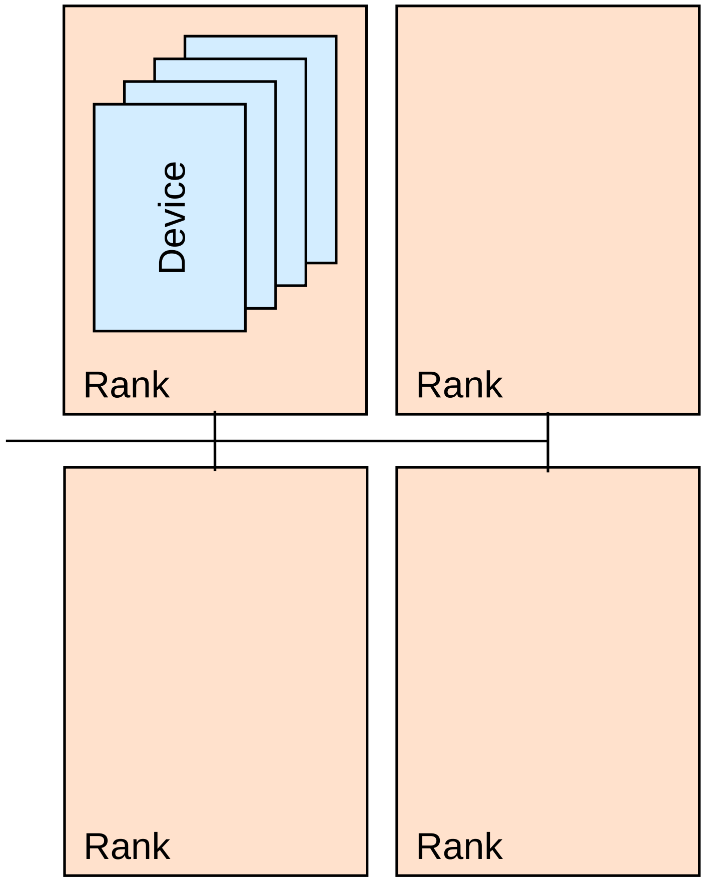
- DRAM cells are organized into arrays.
  - A whole row is accessed at once.
  - But only one bit is read out of the row.
- Multiple arrays are grouped into banks.
  - All arrays in a bank are accessed simultaneously.
  - A bank with N arrays provides N bits per access.



DRAM Organization

- DRAM banks are grouped into devices.
- Each device bank operates independently.
  - This allows multiple accesses to occur concurrently.
- Devices may be grouped into ranks.
  - All devices in a rank are accessed together.
  - This provides bandwidth.





Techniques for Improving DRAM Performance

Row buffer

- To buffer recently accessed data without having to make another access
- Read and refresh several words and in parallel.
- The row buffer is essentially the sense amps at the bottom of each array.

Double data rate

- Transfer data on the rising and falling clock edges to double the bandwidth.

DRAM banking

- Increase the number of parallel banks to improve bandwidth with simultaneous accesses.

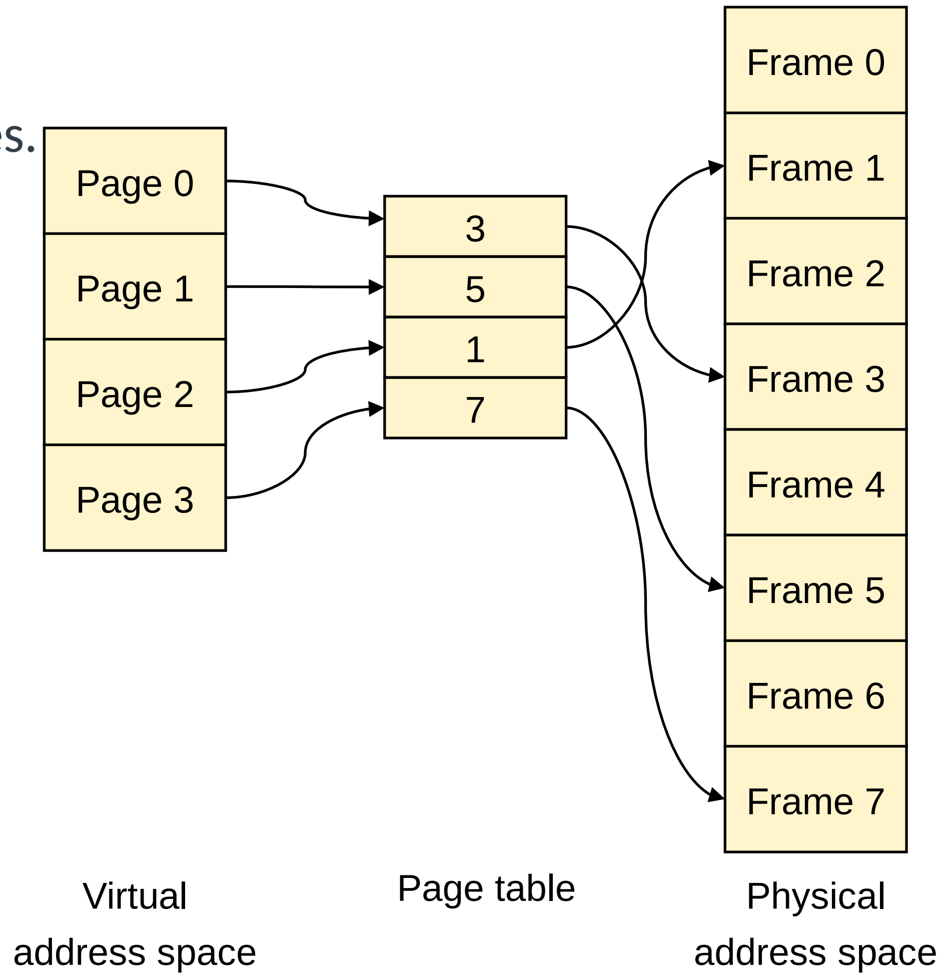
Memory Management

Motivation – Why Manage Memory?

- Applications’ instructions and data are stored in main memory.
  - And cached when required (see the next module)
- While this is fine for a single application, it poses a security risk with multiple programs.
  - An untrusted program could read sensitive data.
  - Or corrupt the state of another application, including taking control of it
- We provide an operating system to prevent this happening.
- What hardware support can we provide to improve the performance of the OS?

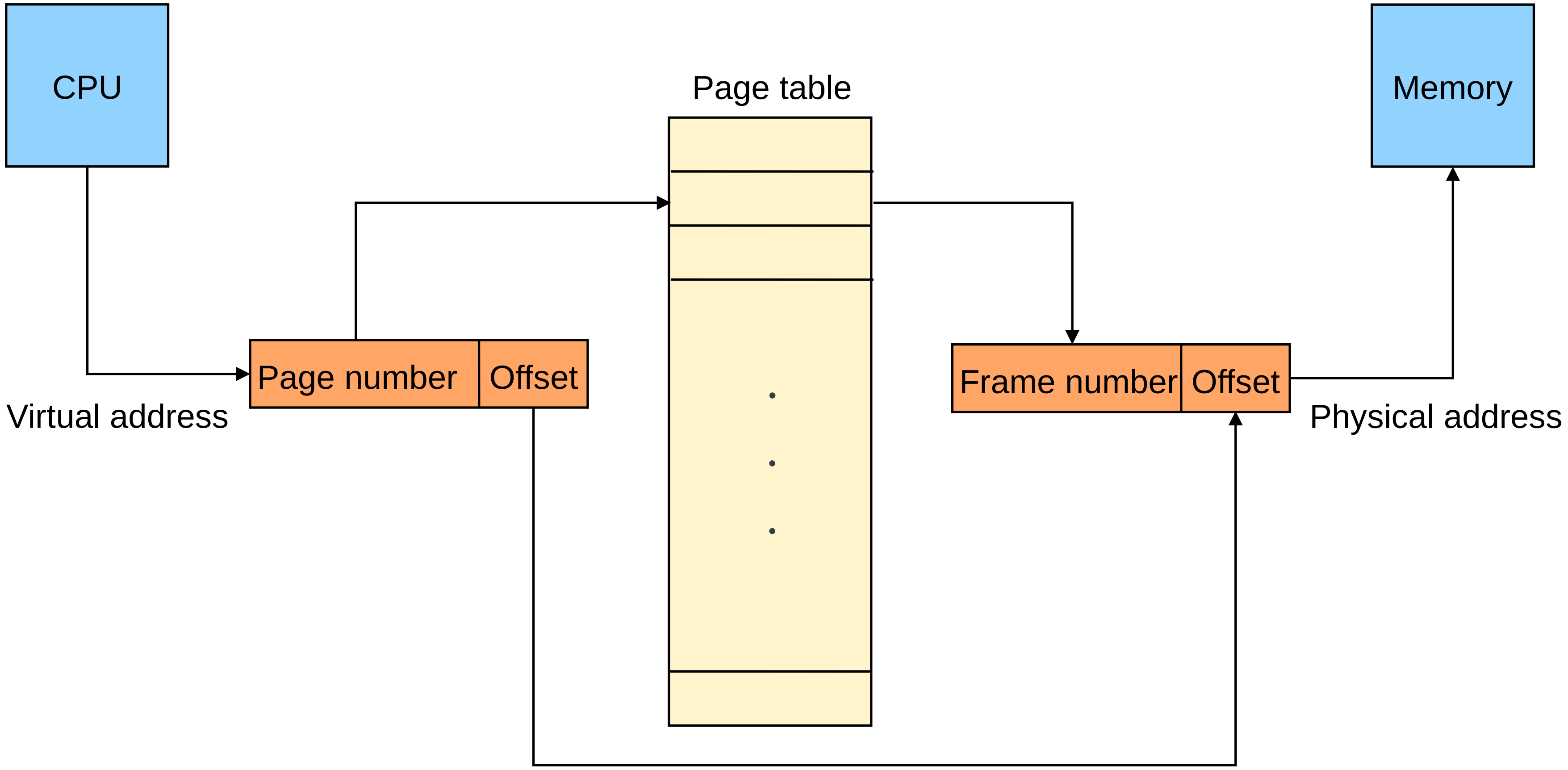
Paging

- The OS forbids direct access to memory.
- Instead, it introduces a layer of indirection.
  - Applications see memory as a range of virtual addresses.
  - The OS maps these to physical addresses.
- Paging is one method to achieve this.
  - Physical memory is split into fixed-sized frames.
  - Virtual memory is split into same-sized pages.
  - Each page is mapped to one frame.
    - Using the high-order bits from the address
    - This information is kept in the page table.



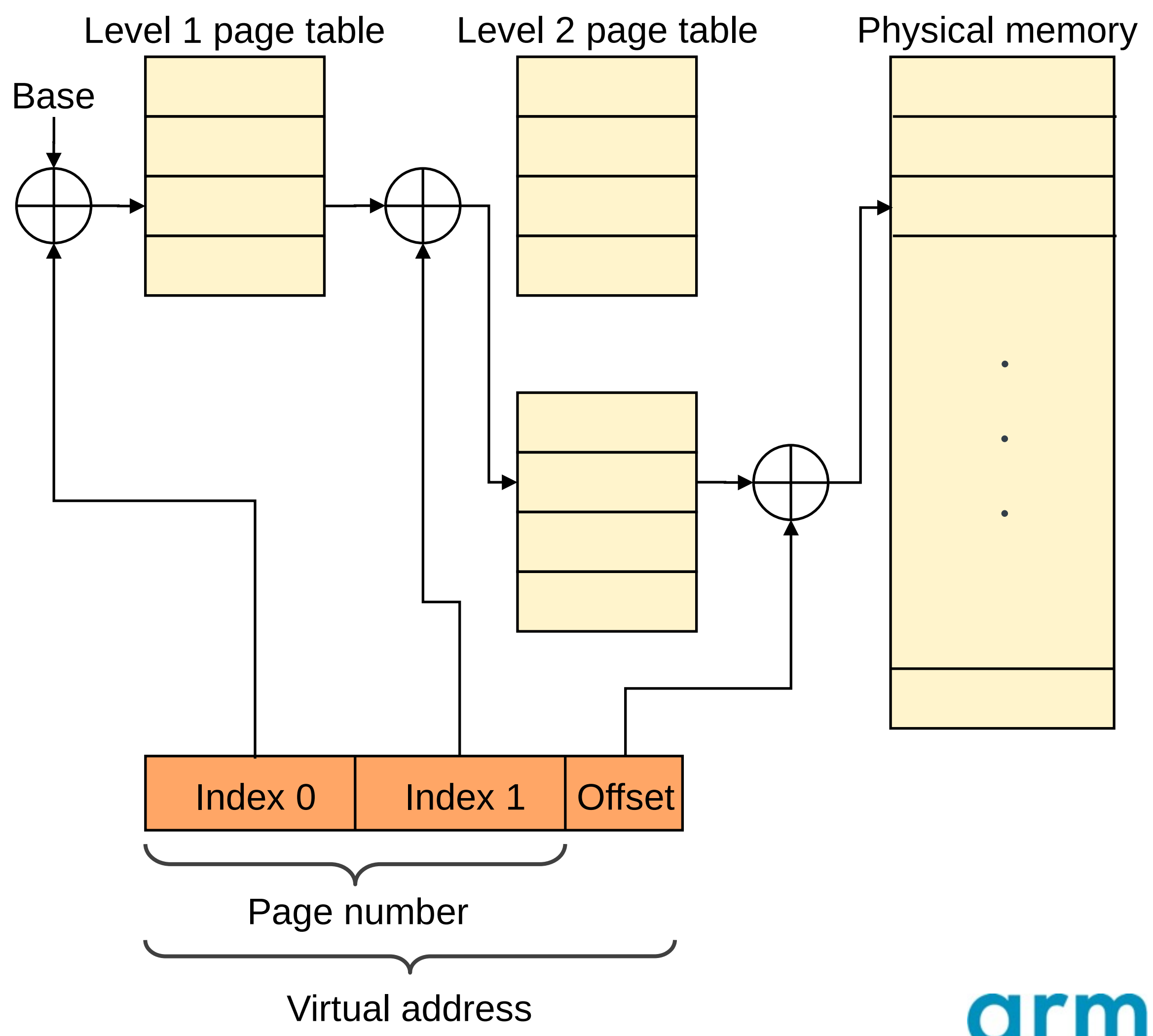
Page Address Translation

- The CPU works on virtual addresses.
  - To access physical memory, the virtual address has to be converted to a physical address by indexing into the page table.
- However, this makes the page table extremely large.



Multi-level Page Tables

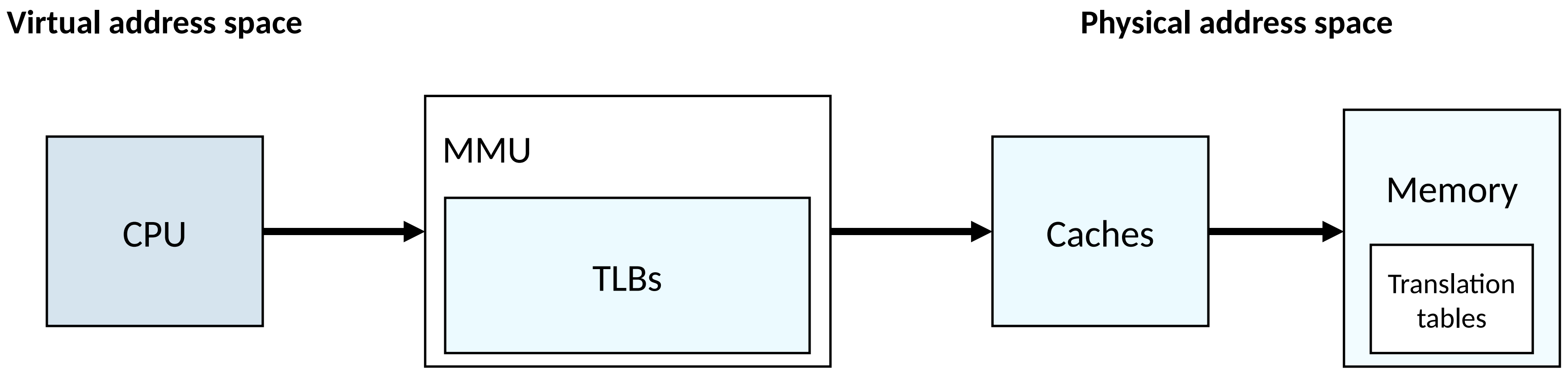
- To prevent the need for a large page table, we can create a tree of page tables.
  - Paging the page table
  - First-level entries point to second-level tables.
  - Second-level entries point to third-level tables, etc.
  - Final-level entries point to memory frames.
- To access a frame, we must walk through the page tables using the virtual address.
  - This is extremely costly.
- Can we provide hardware to help?



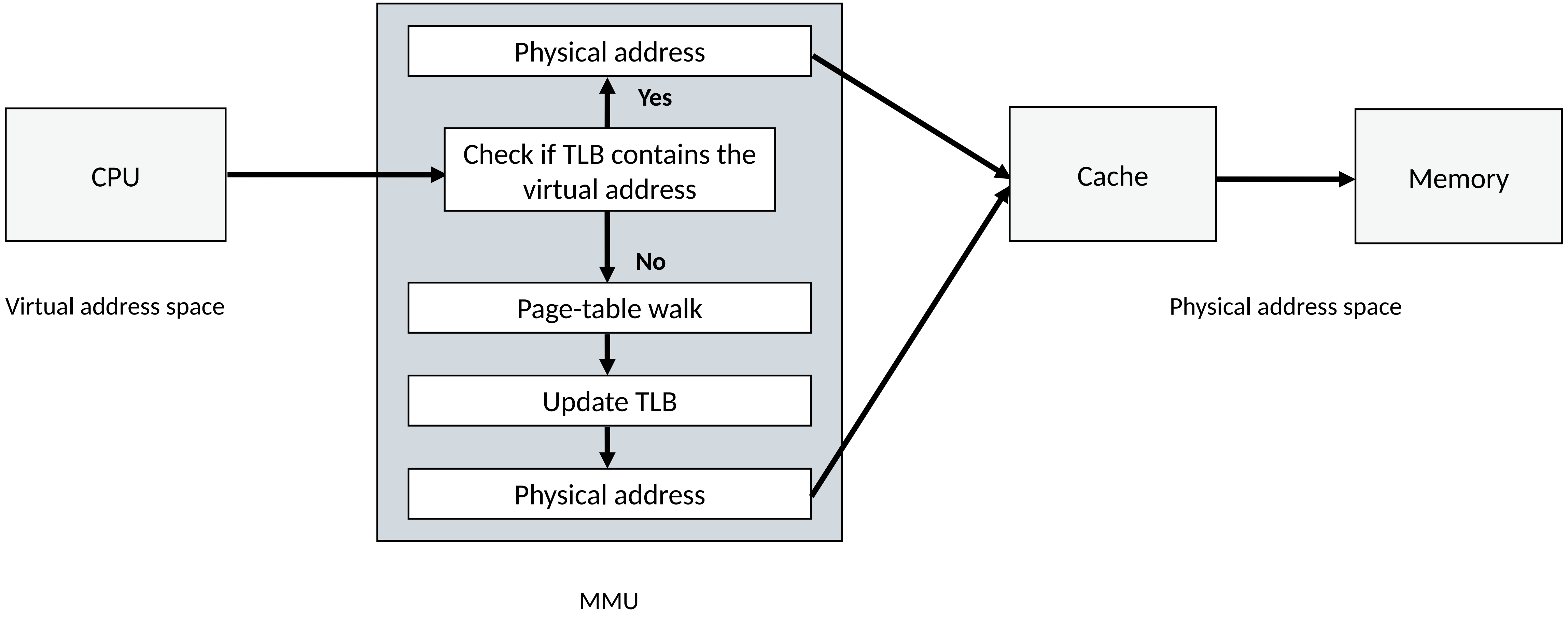
Memory-management Unit (MMU)

Handles translation of virtual addresses to physical addresses

- The MMU provides hardware to read translation tables in memory.
- The translation lookaside buffers (TLBs) cache recent translations.



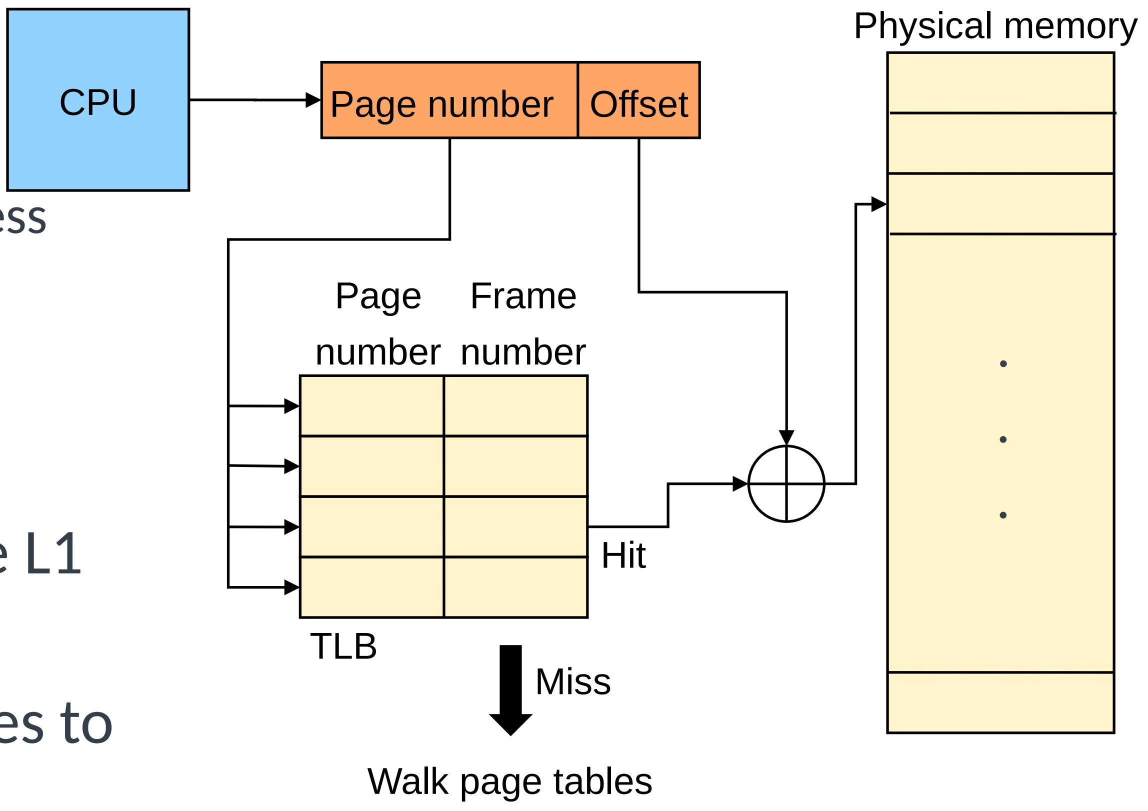
Overview of Memory Access Using an MMU



TLB

Translation lookaside buffer

- The TLB is a cache of page translations.
  - Provides access to recent virtual to physical address mappings quickly
- Each block is one or two page-table entries.
- TLBs are usually fully associative.
- On a hit, forward the physical address to the L1 cache.
- On a miss, the MMU will walk the page tables to find the translation.

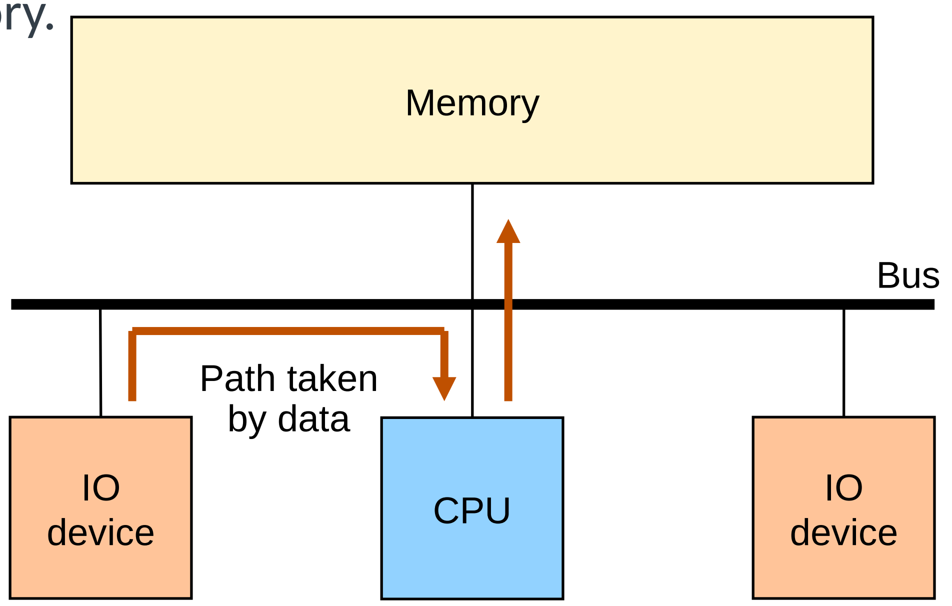


DMA

DMA

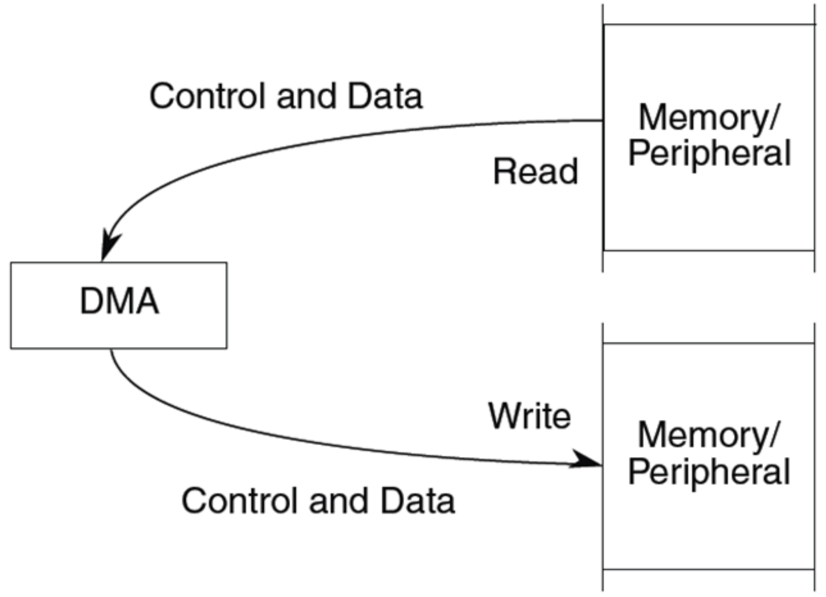
Direct memory access

- When an application wants to access IO, it does so through the OS.
  - Using a system call to read from or write to the device
- This requires the CPU to be involved in transferring data between the device and memory.
  - On a read, data are then available to the program in memory.
- This data transfer is costly and wasteful.
  - CPU can’t do any useful work whilst waiting.
  - Data travel further and take longer to transmit.

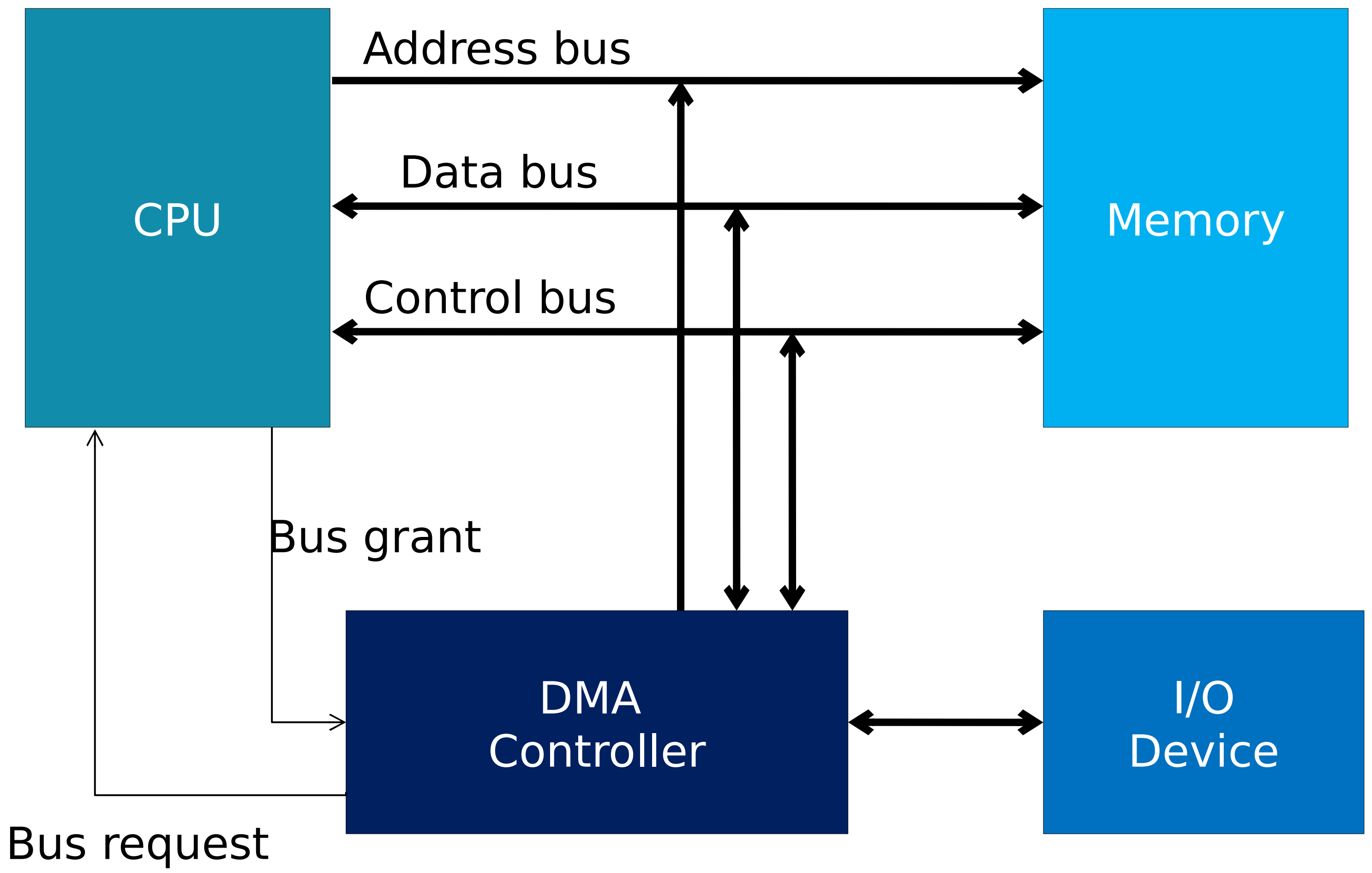


DMA

- Instead of involving the CPU, provide dedicated hardware to control the transfer.
  - A controller for direct memory access, or DMA.
  - CPU now just required to configure this DMA controller correctly.
- Typically supports multiple configurable options:
  - Number of data items to copy
  - Source and destination addresses
    - Fixed or changeable (e.g., increment and decrement)
  - Size of data item
  - Timing of transfer start
- A DMA controller can also work with interrupts, e.g., interrupt CPU at the end of a transfer.
- The main idea is to exempt the CPU from busy-waiting and frequent interruptions.



DMA Architecture



DMA Transfer Modes

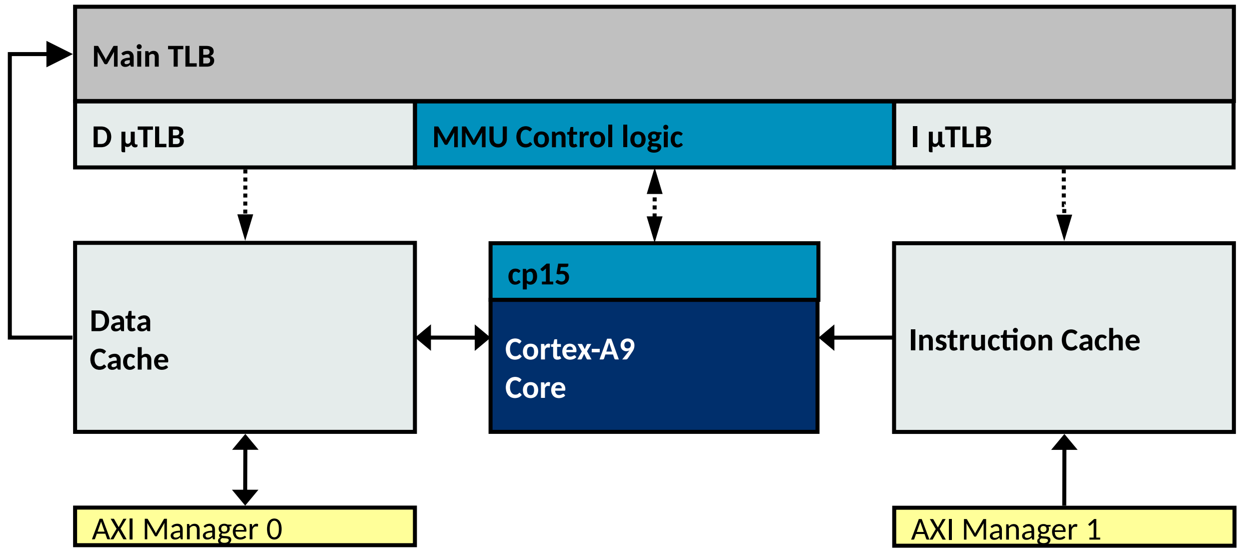
- Burst
  - An entire block of data is transferred in one contiguous sequence.
  - The CPU remains inactive for relatively long periods of time (until the whole transfer is completed).
- Cycle stealing
  - DMA transfers one byte of data and then releases the bus returning control to the CPU.
  - Continually issues requests, transferring one byte per request, until it has transferred the entire block of data
  - It takes longer to transfer data/the CPU is blocked for less time.
- Transparent
  - DMA transfers data when the CPU is performing operations that do not use the system buses.

Case Study: Cortex-A9 MMU

- The MMU in the Cortex-A9
  - Works with the L1 and L2 caches for virtual-to-physical address translation
  - Controls access to and from external memory
  - Is based on the Virtual Memory System Architecture from the Armv7-A architecture
  - Checks access permissions and memory attributes
  - Checks the virtual address (VA) and address space identifier (ASID)

Case Study: Cortex-A9 MMU

- Main TLB along with separate micro-TLBs for instructions and data for quick access
- Page-table walks can be configured to go through the L1 data cache.
  - Allows page tables to be cached



Case Study: Cortex-A9 MMU

- Memory access sequence

1. MMU performs a lookup for the requested VA and current ASID in the relevant micro-TLB.
2. If there is a miss in the micro-TLB, look in the main TLB.
3. If there is a miss in the main TLB, the MMU performs a hardware page-table walk.

- If the MMU finds a matching TLB entry, the MMU

1. Does permission checks; if these fail, the MMU signals a memory abort
2. Determines if the access is secure/non-secure, shared/non-shareable, and memory attributes
3. Performs translation for the memory access

Conclusions

- SRAM is expensive but fast and typically used for on-chip caches.
- DRAM is slower but denser and cheaper, typically used for main memory.
- DRAM cells grouped into arrays, banks, devices, and ranks.
  - Mixture of synchronized operation and concurrency at different levels gives bandwidth and parallel access.
- Translating from the application’s view of memory to physical addresses is costly.
  - The memory-management unit, in particular the TLB, helps.
  - As do hardware page-table walkers
- DMA controllers free up the CPU from dealing with transfers between IO and memory.