Virtual Memory

Tabish Syed
COMP 273, Winter 2020

Virtual and Physical Memory

Virtual Memory

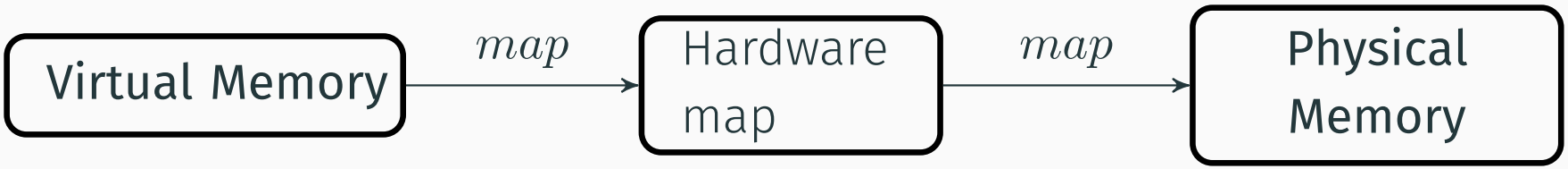
- Each *process* operates in its own virtual space, as the only program running.

Virtual and Physical Memory

Virtual Memory

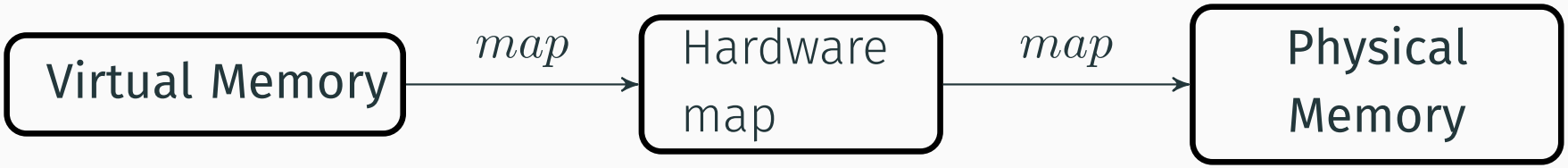
- Each *process* operates in its own virtual space, as the only program running.
- Memory Translation Hardware Translates virtual address to physical address.

Virtual and Physical Memory



- Each *process* operates in its own virtual space, as the only program running.
- Memory Translation Hardware Translates virtual address to physical address.
- Each *process* is protected from other running processes.

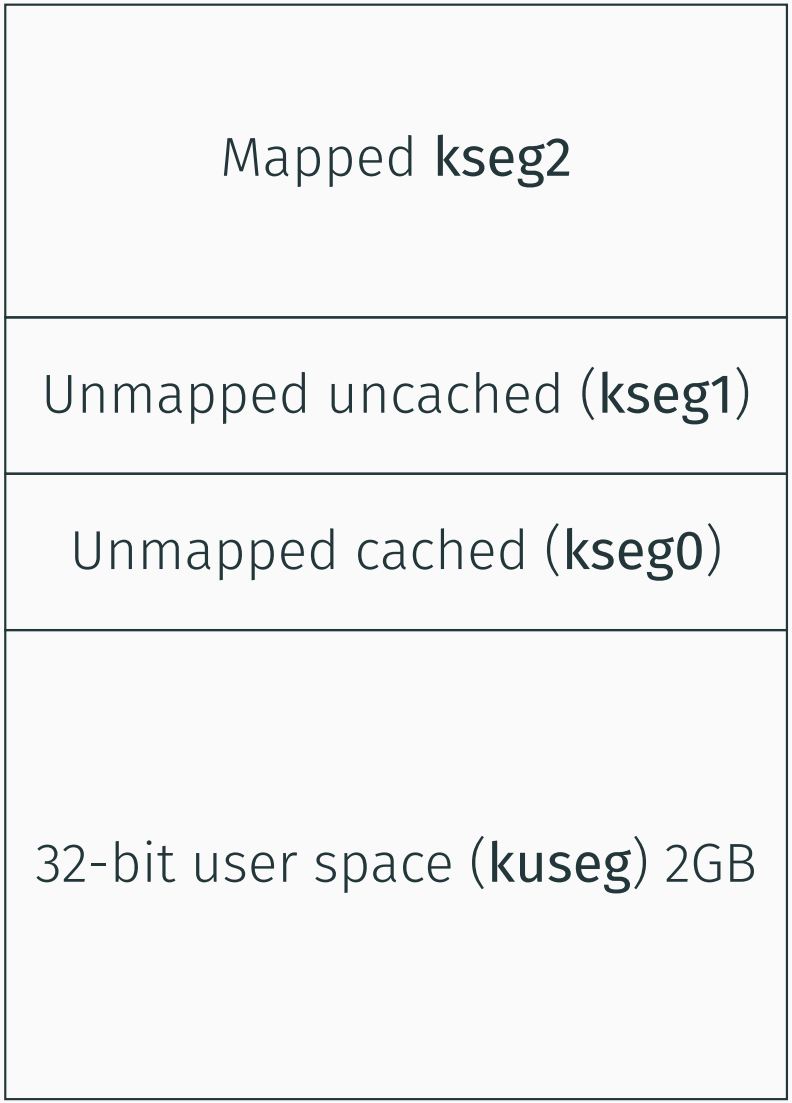
Virtual and Physical Memory



- Each *process* operates in its own virtual space, as the only program running.
- Memory Translation Hardware Translates virtual address to physical address.
- Each *process* is protected from other running processes.
- OS(software) can decide where each process goes in the physical memory.

Virtual Memory Layout

MIPS Virtual Memory Map



32 Bit MIPS memory map.

MIPS Virtual Memory Map

- kuseg: (`0x000 000 - 0x7fff ffff`) User segment is the lower 2 GB of the 32-bit address space. These addresses are allowed only in user mode. Mapped to physical memory by (Memory Management Hardware/Unit)MMU

MIPS Virtual Memory Map

- **kuseg**: (`0x000 000 - 0x7fff ffff`) User segment is the lower 2 GB of the 32-bit address space. These addresses are allowed only in user mode. Mapped to physical memory by (Memory Management Hardware/Unit)MMU
- **kseg0**: (`0x8000 000 - 0x9fff fff`) This 512 MB chunk are translated to the lowest 512 MB of physical memory (by setting MSB to 0). Used for OS kernel and access is through cache.

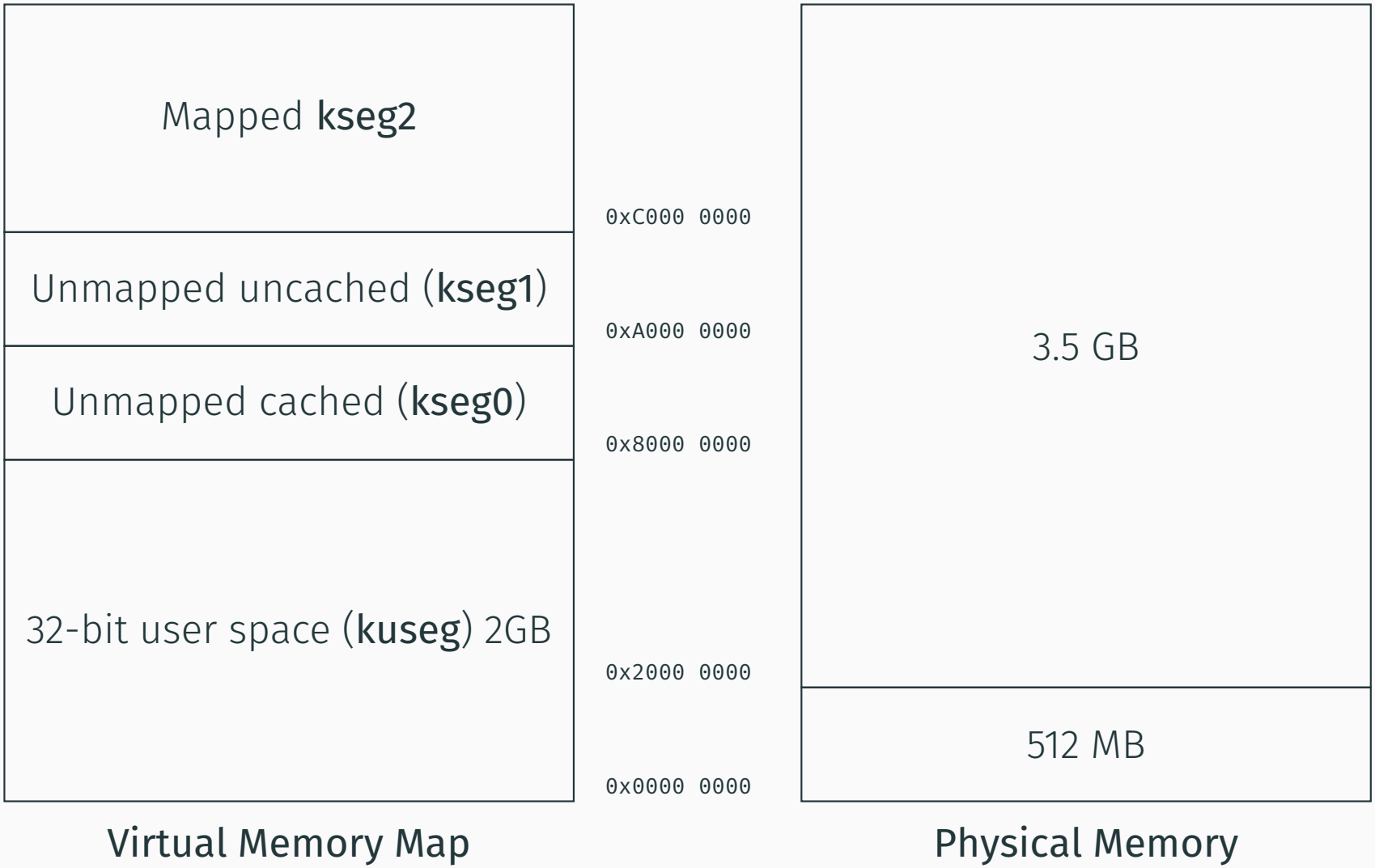
MIPS Virtual Memory Map

- **kuseg**: (`0x000 000 - 0x7fff ffff`) User segment is the lower 2 GB of the 32-bit address space. These addresses are allowed only in user mode. Mapped to physical memory by (Memory Management Hardware/Unit)MMU
- **kseg0**: (`0x8000 000 - 0x9fff fff`) This 512 MB chunk are translated to the lowest 512 MB of physical memory (by setting MSB to 0). Used for OS kernel and access is through cache.
- **kseg1**: (`0xa000 000 - 0xbfff fff` Duplicate mapping of lower 512 MB of physical memory. Mapped to physical memory by setting 3 MSBs to 0. This region is not cached and therefore behaves correctly at startup. Initial ROM is placed here.)

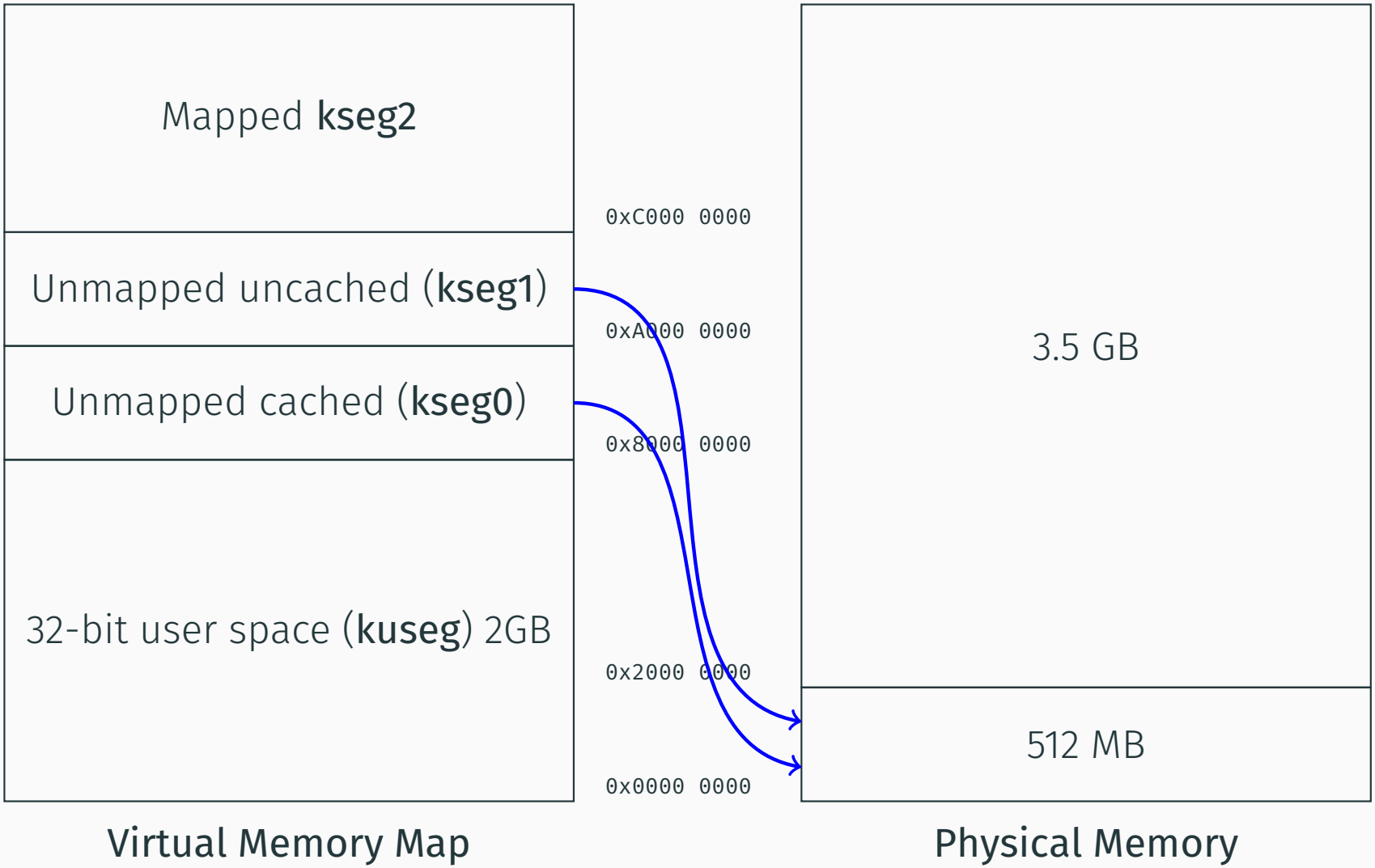
MIPS Virtual Memory Map

- **kuseg**: (`0x000 000 - 0x7fff ffff`) User segment is the lower 2 GB of the 32-bit address space. These addresses are allowed only in user mode. Mapped to physical memory by (Memory Management Hardware/Unit)MMU
- **kseg0**: (`0x8000 000 - 0x9fff fff`) This 512 MB chunk are translated to the lowest 512 MB of physical memory (by setting MSB to 0). Used for OS kernel and access is through cache.
- **kseg1**: (`0xa000 000 - 0xbfff fff` Duplicate mapping of lower 512 MB of physical memory. Mapped to physical memory by setting 3 MSBs to 0. This region is not cached and therefore behaves correctly at startup. Initial ROM is placed here.)
- **kseg2**: (`0xc000 000 - 0xffff ffff`) This 1 GB is only accessible in kernel mode. Mapped to physical memory by MMU

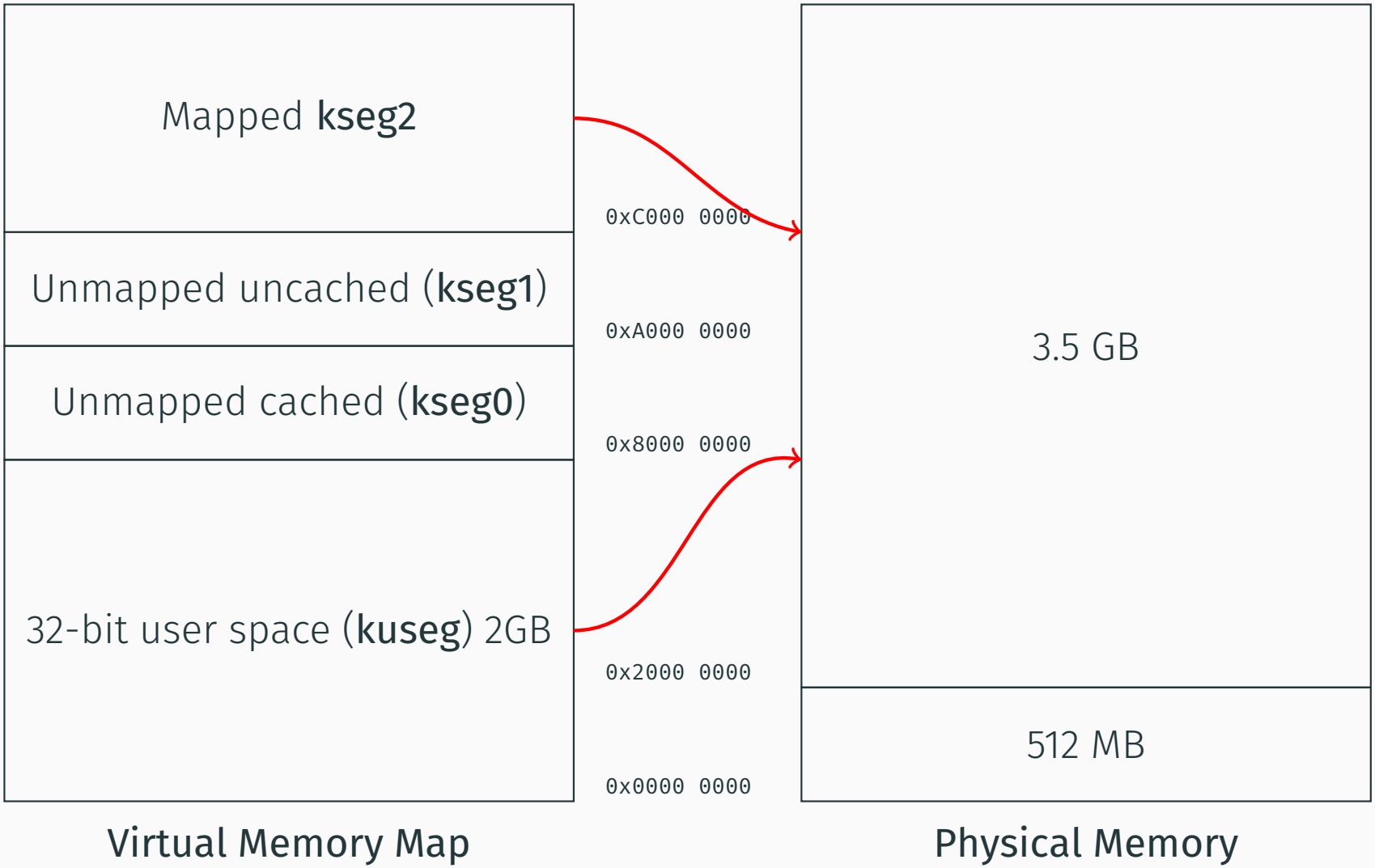
MIPS Virtual Memory Map



MIPS Virtual Memory Map



MIPS Virtual Memory Map



Kernel and User Segments

Privilege level of a program

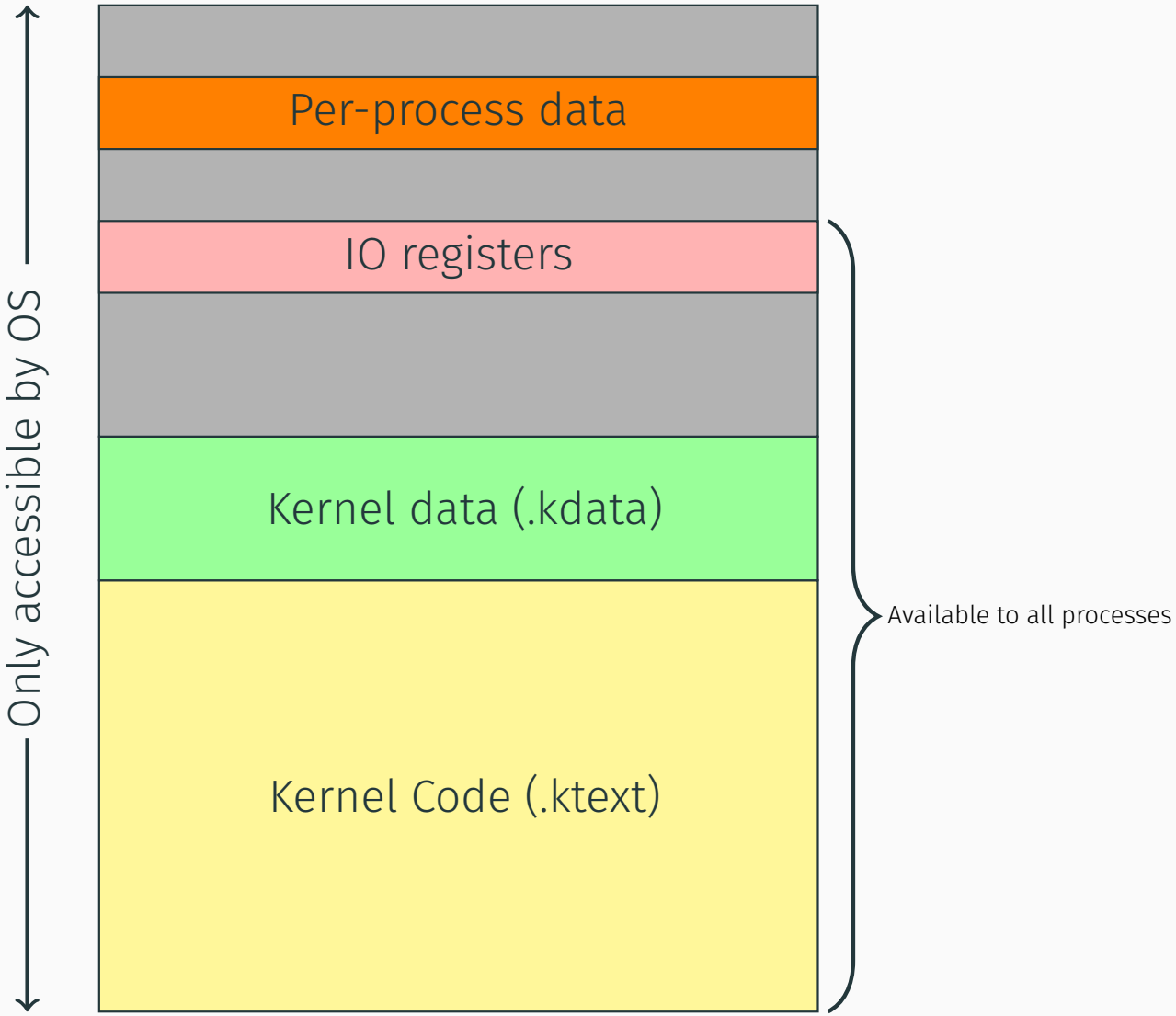
- **User Privilege:** In User mode program addresses above 2GB ($MSB = 1$) are illegal. Additionally, some instructions, like some CPU control instructions, are illegal.

Kernel and User Segments

Privilege level of a program

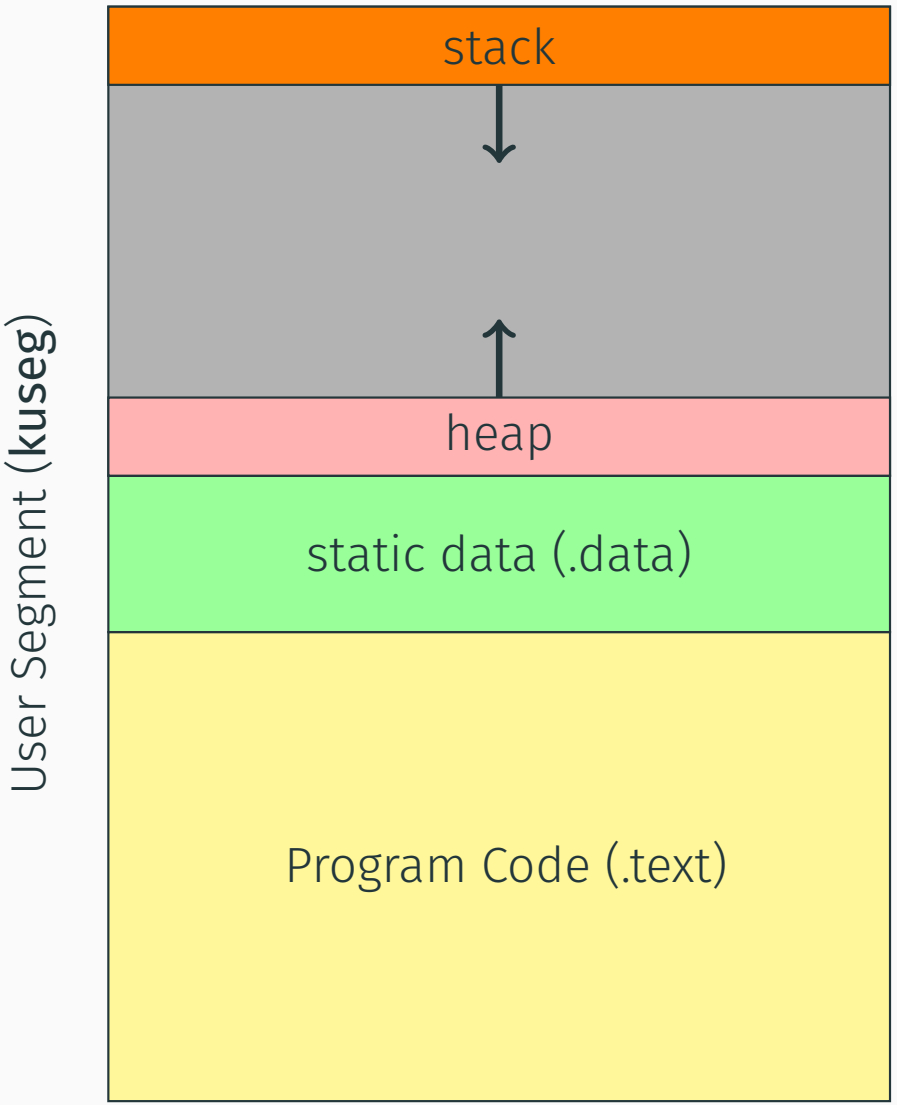
- **User Privilege:** In User mode program addresses above 2GB ($MSB = 1$) are illegal. Additionally, some instructions, like some CPU control instructions, are illegal.
- **Kernel Privilege:** In this mode a program can do anything. Generally OS runs in this mode.

Kernel and User Segments



Kernel Segment.

Kernel and User Segments



User Segment.

Mapping To Physical Memory

Virtual to Physical Address Mapping

- Mapping Virtual program addresses is done by OS when program is loaded.

Virtual to Physical Address Mapping

- Mapping Virtual program addresses is done by OS when program is loaded.
- Mapping also allows OS to discriminate and therefore protect different parts of memory.

Virtual to Physical Address Mapping

- Mapping Virtual program addresses is done by OS when program is loaded.
- Mapping also allows OS to discriminate and therefore protect different parts of memory.
- The Kernel part of a processes address space is shared by all processes and therefore maps to fixed location.

Hardware Memory Translation

Memory Translation hardware should achieve the following:

- **Relocation**: The Memory translation system allows a program to run anywhere in physical memory.

Hardware Memory Translation

Memory Translation hardware should achieve the following:

- **Relocation**: The Memory translation system allows a program to run anywhere in physical memory.
- **Protection**: Programs with user level privilege can only access the *kuseg* segment. Regions of memory can be write protected.

Hardware Memory Translation

Memory Translation hardware should achieve the following:

- **Relocation**: The Memory translation system allows a program to run anywhere in physical memory.
- **Protection**: Programs with user level privilege can only access the *kuseg* segment. Regions of memory can be write protected.
- **Demand Paging**: Programs can run as if all the memory resources they needed were already allocated, but OS actually allocates the space when needed.

Paged Mapping

- Mapping is generally done in **pages** of 4KB.

Paged Mapping

- Mapping is generally done in **pages** of 4KB.
- Mapping translates **virtual page number** (VPN) into **physical *frame* number** (PFN).

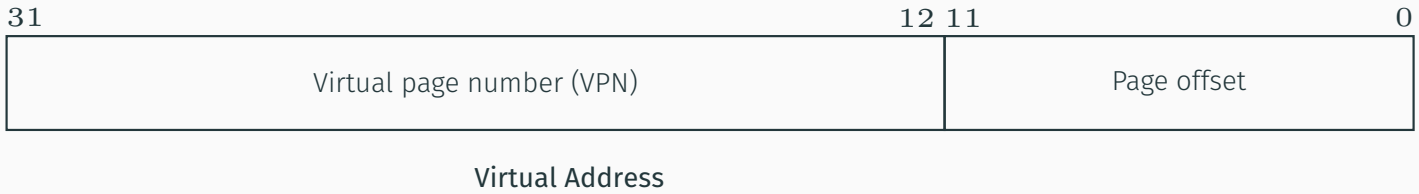
Paged Mapping

- Mapping is generally done in **pages** of 4KB.
- Mapping translates **virtual page number** (VPN) into **physical *frame* number** (PFN).
- To allow for efficient mapping we maintain a **page table** containing entries for each page.

Paged Mapping

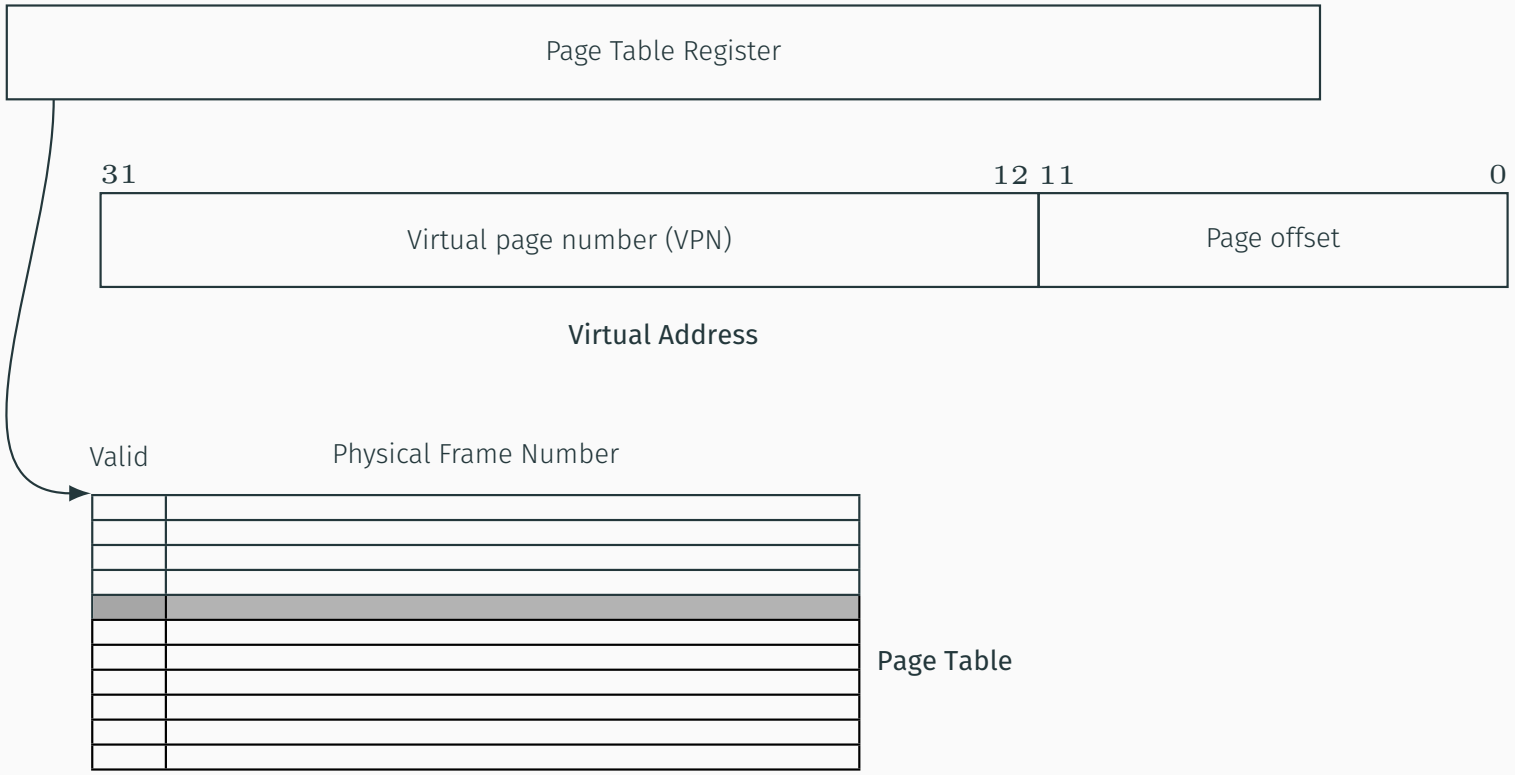
- Mapping is generally done in **pages** of 4KB.
- Mapping translates **virtual page number** (VPN) into **physical *frame* number** (PFN).
- To allow for efficient mapping we maintain a **page table** containing entries for each page.
- To improve translation speed (part of) the page table resides in a fast cache called **Translation lookaside buffer** (TLB) (Which part?).

Paged Mapping



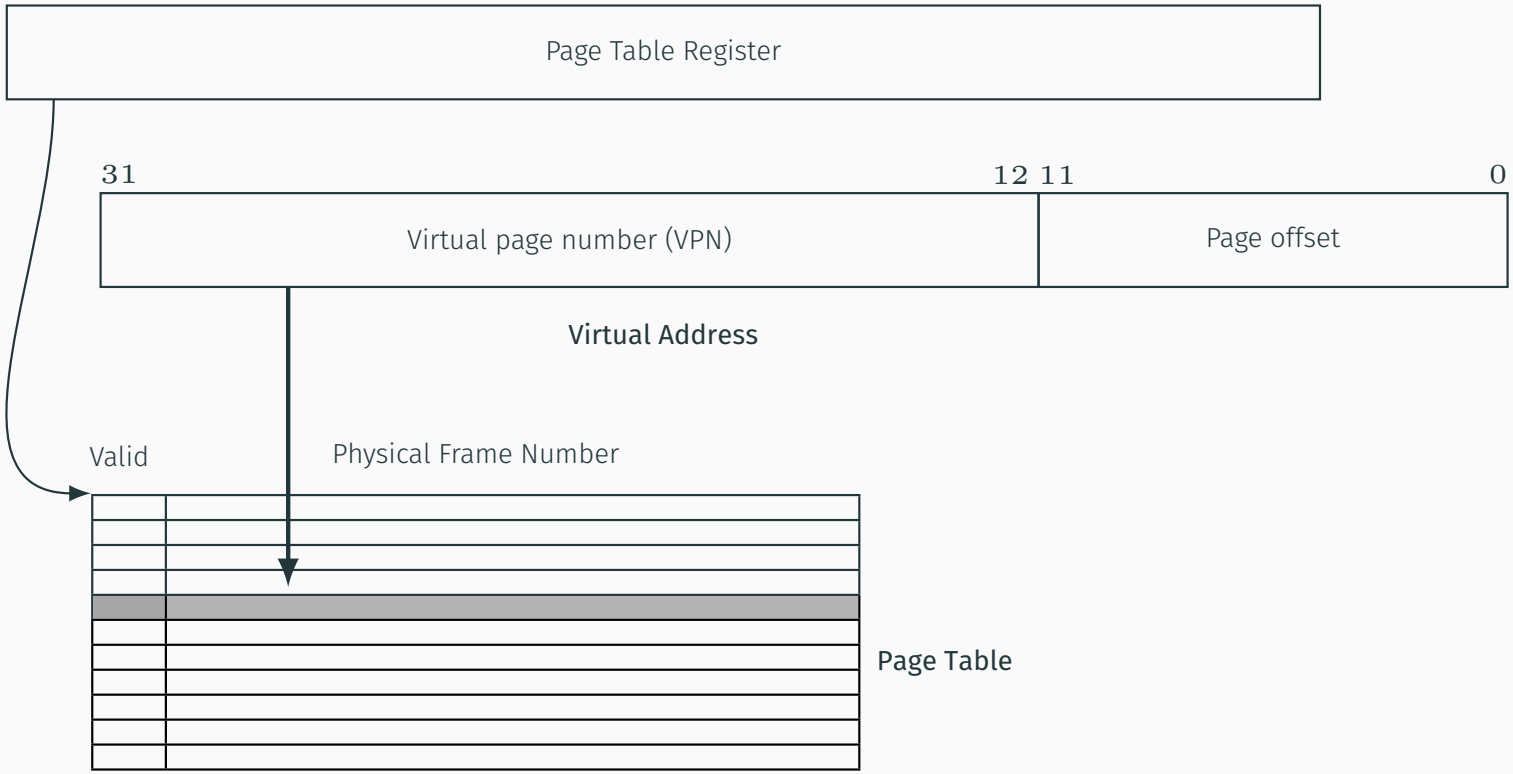
Virtual Address

Paged Mapping



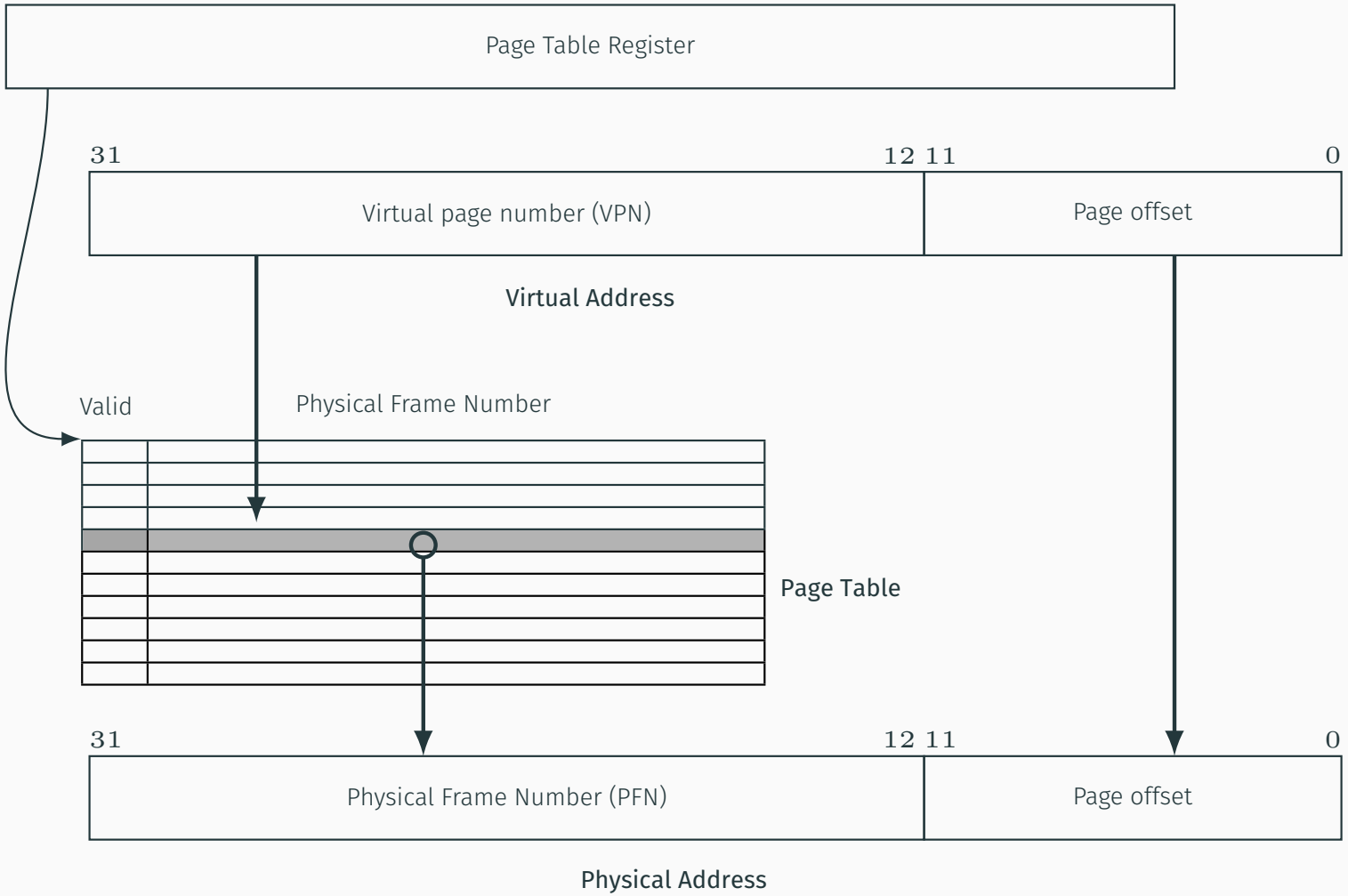
Mapping Virtual Addresses to Physical Addresses

Paged Mapping



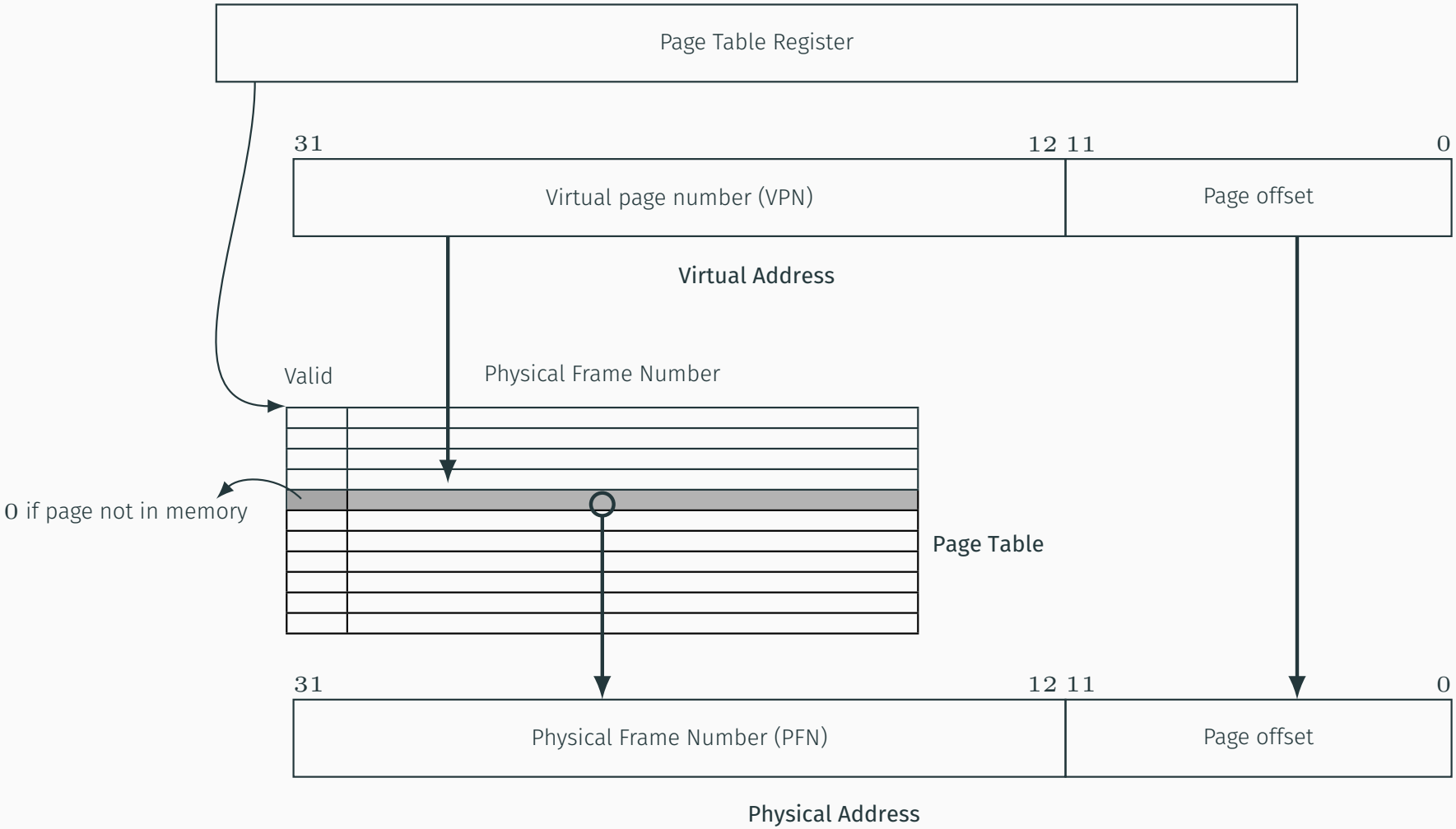
Mapping Virtual Addresses to Physical Addresses

Paged Mapping



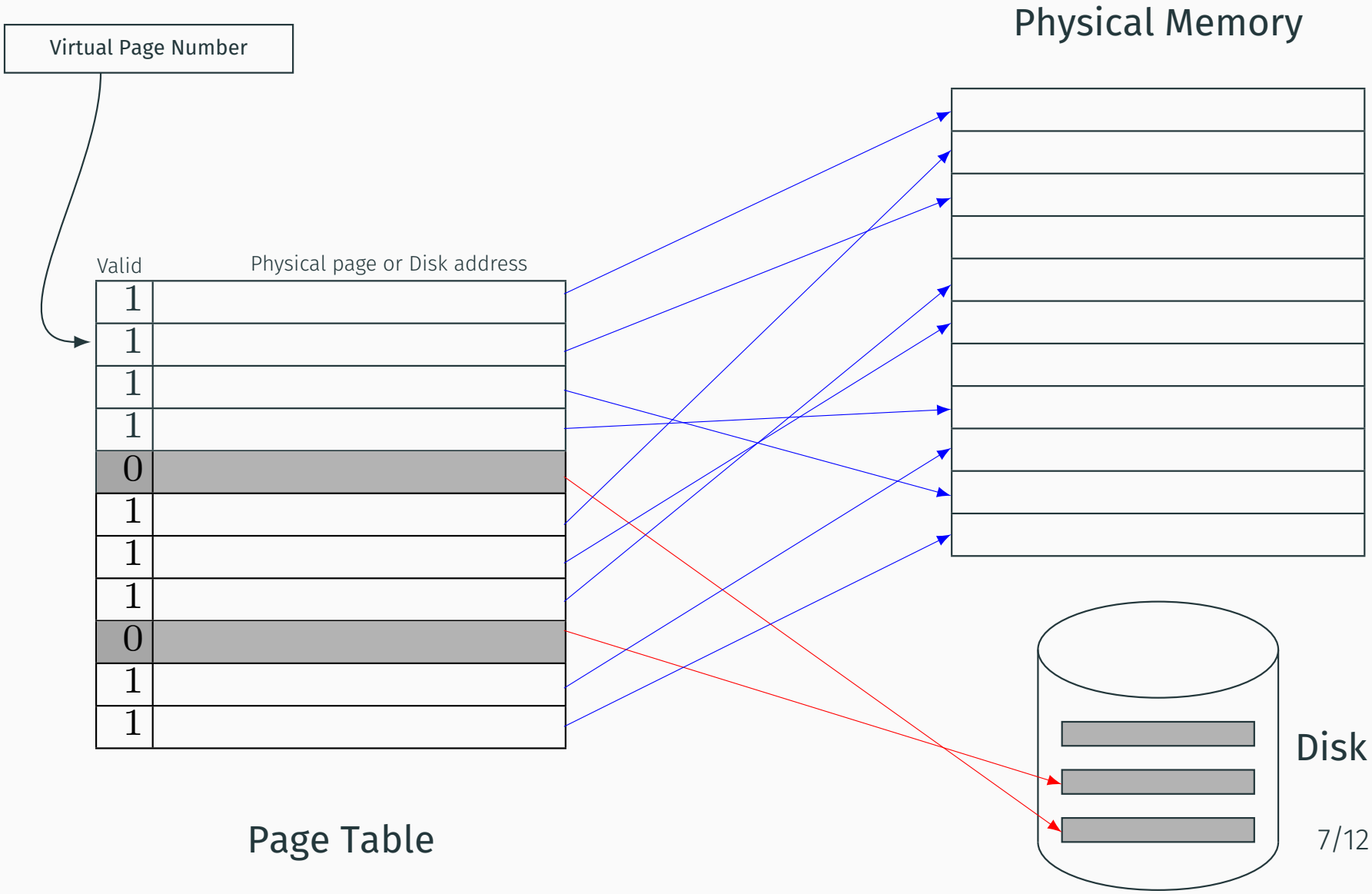
Mapping Virtual Addresses to Physical Addresses

Paged Mapping

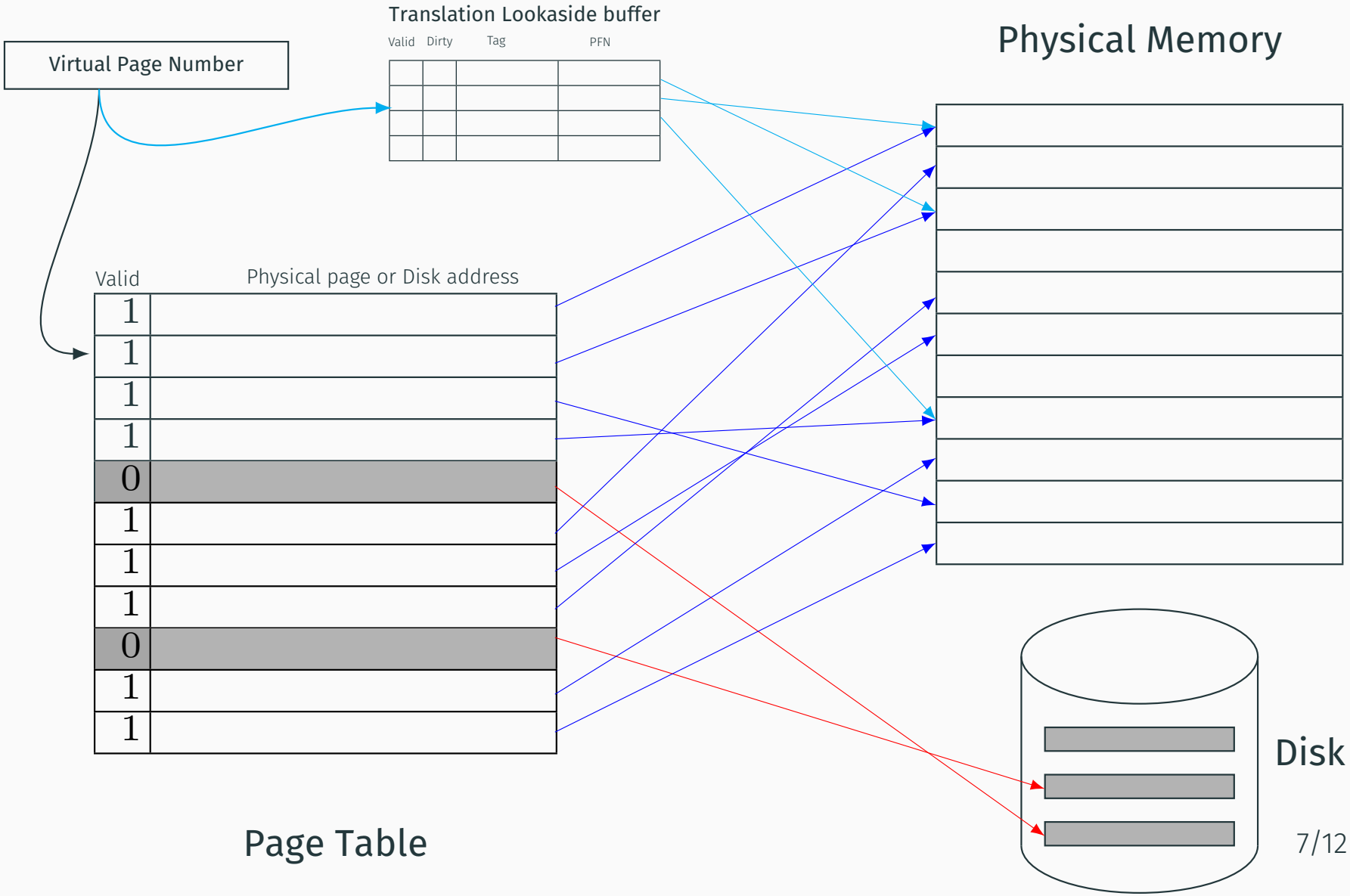


Mapping Virtual Addresses to Physical Addresses

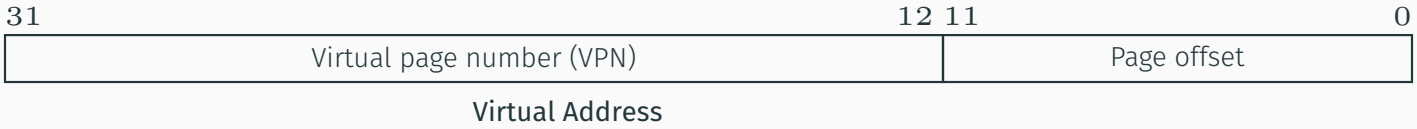
Paged Mapping



Paged Mapping

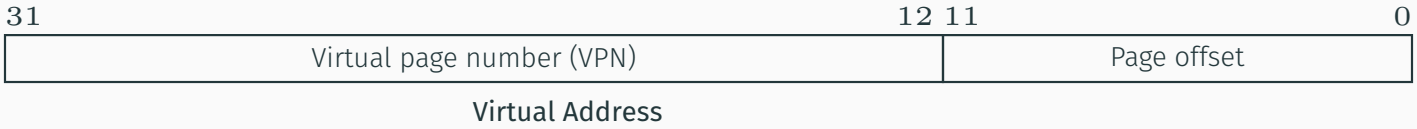


Cache and Virtual Memory



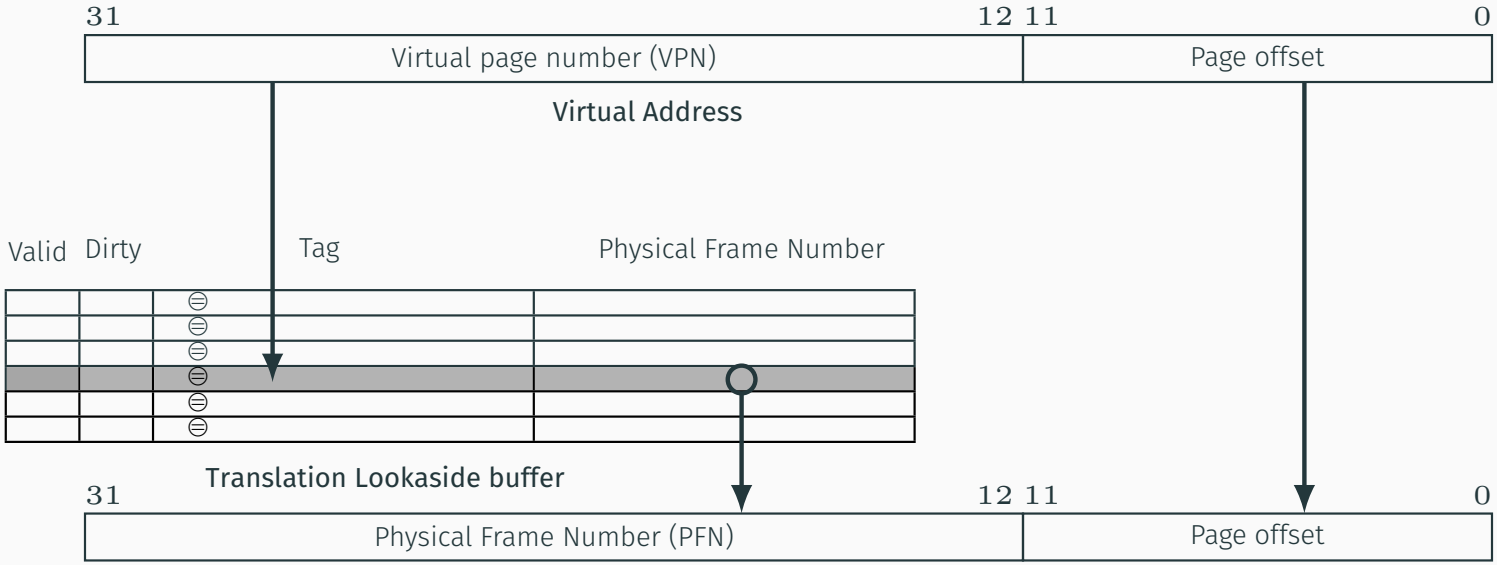
Virtual Address

Cache and Virtual Memory

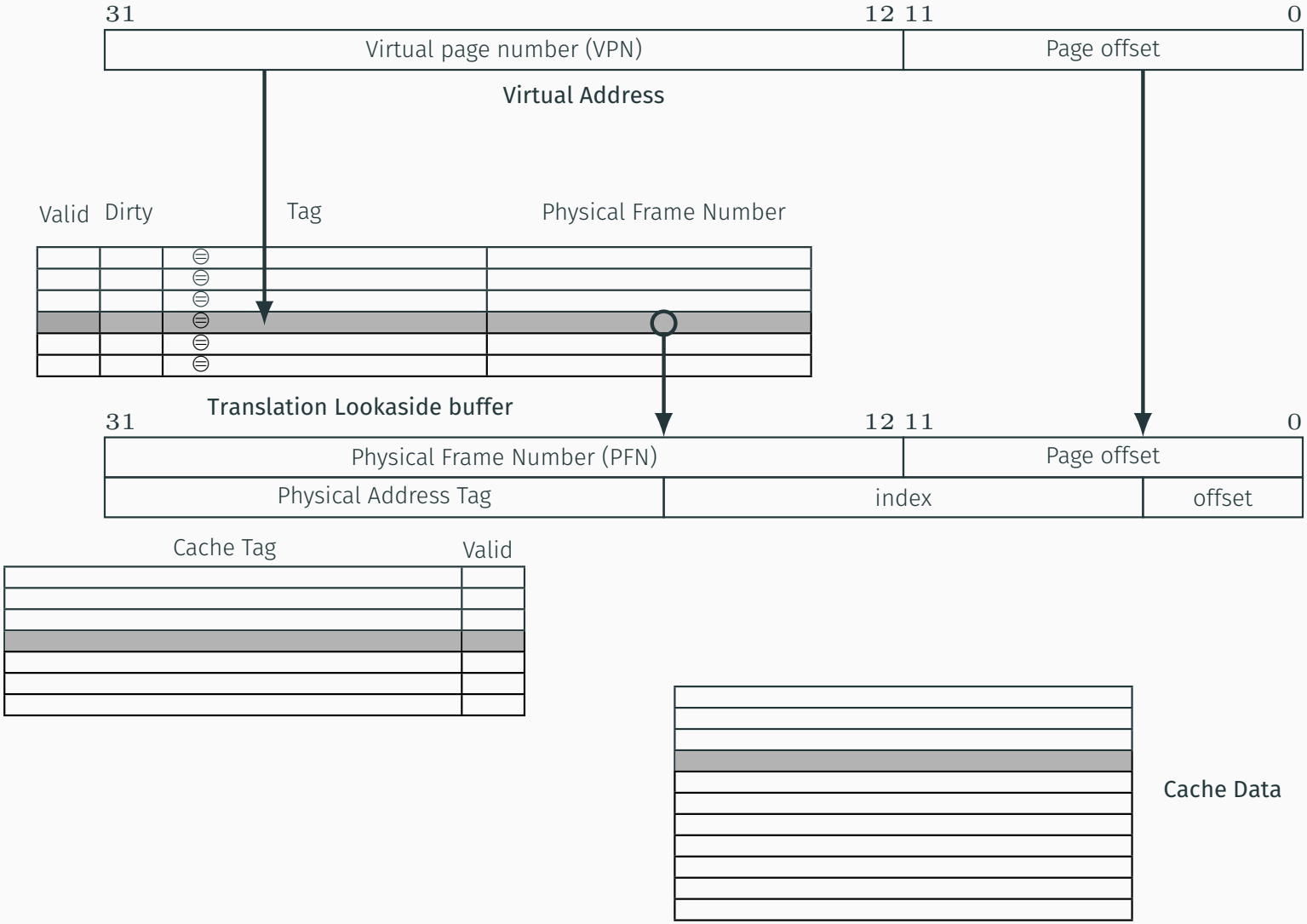


Virtual Address

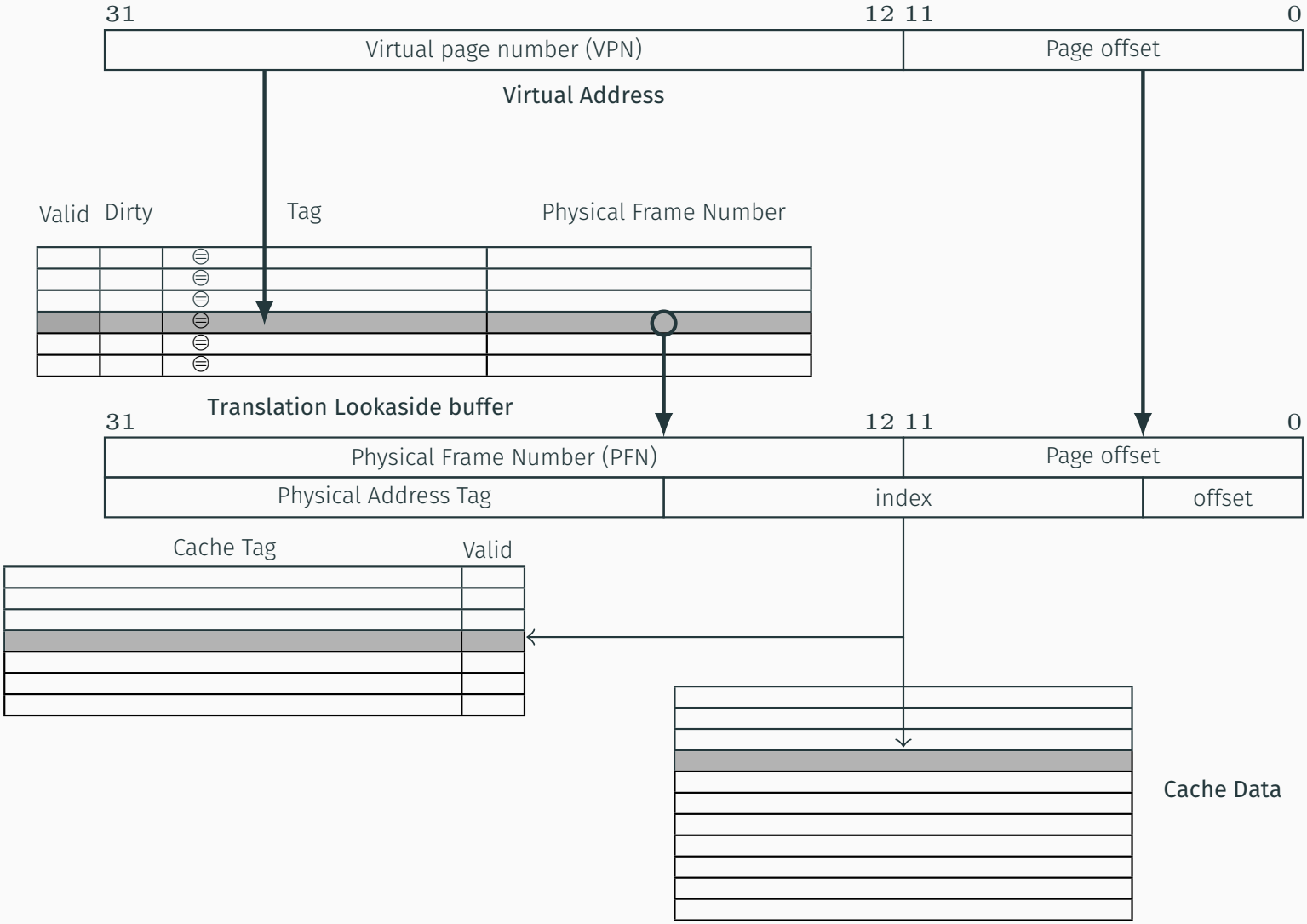
Cache and Virtual Memory



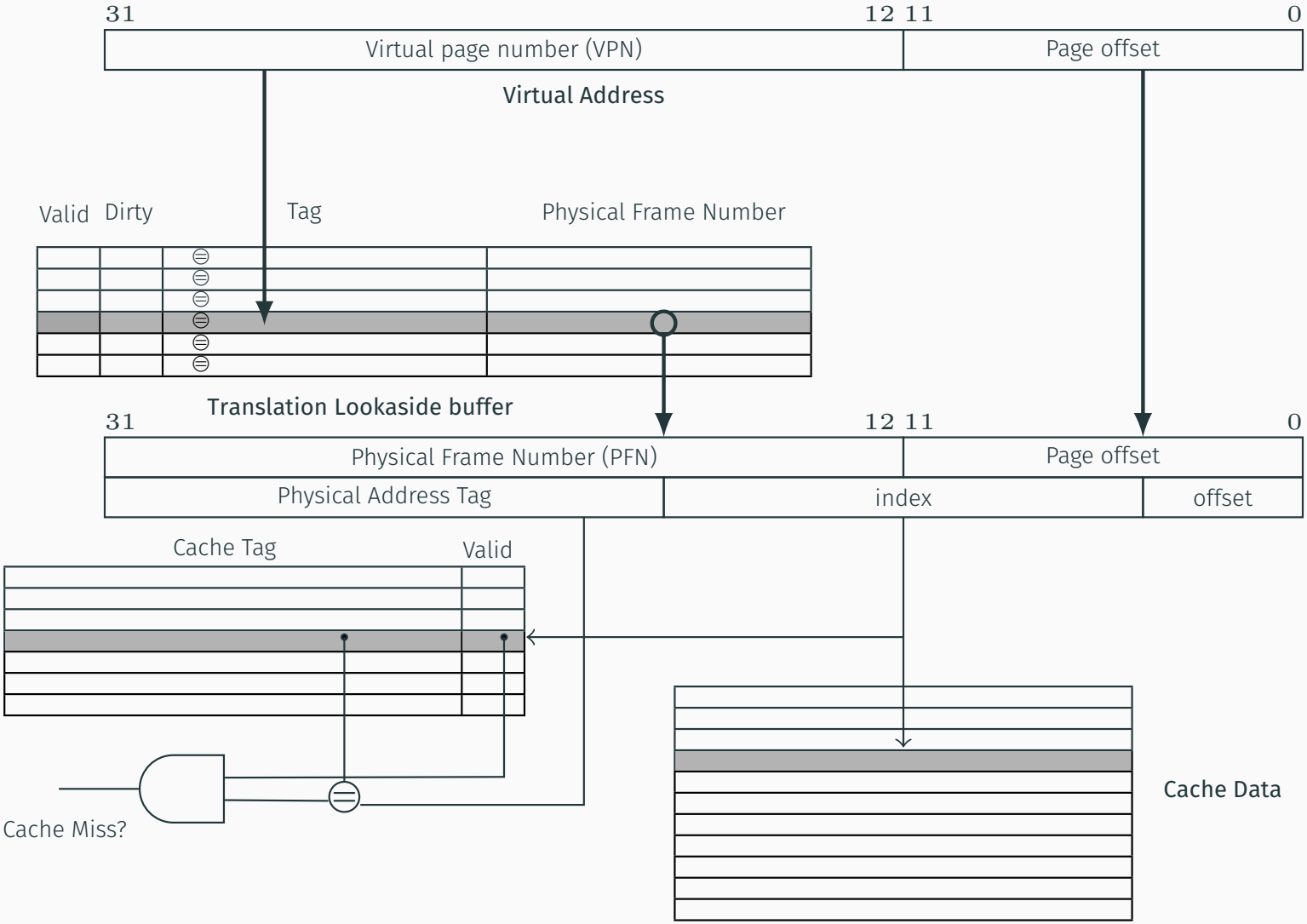
Cache and Virtual Memory



Cache and Virtual Memory



Cache and Virtual Memory



Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Miss	Hit	Hit

- **TLB** Misses, but entry present in Page Table. Data found in cache.

Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Miss	Hit	Hit
Miss	Hit	Miss

- **TLB** Misses, but entry present in Page Table. Data not in cache.

Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Miss	Hit	Hit
Miss	Hit	Miss
Miss	Miss	Miss

- **TLB** Misses, followed by page table miss (**Page Fault**). Always followed by **Compulsory Cache miss**

Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Hit	Miss	Miss

- Cannot have entry in **TLB** if page is not in memory.

Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Hit	Miss	Miss
Hit	Miss	Hit

- Cannot have entry in **TLB** if page is not in memory.

Cache Misses, TLB Misses and Page Faults

TLB	Page Table	Cache
Hit	Miss	Miss
Hit	Miss	Hit
Miss	Miss	Hit

- Cannot have data in cache if page is not in memory.

Summary

Caches

- Cache Block(Line)

Virtual Memory

- Memory Page

Summary

Caches

- Cache Block(Line)
- Cache Miss

Virtual Memory

- Memory Page
- Page Fault

Summary

Caches

- Cache Block(Line)
- Cache Miss
- Cache Size: 32-64 B

Virtual Memory

- Memory Page
- Page Fault
- Page Size: 4KB

Summary

Caches

- Cache Block(Line)
- Cache Miss
- Cache Size: 32-64 B
- N-Way set associative

Virtual Memory

- Memory Page
- Page Fault
- Page Size: 4KB
- Fully Associative