Sistemas Operativos: VM Paging

Pedro F. Souto (pfs@fe.up.pt)

May 6, 2020

Memory Virtualization

Idea

Goal Illusion that each process has its own memory (address space)

Mechanism Address translation

- On every memory access, the VM subsystem maps the virtual address to a physical address

Early Implementations (HW based)

Base and Bounds

+ Fast and simple
- Wastes lots of physical memory

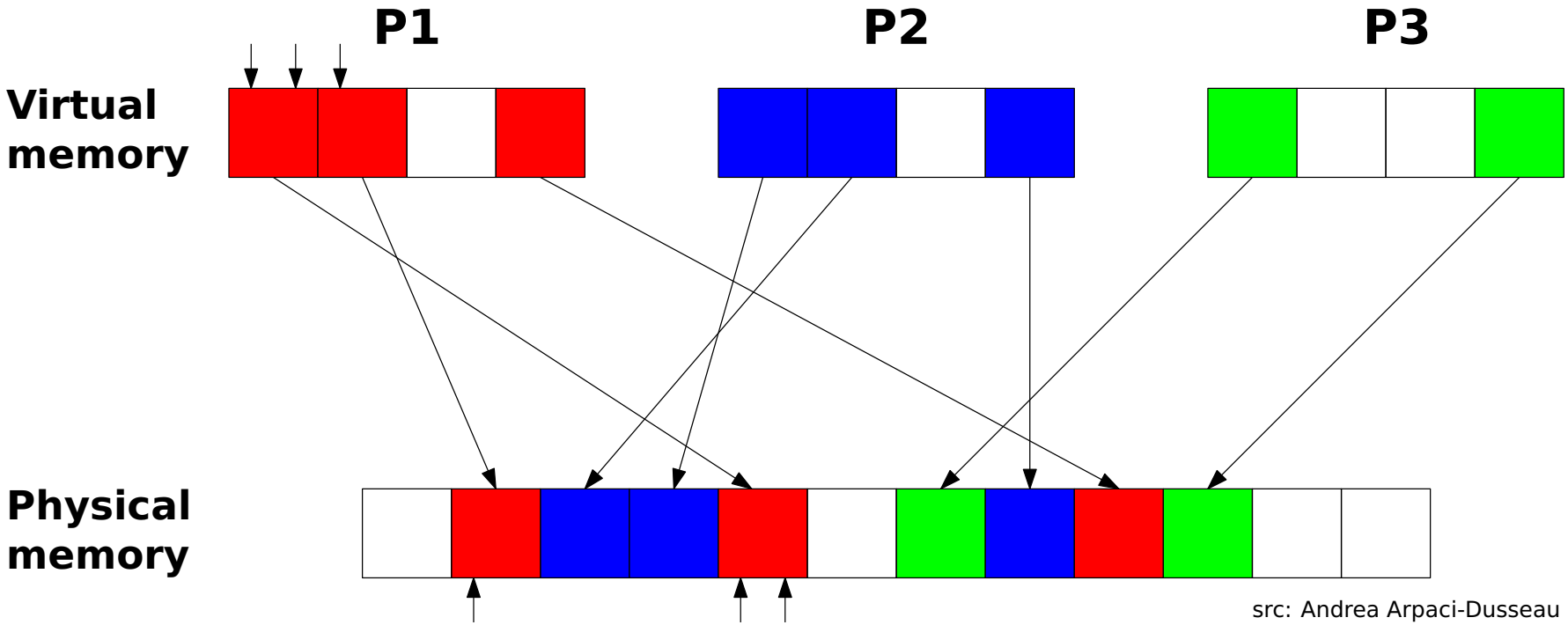
Segmentation

+ Still, fast and simple
- Mainly external fragmentation
  - But also, internal fragmentation

Paging

Idea

1. Divide address space into *fixed-size* (2^n) units: *pages*
   - Typically 4KiB, but also 8KiB or even 1 MiB (super-pages)
2. Divide physical memory in same size units: *physical pages* or *page frames*
3. Map virtual pages to page frames
   - Relocate each page independently in memory.



src: Andrea Arpaci-Dusseau

Paging: Address Translation

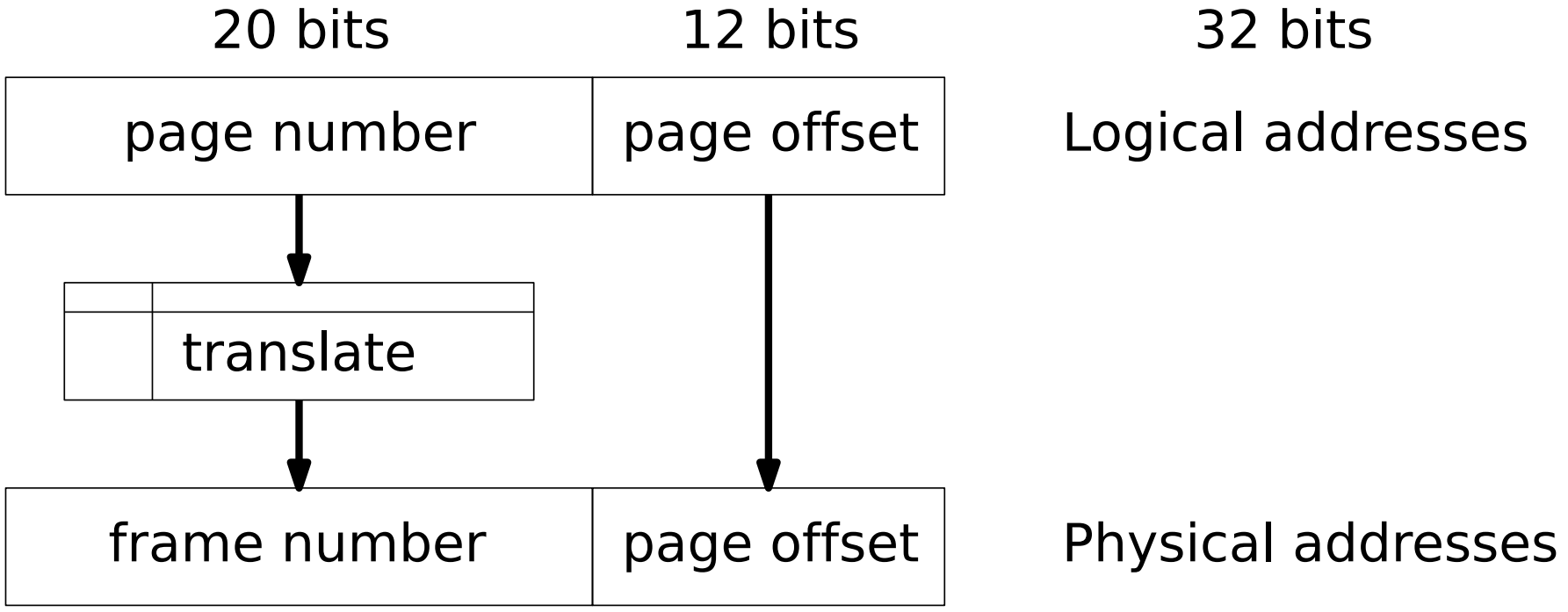
Question How to translate virtual address to physical address?

Answer Map the virtual page number (VPN) to the page frame number (PFN)

- Virtual pages and page frames have the same size

  High order bits of VA specify the page number

  Low order bits of VA specify the offset within page



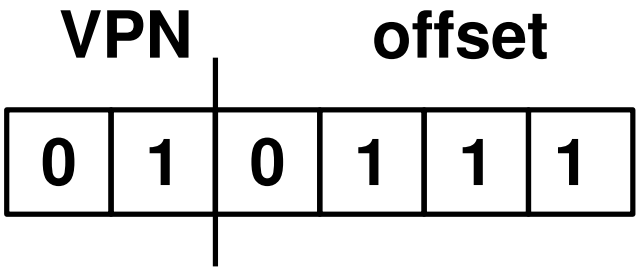
No addition needed; just append bits correctly

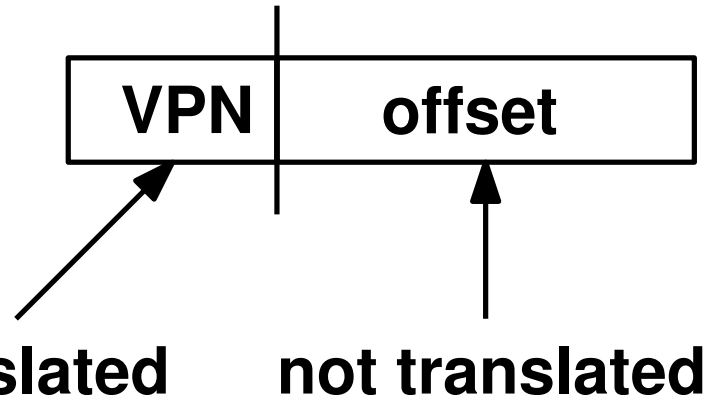
Paging: Address Translation

Page size
16 bytes -> $log_2(16) = 4$ bits for offset

Address space
64 bytes -> $log_2(64) = 6$ bits for address -> 2 bits for VPN

LD 0x17,R1





Virtual page number (VPN) used to compute the page frame number (PFN)

Offset append it to the PFN, without any translation

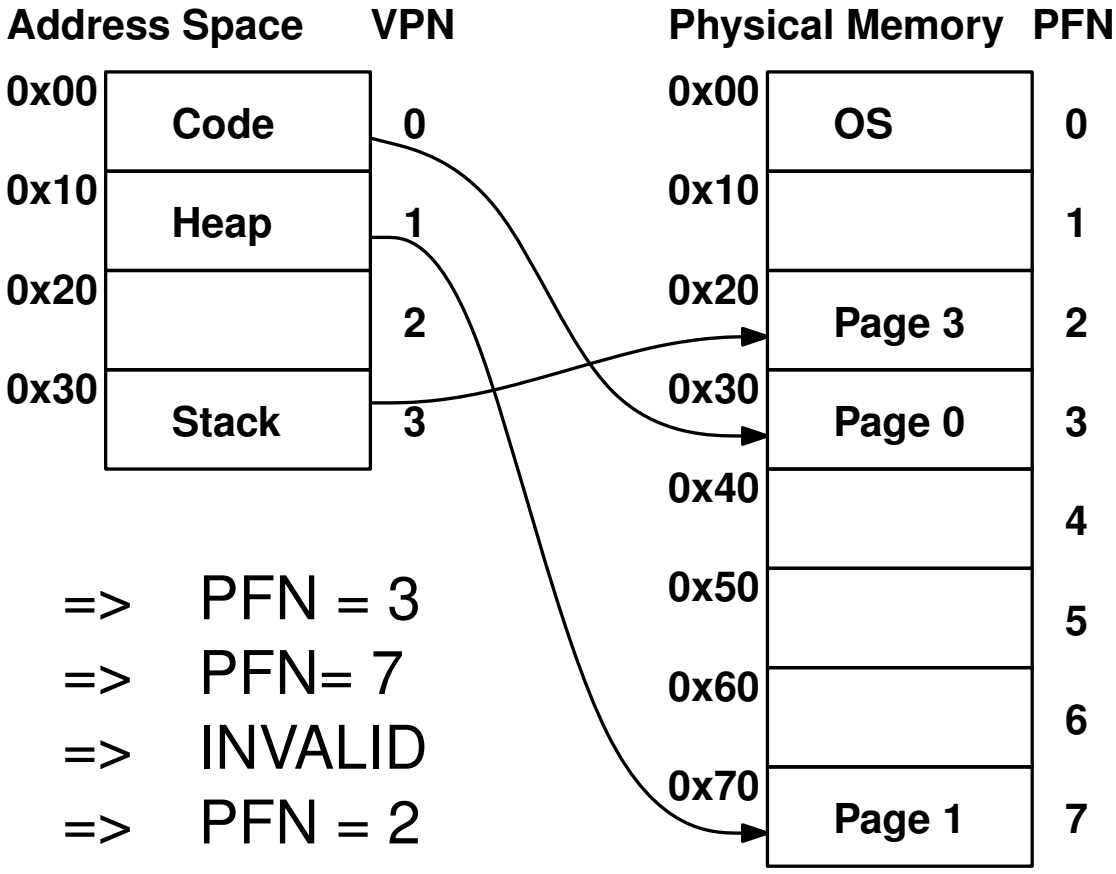
Virtual Page Number to Page Frame Number Mapping

Page size 16 bytes

Physical memory 8 page frames

Address space 64 bytes, i.e. 4 pages. Let:

- page 0 for code, page 1 for heap, page 3 for stack
- page 2, unused

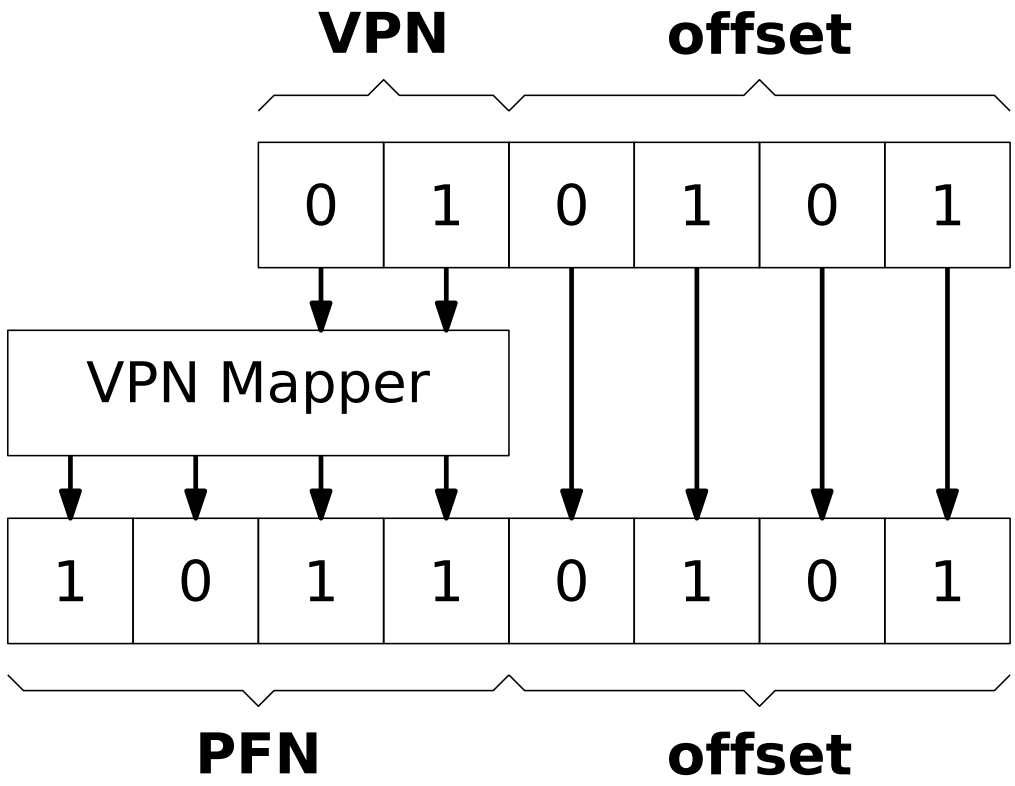


Mapping

VPN = 0 => PFN = 3
VPN = 1 => PFN= 7
VPN = 2 => INVALID
VPN = 3 => PFN = 2

Virtual Page Number to Page Frame Number Mapping

The number of bits of the virtual address does not need to equal the number of bits of the physical address

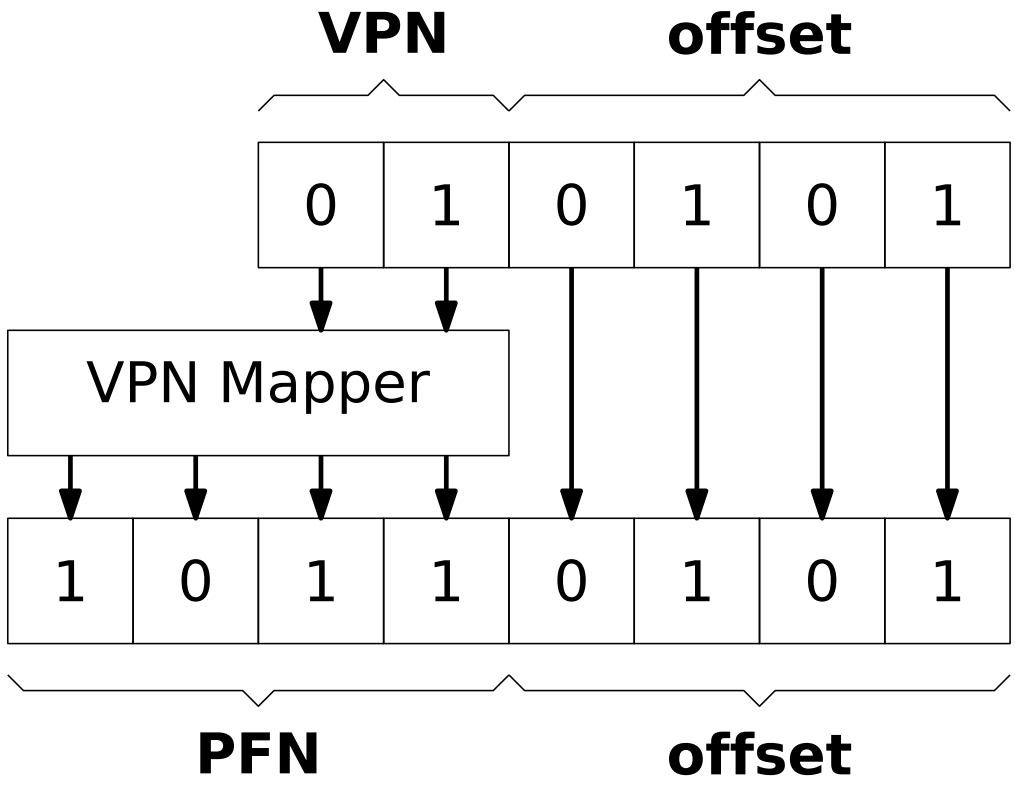


src: Andrea Arpaci-Dusseau

Question What data structure should we use for translating VPN to PPN?

Virtual Page Number to Page Frame Number Mapping

The number of bits of the virtual address does not need to equal the number of bits of the physical address

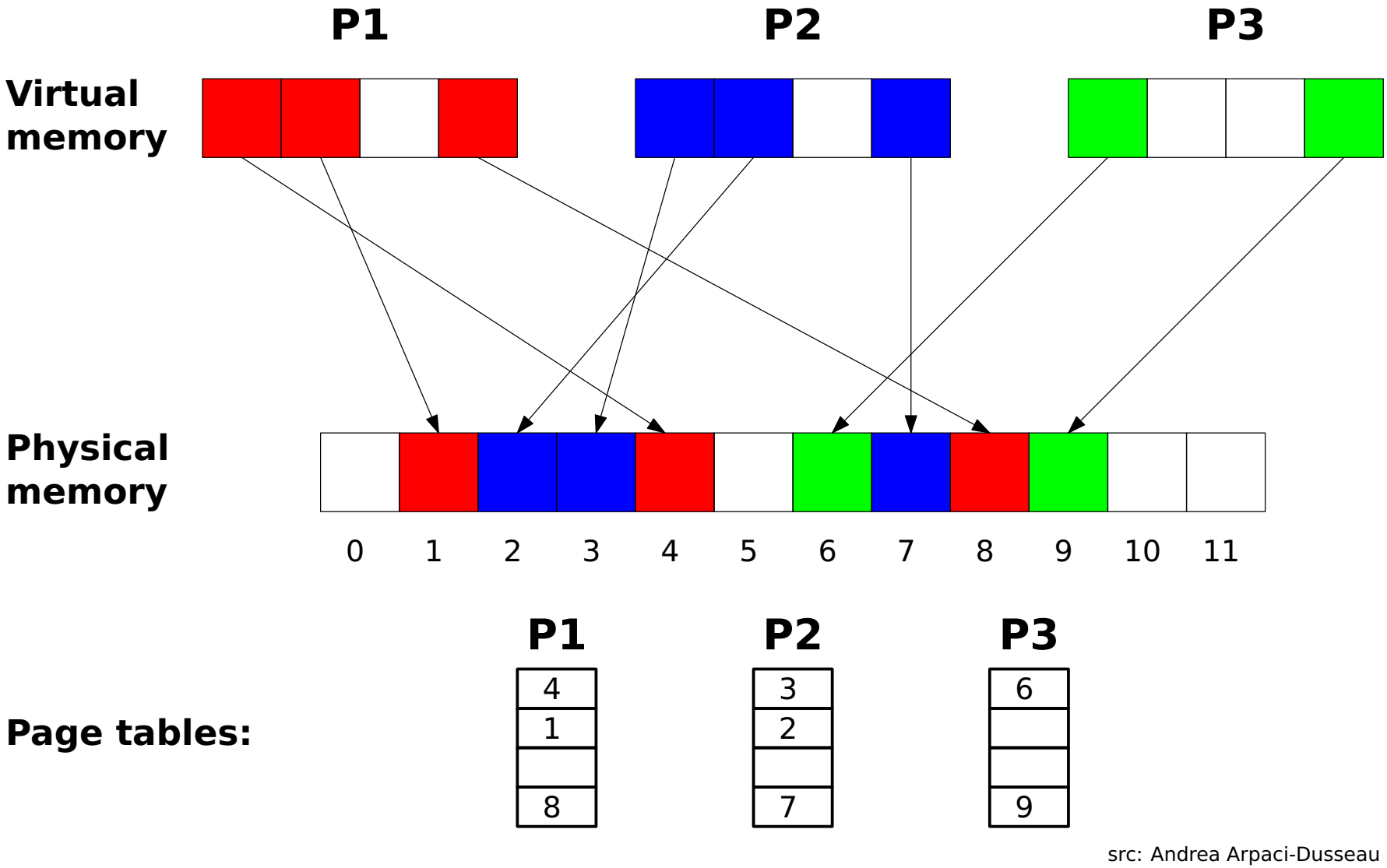


src: Andrea Arpaci-Dusseau

Question What data structure should we use for translating VPN to PPN?

Answer An array: the **page table**

Page Table: Examples



src: Andrea Arpaci-Dusseau

IMP. There must be a page-table per process.

Page Table Entries (PTE)

PT Info (other than Page Frame Number (PFN))

Valid bit allows the address space to have holes

- An attempt to access to an invalid page generates a "Segmentation fault"

Supervisor bit whether may be accessed in user mode

Protection bits whether it is possible to read, write or execute

Present bit to be discussed later

Reference bit to be discussed later

Dirty bit to be discussed later

Note

- Usually the format of a PTE is determined by the HW
  - Although, in some RISC architectures, the HW delegates to the OS the handling of paging.

Page Table: Where to store it?

Question How big is a typical page table?

Assume

- a **32-bit** address space
- a 4KiB page
- a 4 byte page table entry (PTE)

Answer

- Page table size = Num entries * size of PTE
- Num entries = num virtual pages = $2^{\text{bits for VPN}}$
- Bits for VPN = 32 - number of bits for page offset
  $= 32 - \log_2(4\text{KiB}) = 32 - 12 = 20$
- Num entries = 2^{20} = 1Mi
- Page table size = Num entries * 4 bytes = 4 MiB

Where is the PT Stored?

Page Table: Where to store it?

Question How big is a typical page table?

Assume

- a **32-bit** address space
- a 4KiB page
- a 4 byte page table entry (PTE)

Answer

- Page table size = Num entries * size of PTE
- Num entries = num virtual pages = $2^{\text{bits for VPN}}$
- Bits for VPN = 32 - number of bits for page offset
  $= 32 - \log_2(4\text{KiB}) = 32 - 12 = 20$
- Num entries = 2^{20} = 1Mi
- Page table size = Num entries * 4 bytes = 4 MiB

Where is the PT Stored? In memory

- Each process has its own PT
- HW finds PT using a register: the PT base register (PTBR) (e.g., CR3 on x86)

Virtual Address to Physical Address Translation: HW Implementation

1. Extract VPN from virtual address
   VPN = VA >> PAGE_SHIFT
2. Compute address of PTE
   PTBR + VPN
3. Read PTE from PT (in memory)
4. Extract PFN (if PTE valid)
5. Extract offset from virtual address
   OFFSET = VA & OFFSET_MASK
6. Concatenate offset to PFN
   PA = PFN | OFFSET

Issue Address translation requires one memory access (to the PTE) – step 3

- Assuming, the PT is a huge array
- So, for every memory reference, paging requires one additional memory reference

OS & HW Involvement

Hardware

On every memory access address translation

OS

Upon process creation allocate page frames and create and initialize page table

Upon context switch save PT base register to process control block (PCB) of process previously running, and load PT base register from PCB of dispatched process

Upon process AS "grow" allocate free page frames and add entries to page table

Upon process termination free all page frames and the page table

Paging vs. Segmentation

Pros

No external fragmentation

- Any page can be placed in any frame in physical memory

Fast to allocate and free

- Just use a bitmap of free/allocated page frames

Cons

Slow translating an address requires a memory access

- To the corresponding page table

Large storage requirements

- Each process has its own PT
- For a 32 bit AS, with 4KiB, PT size can be as large as 4 MiB
  - Even if the process uses only a few pages

Virtual Address Format

Question How does the format of an address affect the number of pages and the size of pages?

No. of bits used to specify the VPN determines the number of pages

No. of bits used to specify the offset determines the size of a page

Example 1 Consider a virtual address with 32 bits

Page size of 4 KiB i.e. 12 LSB for offset

- Then, 20 (=32-12) MSB are used for the VPN, i.e. the address space has 1 Mi pages

What if page size of 1 MiB?