

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.
Seoul National University

Fall 2023

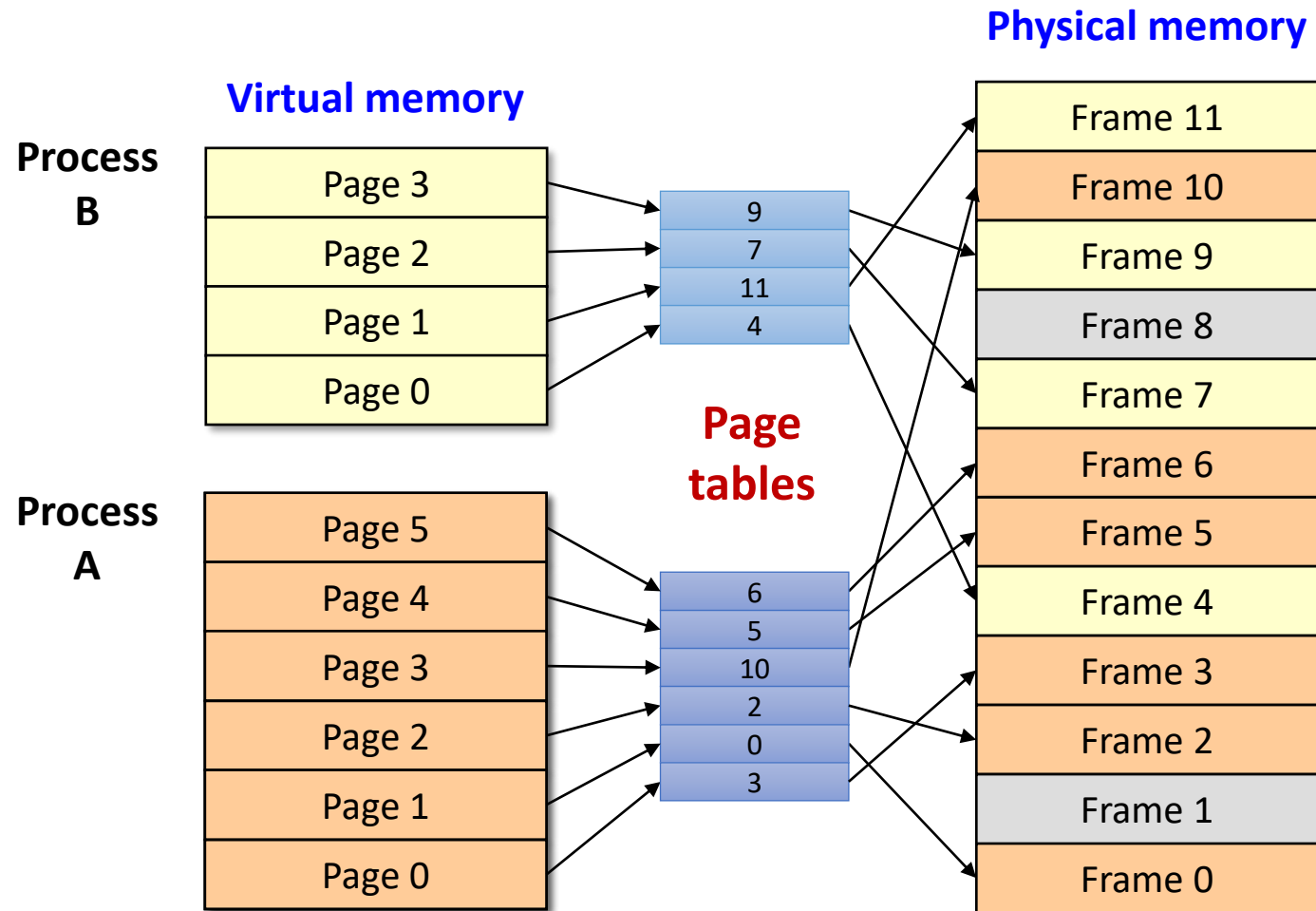
Paging



Paging

- Allows the physical address space of a process to be noncontiguous
 - Divide virtual memory into blocks of same size (**pages**)
 - Divide physical memory into fixed-size blocks (**frames**)
 - Page (or frame) size is power of 2 (typically 512B – 8KB)
- Eases memory management
 - OS keeps track of all free frames
 - To run a program of size n pages, need to find n free frames and load the program
 - Set up a **page table** to translate virtual to physical addresses
 - No _____ fragmentation

Paging Overview



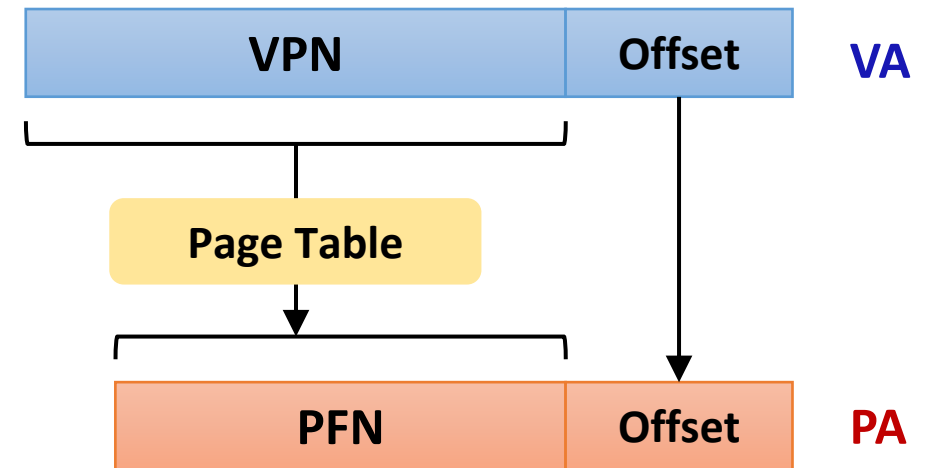
Address Translation (I)

■ Translating virtual addresses

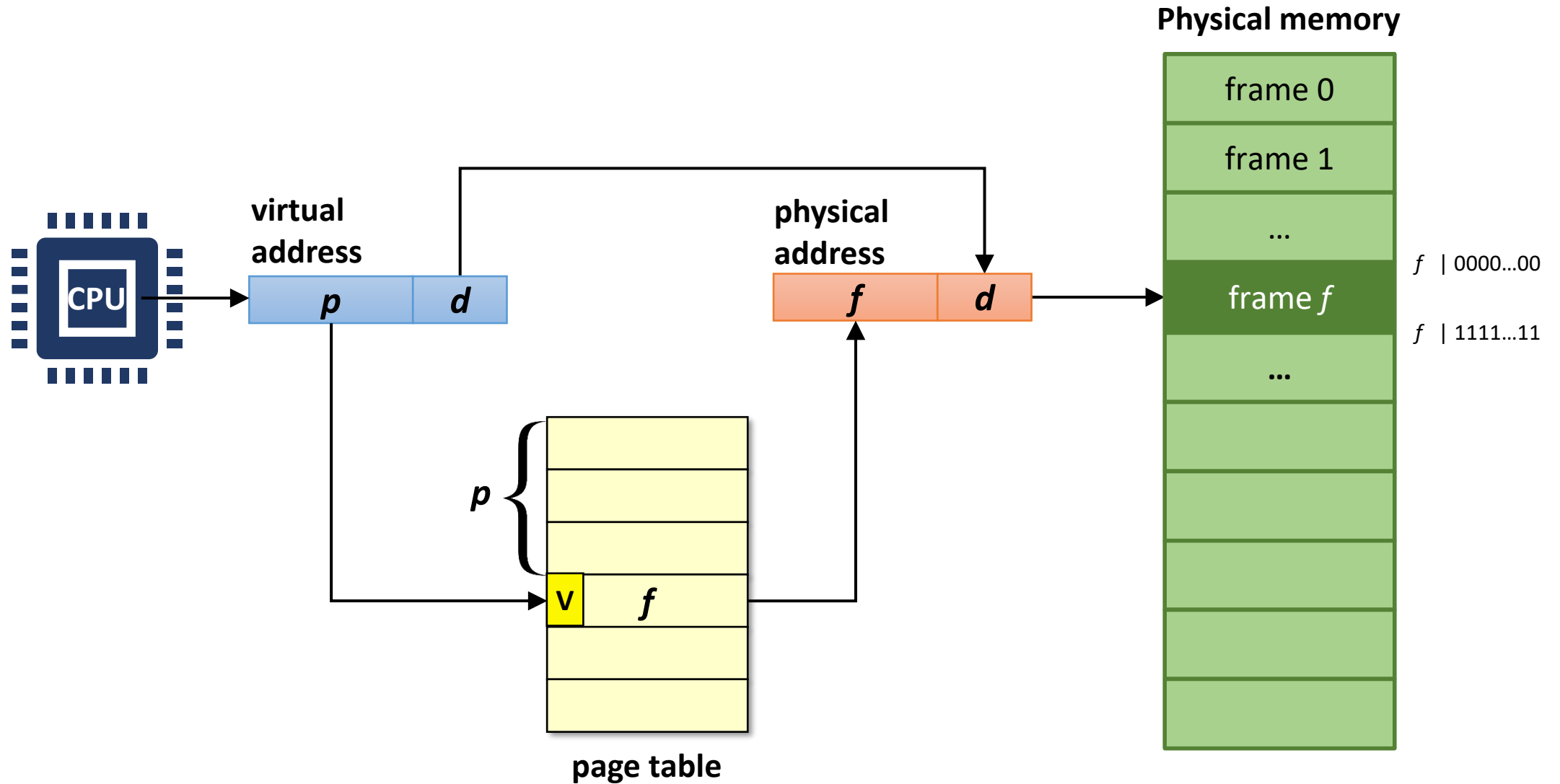
- A virtual address has two parts:
<Virtual Page Number (VPN), Offset>
- VPN is an index into the page table
- Page table determines Page Frame Number (PFN)
- Physical address is <PFN, Offset>
- Usually, $|VPN| \geq |PFN|$

■ Page tables

- Managed by _____
- Map VPN to PFN
- One Page Table Entry (PTE) per page in virtual address space



Address Translation (2)

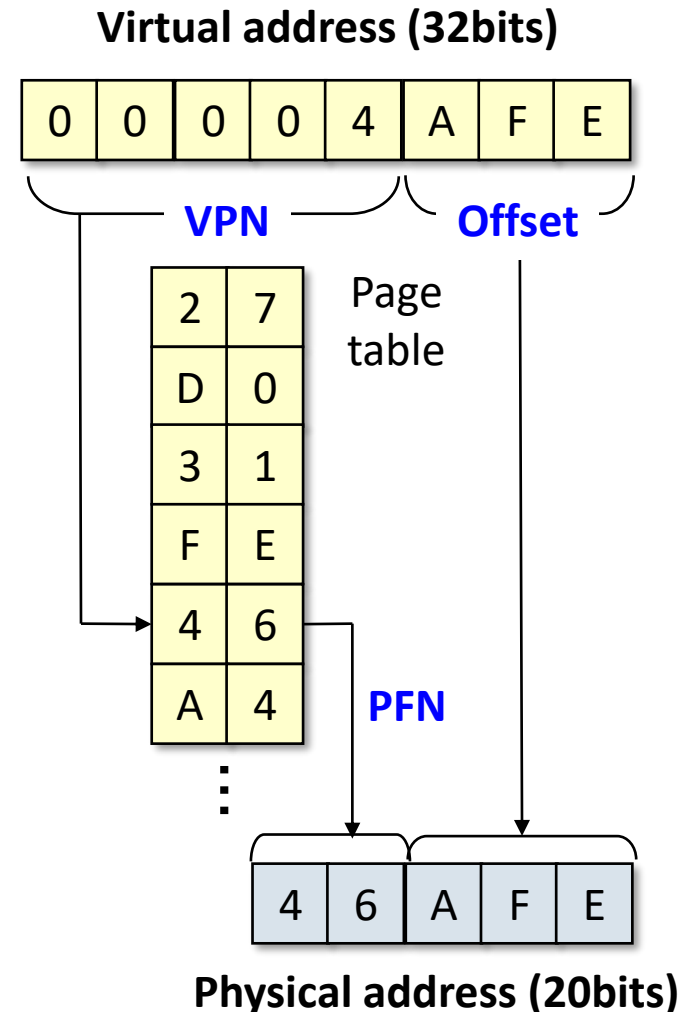


Address Translation (3)

■ Example

- Virtual address: 32 bits
- Physical address: 20 bits
- Page size: 4KB
- 4 bytes / PTE

- Offset: _____ bits
- VPN: _____ bits
- Total number of PTEs: _____
- Page table size: _____

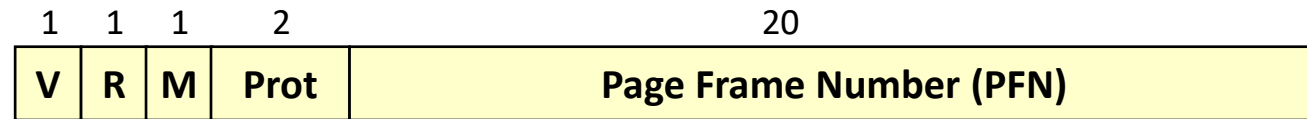


Protection

- **Separate page table for each process**
 - No way to access the physical memory of other processes
 - On context switch, an MMU register is set to point to the base address of the current page table (e.g., CR3 in x86, satp in RISC-V)
- **Page-level protection**
 - Memory protection is implemented by associating protection bits with each PTE
 - Valid / invalid bit
 - “Valid”: the page is in the process’ address space and in use
 - “Invalid”: the page is not allocated
 - Finer level of protection is possible for valid pages
 - Read-only, Read-write, or execute-only protections

PTE

■ Page Table Entry



- V (Valid) bit says whether or not the PTE can be used
 - It is checked each time a virtual address is used
- R (Reference) bit says whether the page has been accessed
 - It is set when a read or write to the page occurs
- M (Modify) bit says whether the page is dirty
 - It is set when a write to the page occurs
- Prot (Protection) bits control which operations are allowed
 - Read, Write, Execute, User/Kernel, etc.
- PFN (Page Frame Number) determines the physical frame

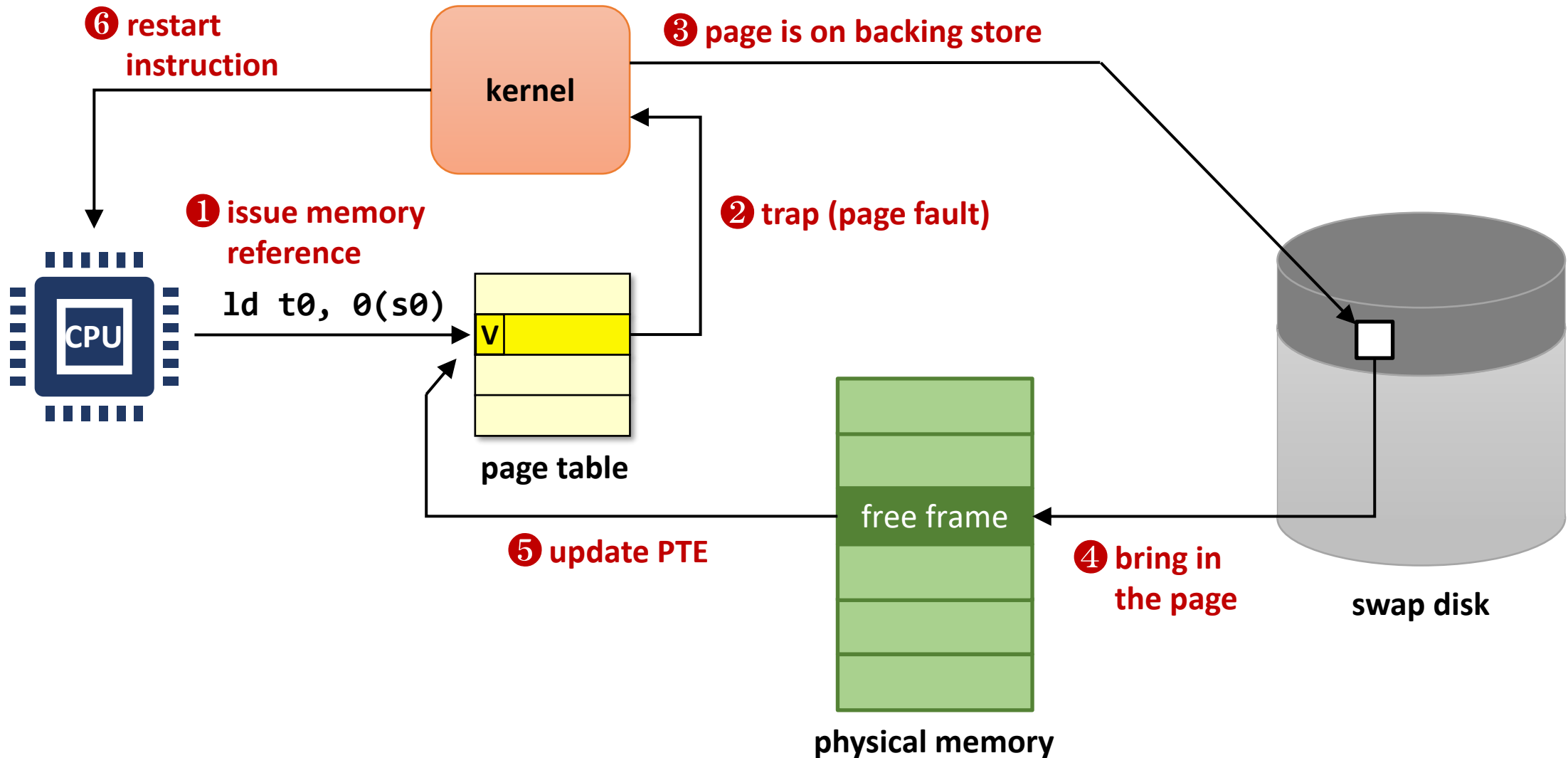
Demand Paging

- OS uses main memory as a (page) cache of all the data allocated by processes in the system
 - Bring a page into memory only when it is needed
 - Pages can be evicted from their physical memory frames
 - Evicted pages go to disk (only dirty pages are written)
 - Movement of pages is transparent to processes
- **Benefits**
 - Less I/O needed
 - Less memory needed
 - Faster response
 - More processes

Page Fault

- An exception raised by CPU when accessing invalid PTE
- Major page faults
 - The page is valid but not loaded into memory
 - OS maintains information on where to find the contents
 - Require disk I/Os
- Minor page faults
 - Page faults can be resolved without disk I/O
 - Used for lazy allocation (e.g., accesses to stack & heap pages)
 - Accesses to prefetched pages, etc.
- Invalid page faults
 - Segmentation violation: the page is not in use

Handling Page Faults



Paging: Pros

- No external fragmentation
- Fast to allocate and free
 - A list or bitmap for free page frames
 - Allocation: no need to find contiguous free space
 - Free: no need to coalesce with adjacent free space
- Easy to “page out” portions of memory to disk
 - Page size is chosen to be a multiple of disk block sizes
 - Use valid bit to detect reference to “paged-out” pages
 - Can run process when some pages are on disk
- Easy to protect and share pages

Paging: Cons

- **Internal fragmentation**
 - Wasted memory grows with larger pages
- **Memory reference overhead**
 - Doubles the number memory references per instruction
 - Solution: get hardware support (TLB)
- **Storage needed for page tables**
 - Needs one PTE for each page in virtual address space
 - 32-bit address space with 4KB page size: 2^{20} PTEs
 - 4 bytes/PTE: 4MB per page table
 - 100 processes in the system: total 400MB of page tables
 - Solution: store valid PTEs only or page the page table