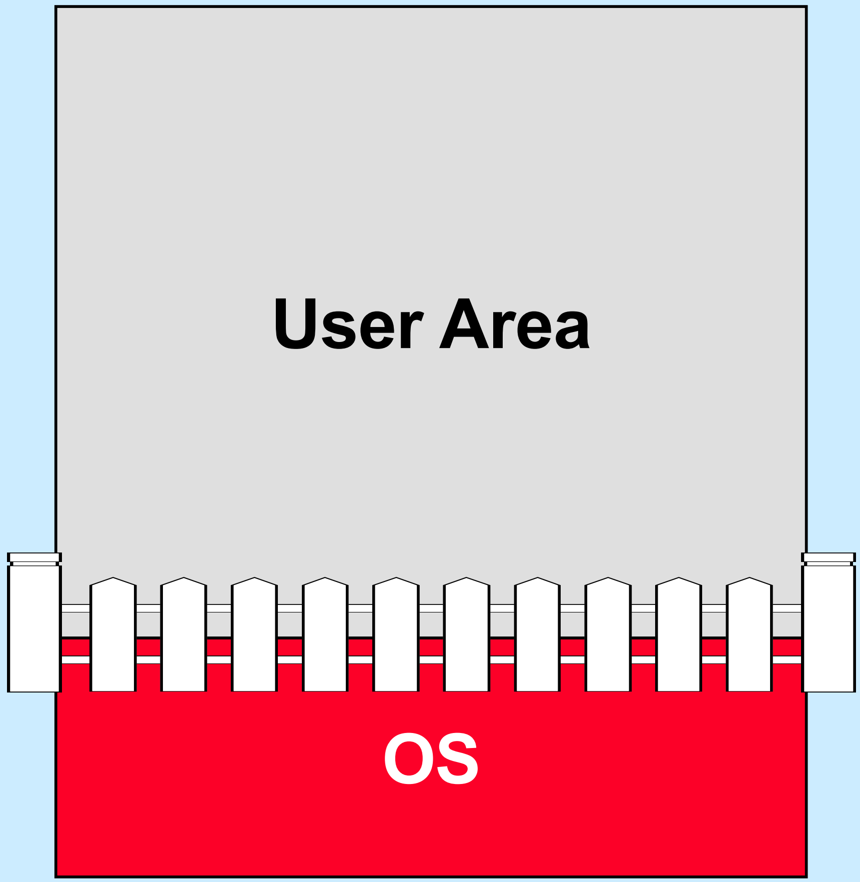
CS 33

Virtual Memory

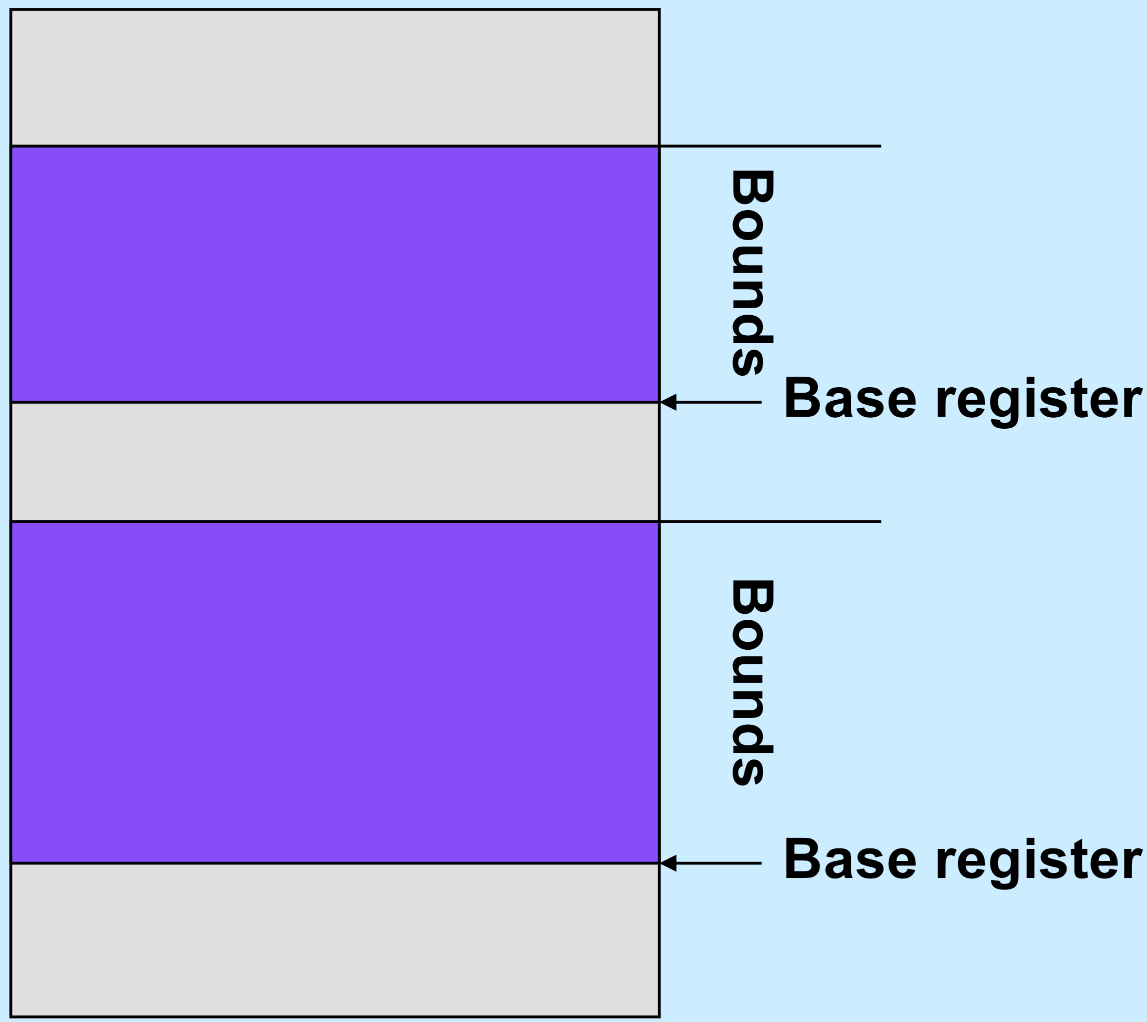
The Address-Space Concept

- Protect processes from one another
- Protect the OS from user processes
- Provide efficient management of available storage

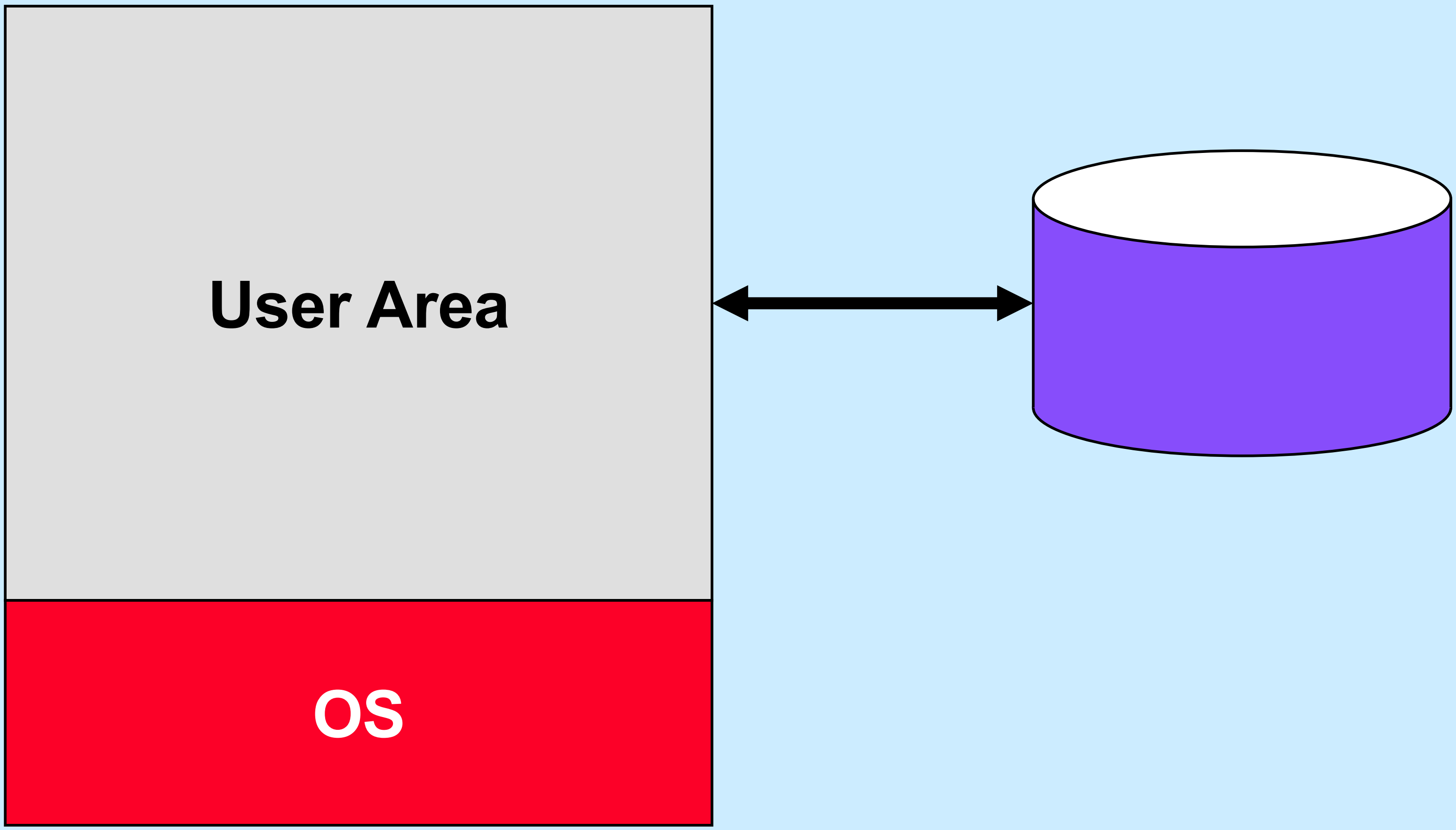
Memory Fence



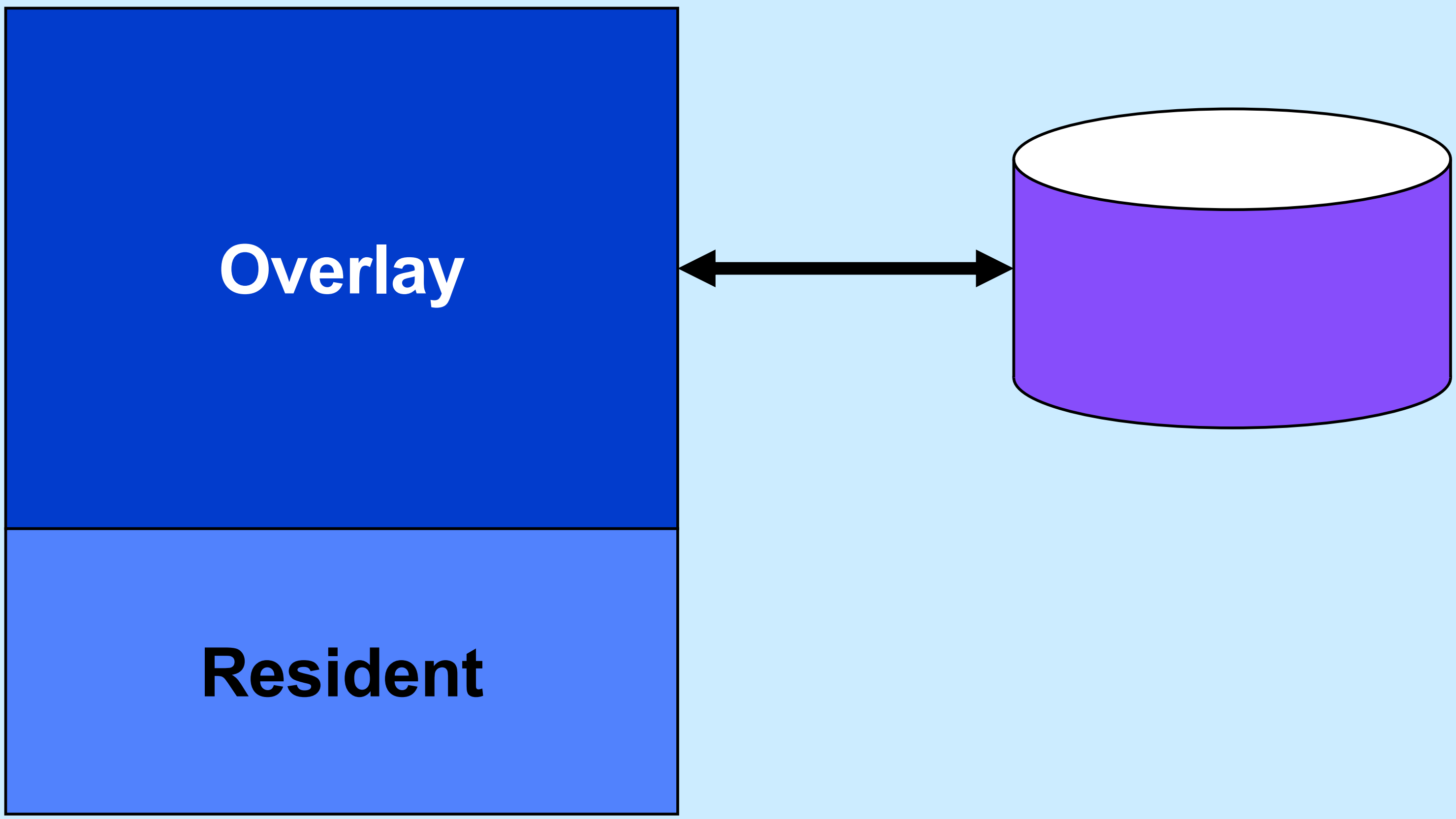
Base and Bounds Registers



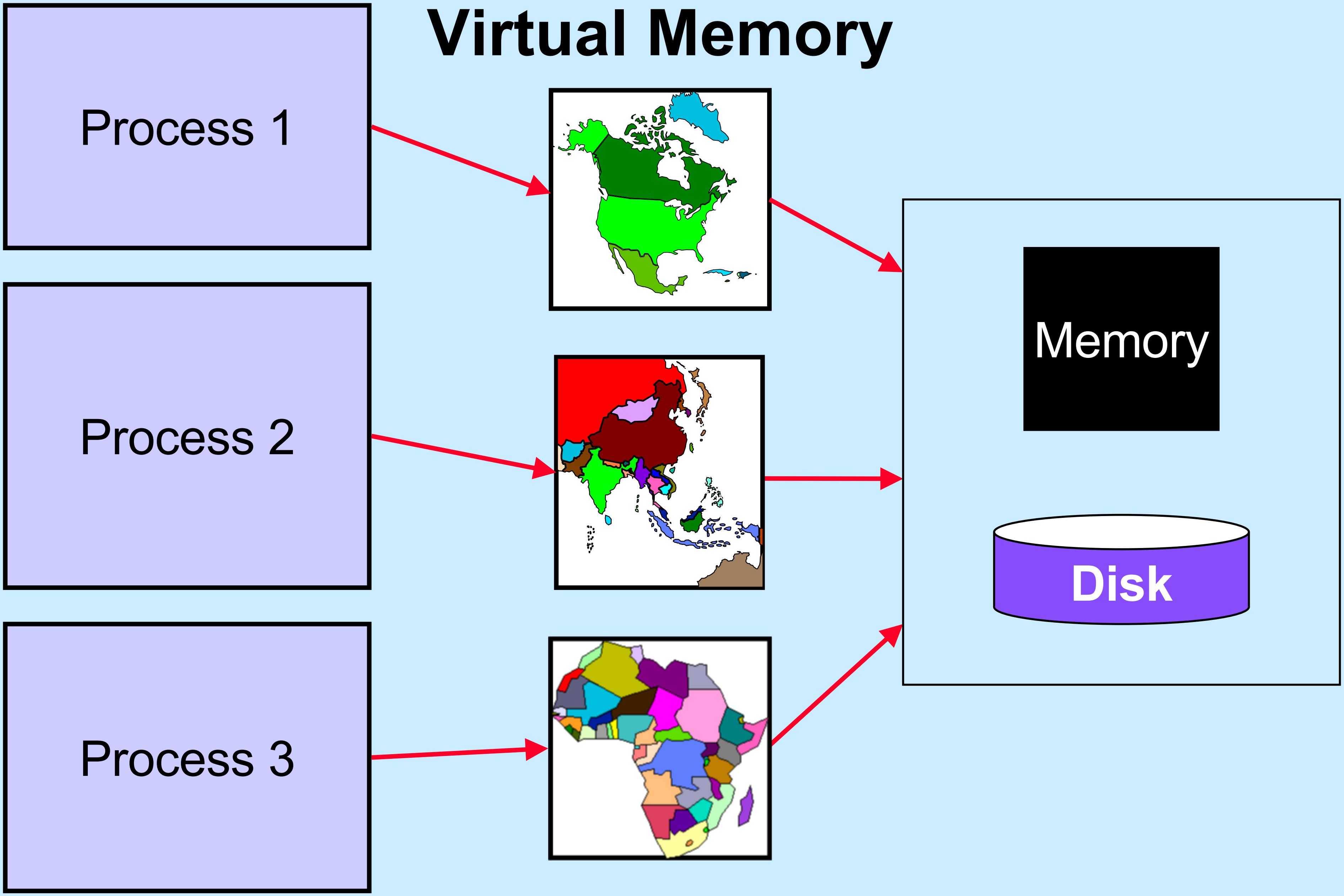
Swapping



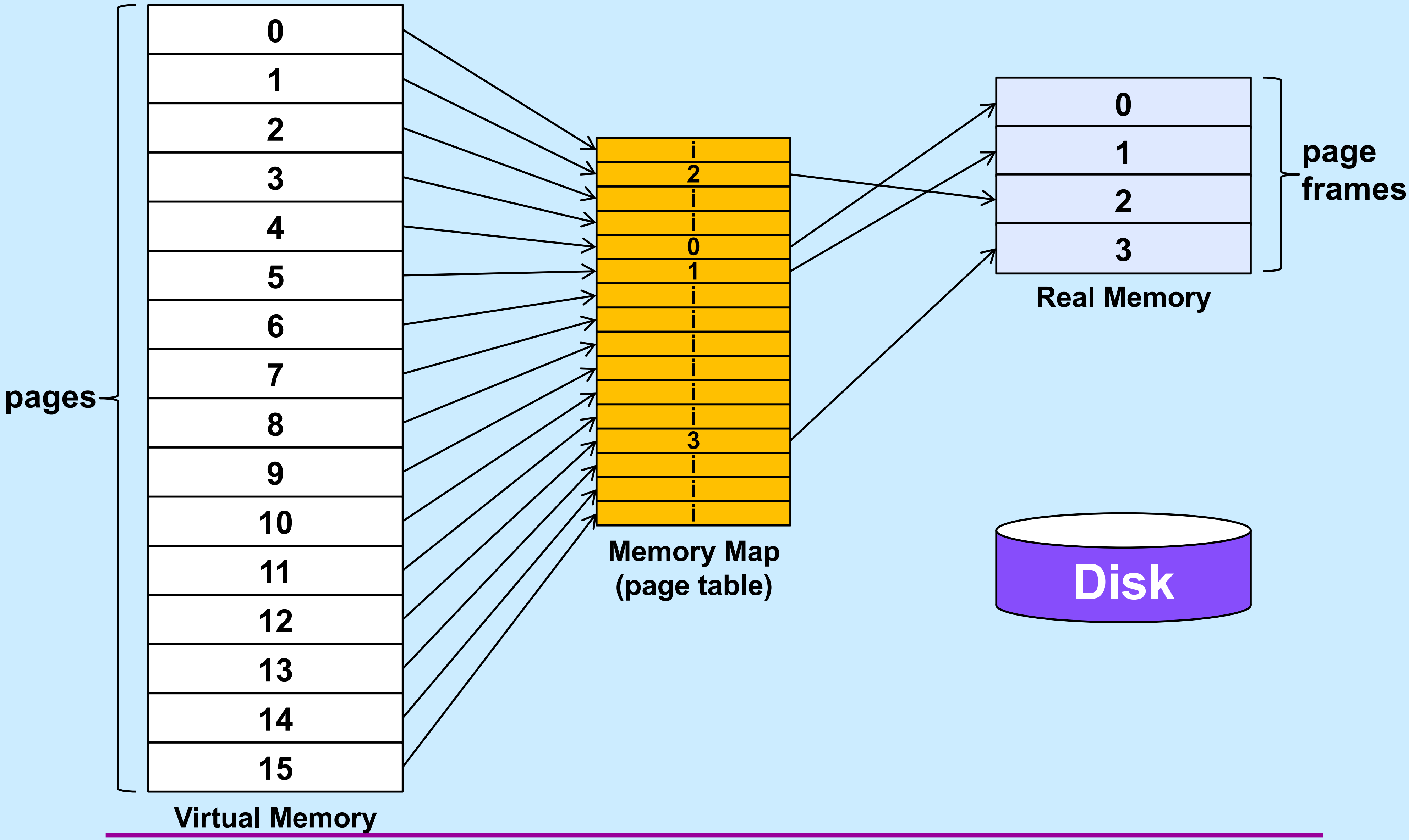
Overlays



Virtual Memory

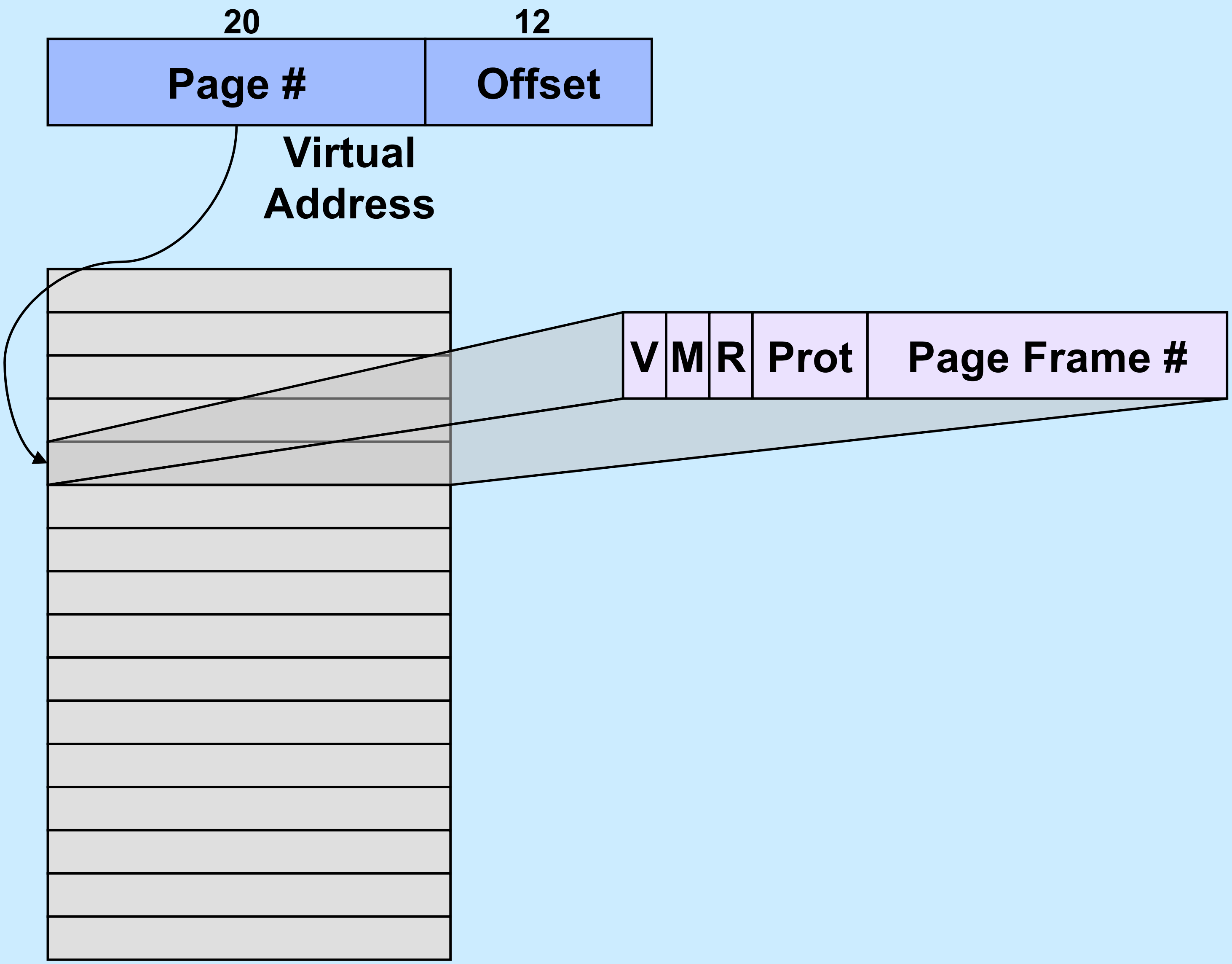


Memory Maps



pages
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Virtual Memory
i
2
i
i
0
1
i
i
i
i
i
i
3
i
i
i
Memory Map
(page table)
0
1
2
3
Real Memory
page
frames
Disk

Page Tables



Quiz 1

How many 2^{12}-byte pages fit in a 32-bit address space?

a) a little over a 1000

b) a little over a million

c) a little over a billion

d) none of the above

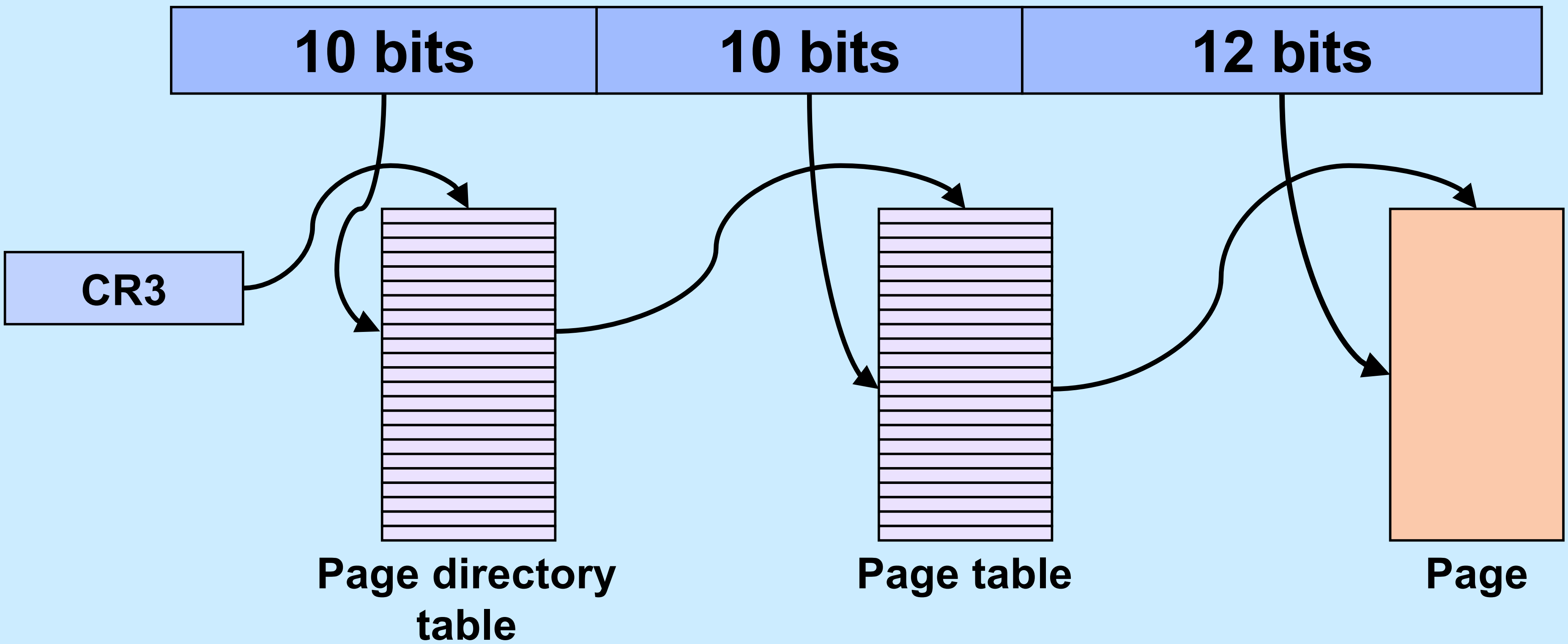
VM is Your Friend ...

- **Not everything has to be in memory at once**
  - pages brought in (and pushed out) when needed
  - unallocated parts of the address space consume no memory
    - » e.g., hole between stack and dynamic areas
- **What's mine is not yours** (and vice versa)
  - address spaces are disjoint
- **Sharing is ok though ...**
  - address spaces don't have to be disjoint
    - » a single page frame may be mapped into multiple processes
- **I don't trust you (or me)**
  - access to individual pages can be restricted
    - » read, write, execute, or any combination

Page-Table Size

- Consider a full 2^{32}-byte address space
  - assume 4096-byte (2^{12}-byte) pages
  - 4 bytes per page-table entry
  - the page table would consist of $2^{32}/2^{12}$ (= 2^{20}) entries
  - its size would be 2^{22} bytes (or 4 megabytes)
    - » at $100/gigabyte
      - around $0.40
- For a 2^{64}-byte address space
  - assume 4096-byte (2^{12}-byte) pages
  - 8 bytes per page-table entry
  - the page table would consist of $2^{64}/2^{12}$ (= 2^{52}) entries
  - its size would be 2^{55} bytes (or 32 petabytes)
    - » at $1/gigabyte
      - over $33 million

IA32 Paging

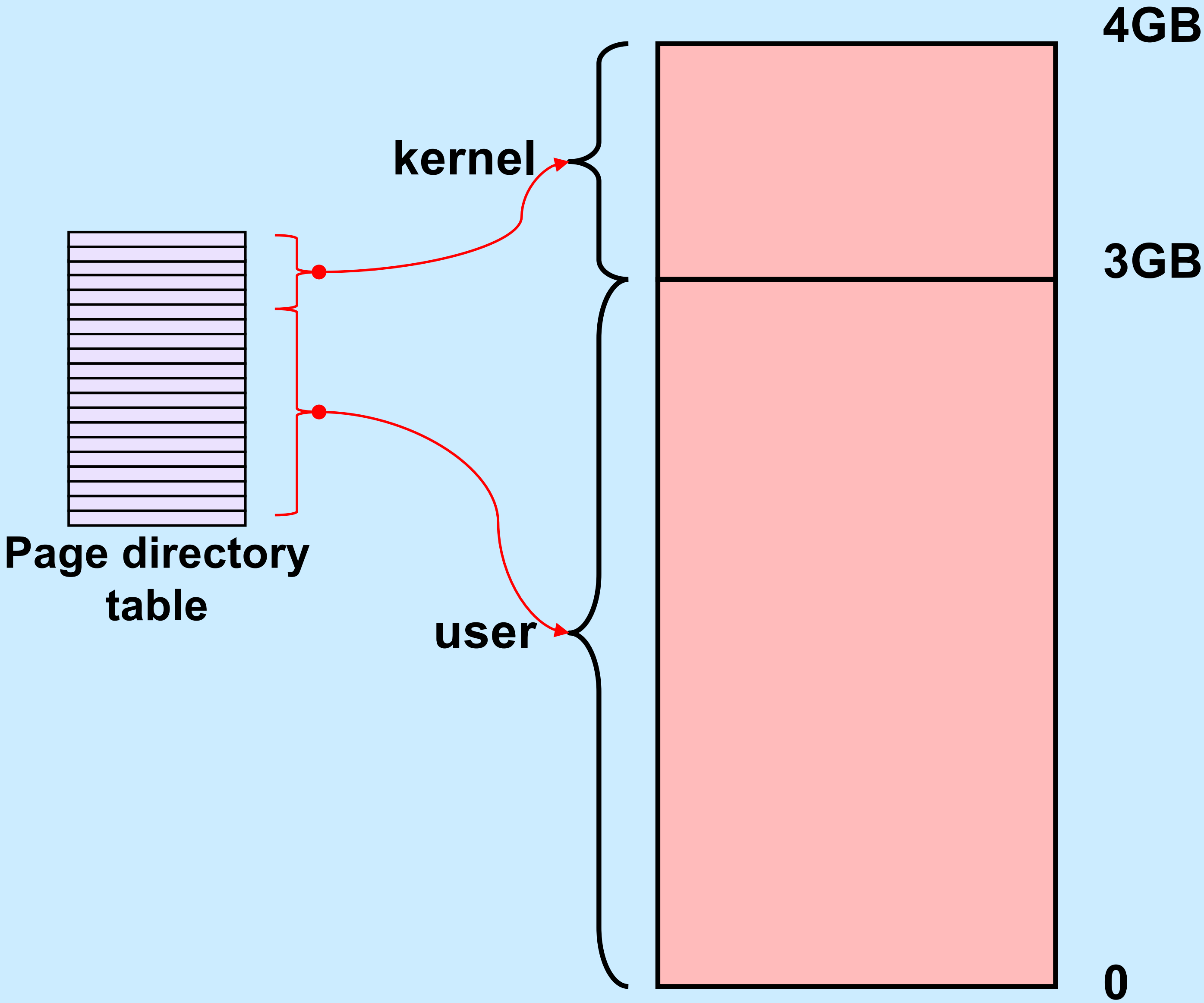

10 bits
10 bits
12 bits
CR3
Page directory table
Page table
Page

Quiz 2

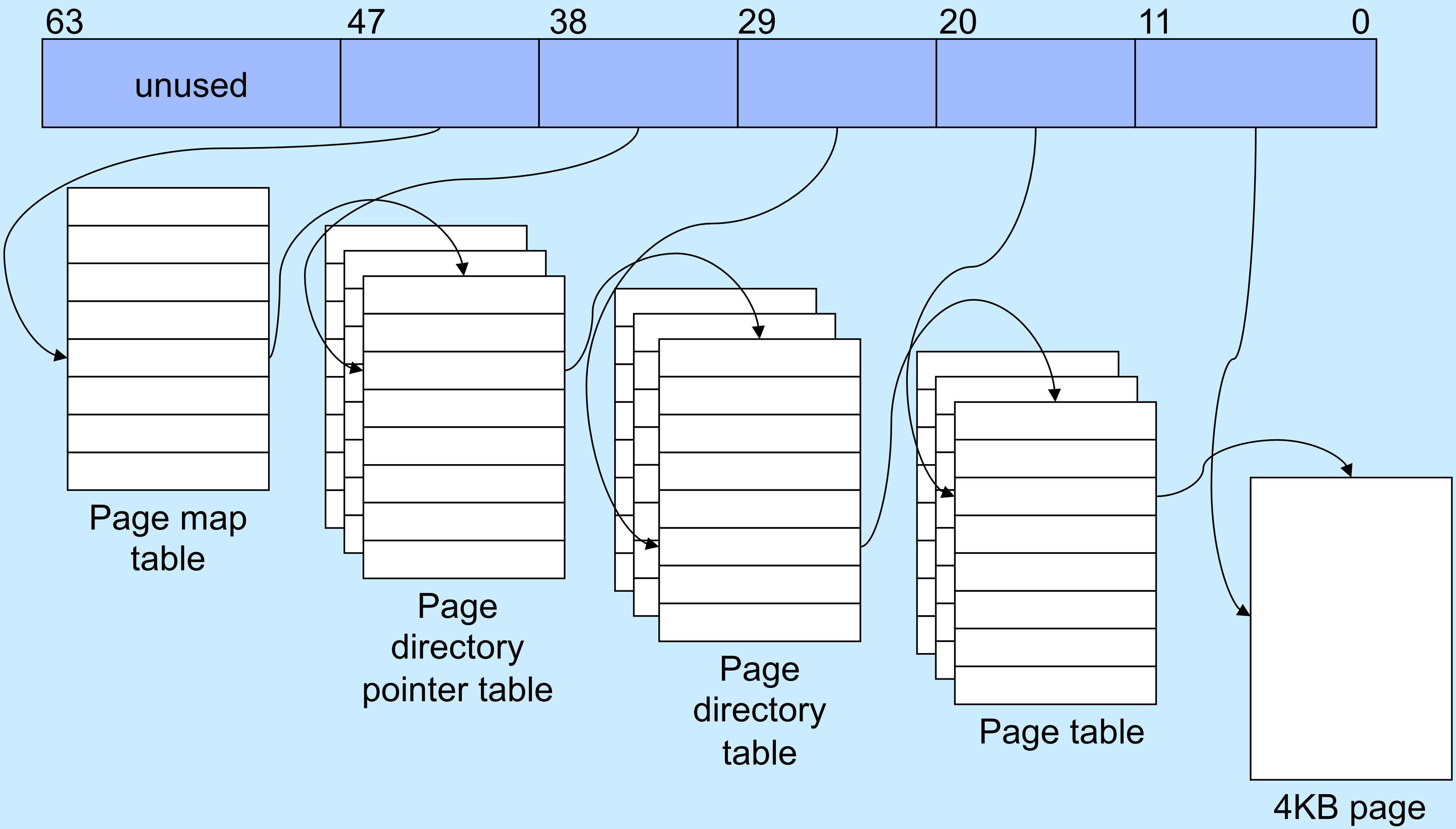
Can a page start at a virtual address that's not divisible by the page size?

a) yes
b) no

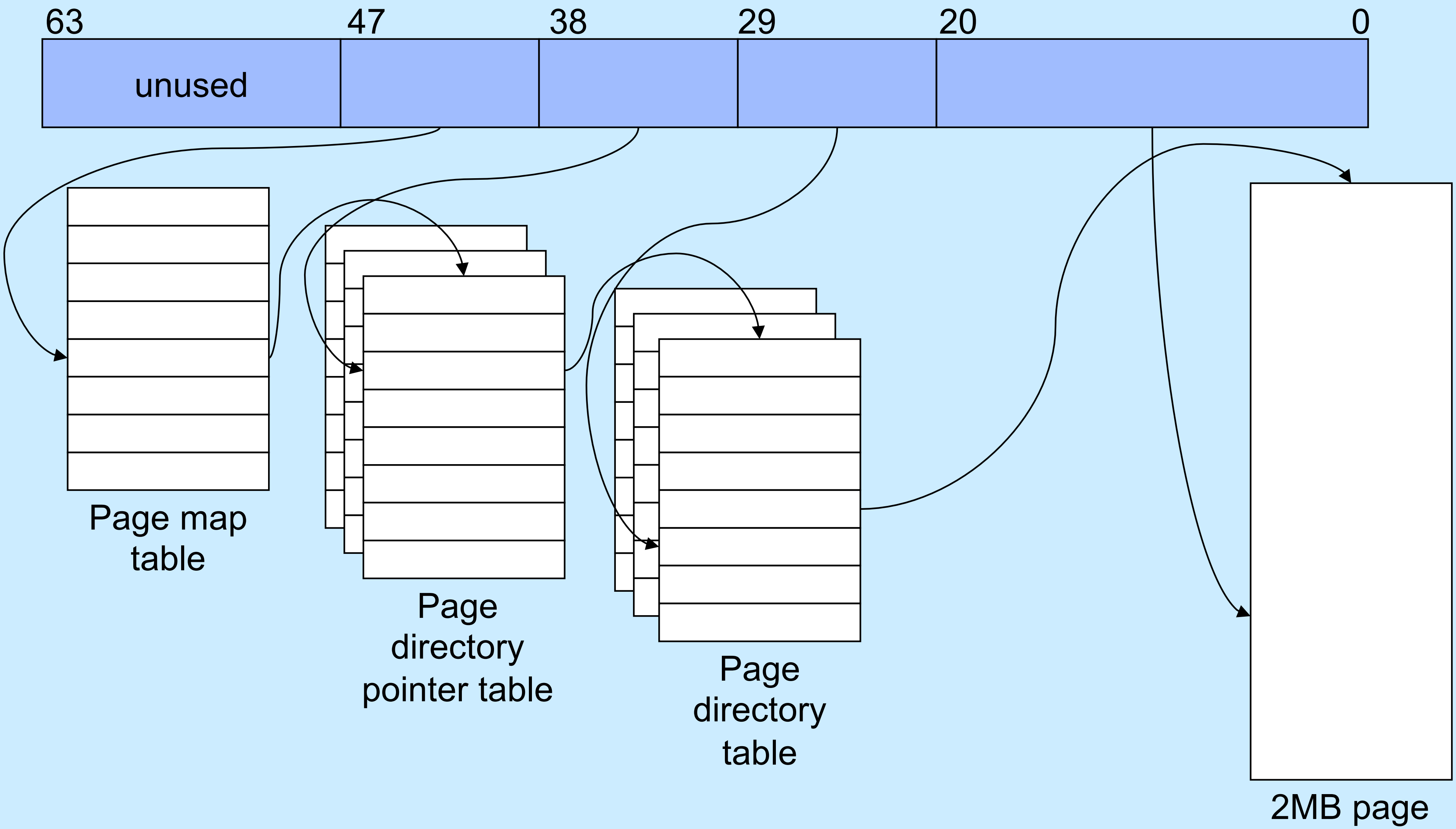
Linux Intel IA32 VM Layout



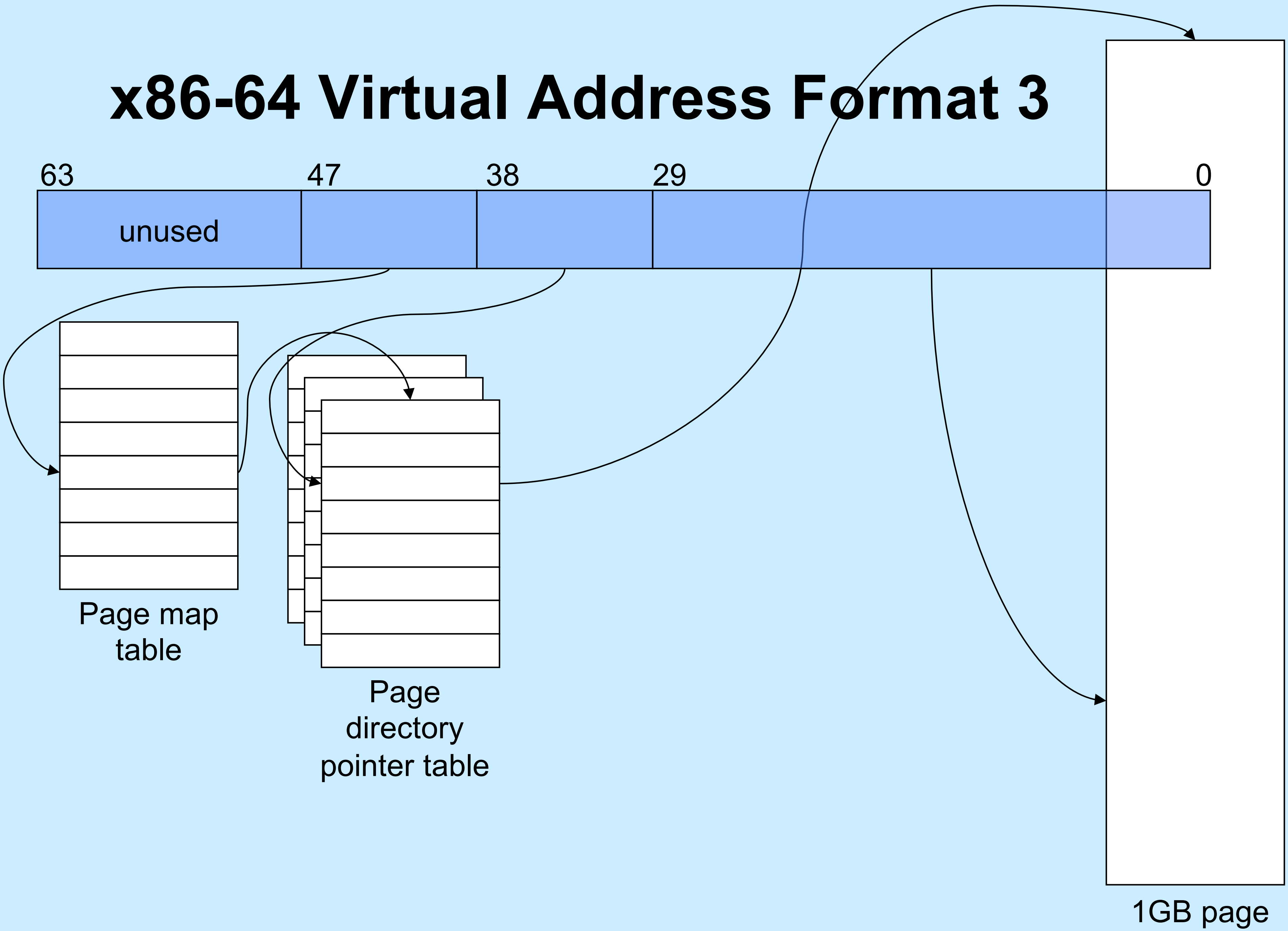
x86-64 Virtual Address Format 1



x86-64 Virtual Address Format 2



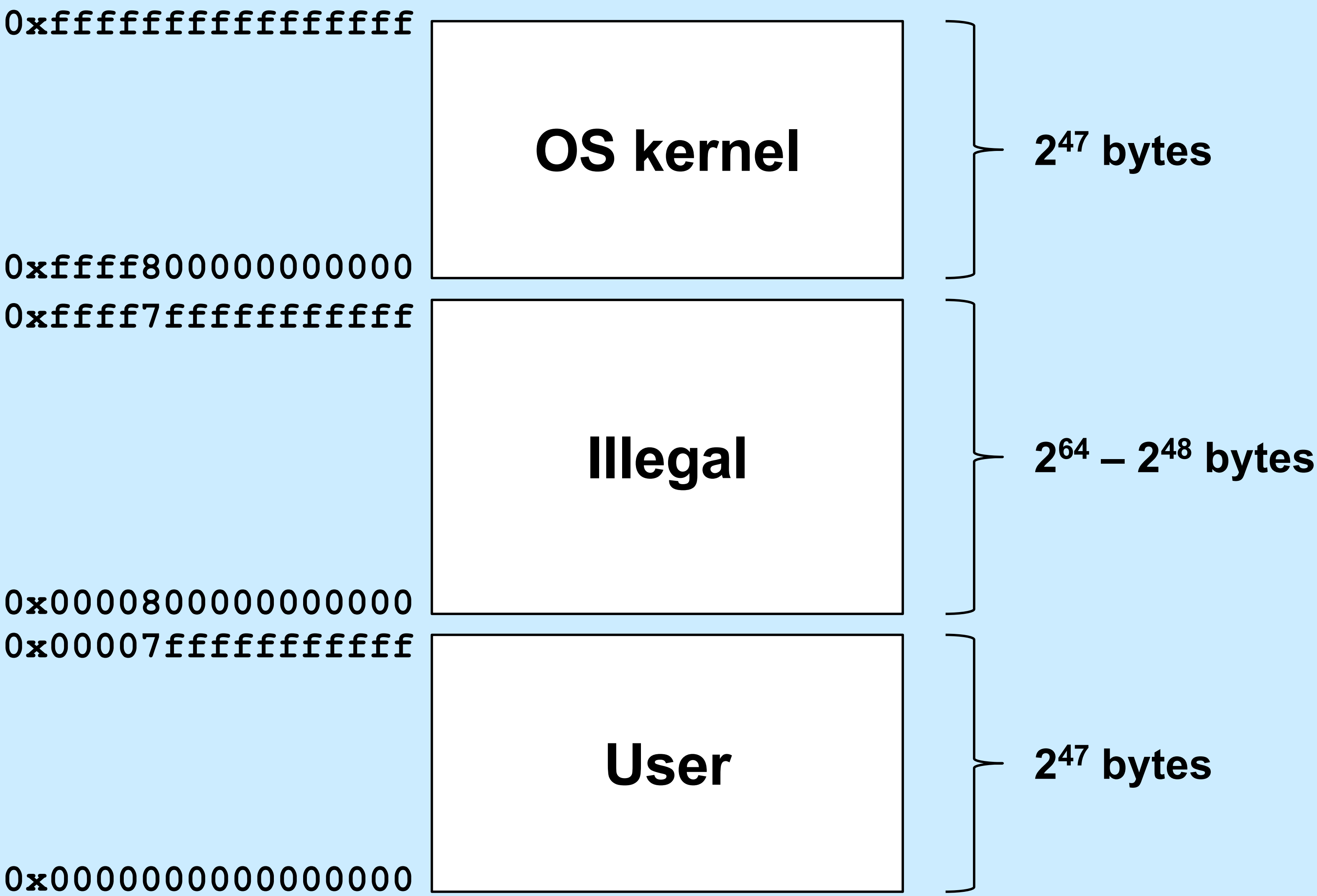
x86-64 Virtual Address Format 3



Why Multiple Page Sizes?

- **Fragmentation**
  - for region composed of 4KB pages, average internal fragmentation is 2KB
  - for region composed of 1GB pages, average internal fragmentation is 512MB
- **Page-table overhead**
  - larger page sizes have fewer page tables
    - » less overhead in representing mappings

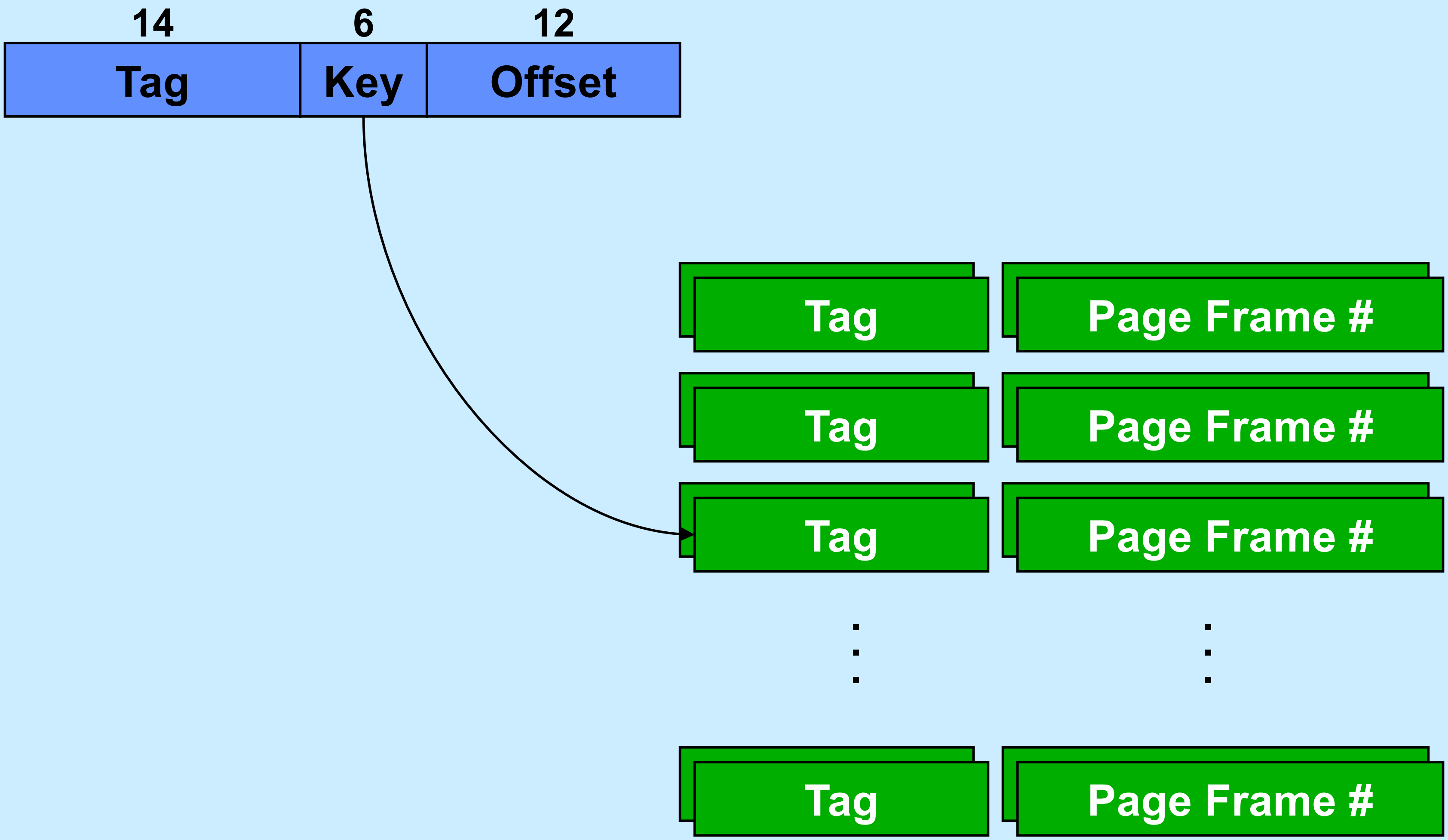
x86-64 Address Space



Performance

- **Page table resides in real memory (DRAM)**
- **A 32-bit virtual-to-real translation requires two accesses to page tables, plus the access to the ultimate real address**
  - three real accesses for each virtual access
  - 3X slowdown!
- **A 64-bit virtual-to-real translation requires four accesses to page tables, plus the access to the ultimate real address**
  - 5X slowdown!

Translation Lookaside Buffers



Quiz 3

Recall that there is a 5x slowdown on memory references via virtual memory on the x86-64. If all references are translated via the TLB, the slowdown will be

a) 1x

b) 2x

c) 3x

d) 4x

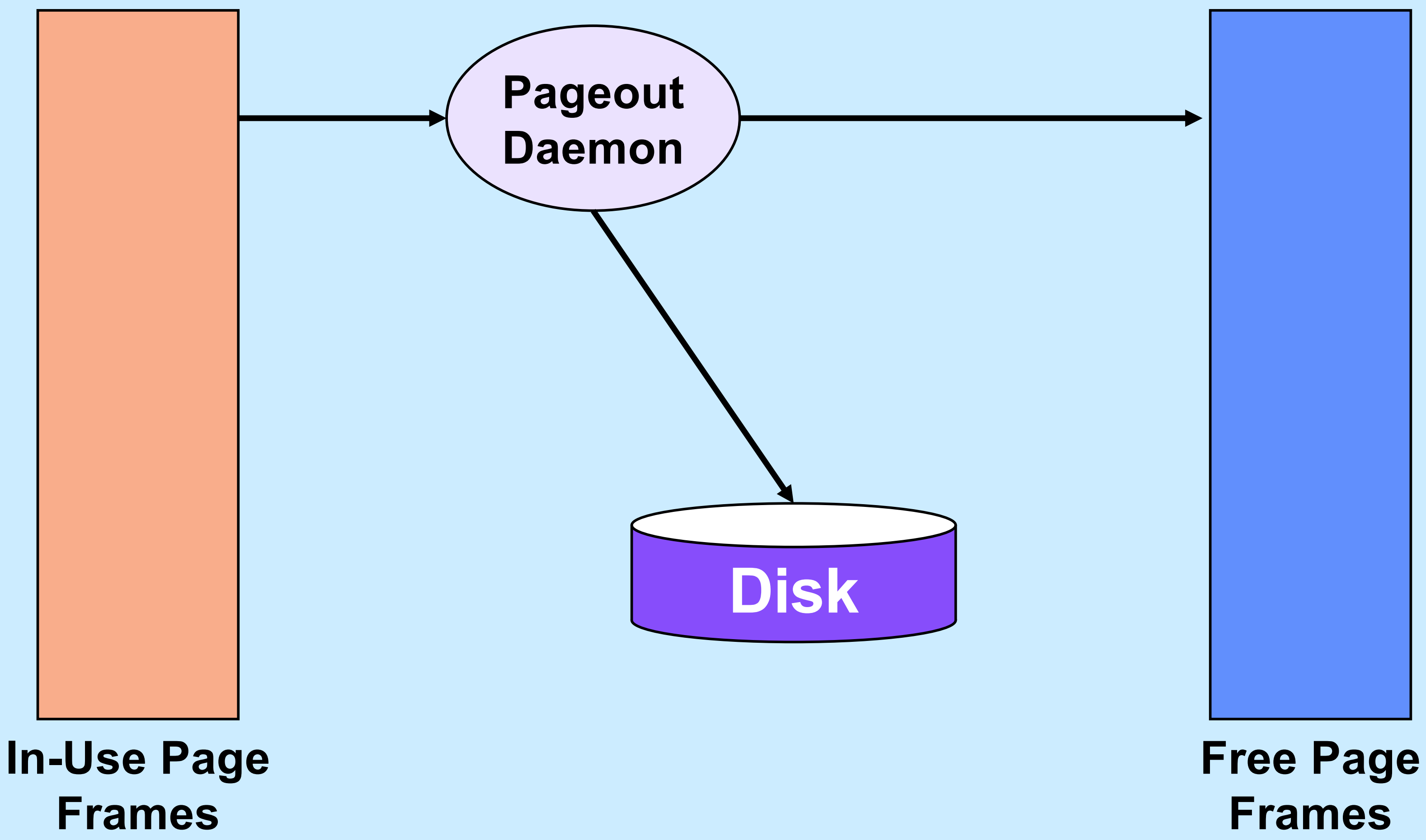
OS Role in Virtual Memory

- **Memory is like a cache**
  - quick access if what's wanted is mapped via page table
  - slow if not — OS assistance required
- **OS**
  - make sure what's needed is mapped in
  - make sure what's no longer needed is not mapped in

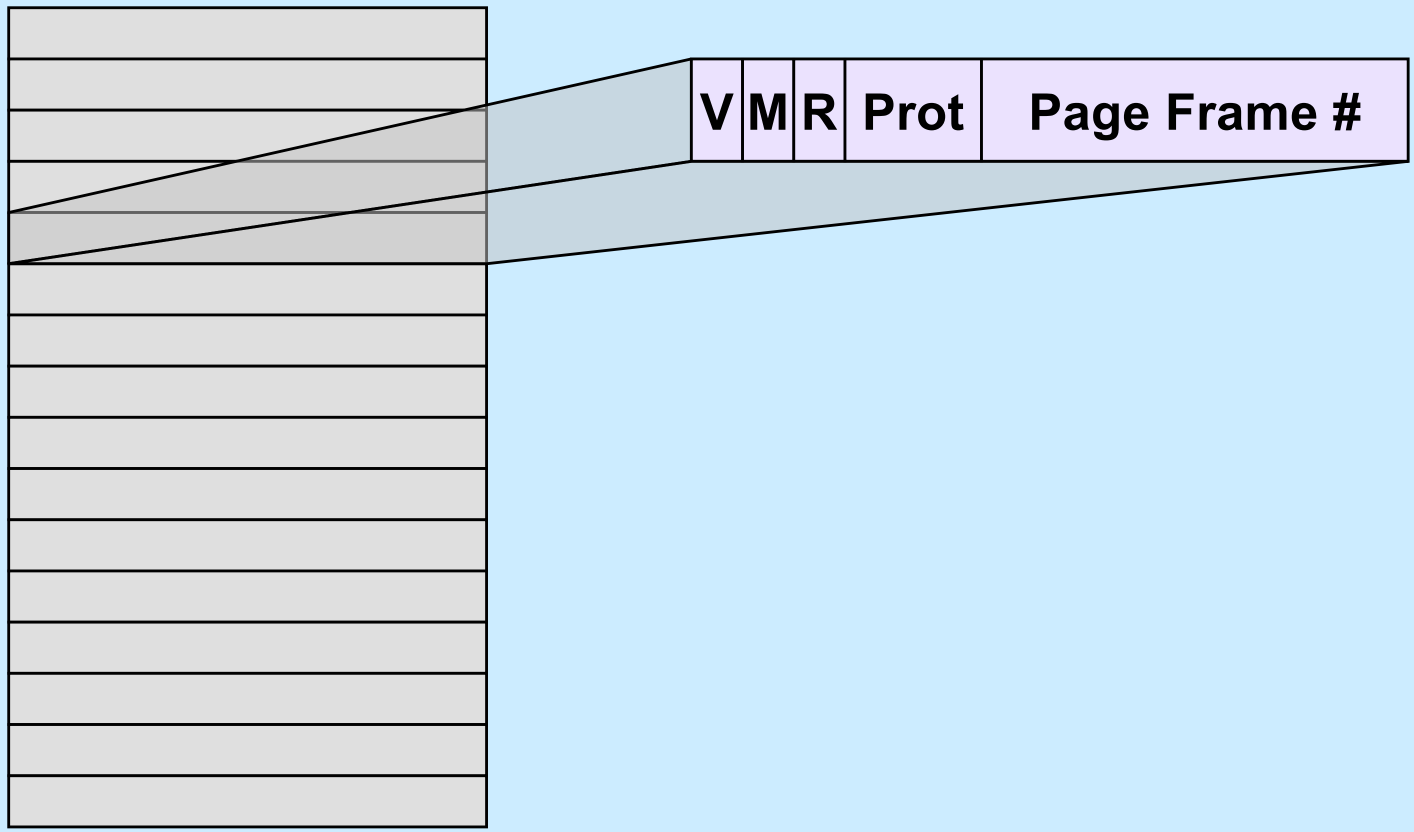
Mechanism

- **Program references memory**
  - if reference is mapped, access is quick
    - » even quicker if translation in TLB and referent in on-chip cache
  - if not, page-translation fault occurs and OS is invoked
    - » determines desired page
    - » maps it in, if legal reference

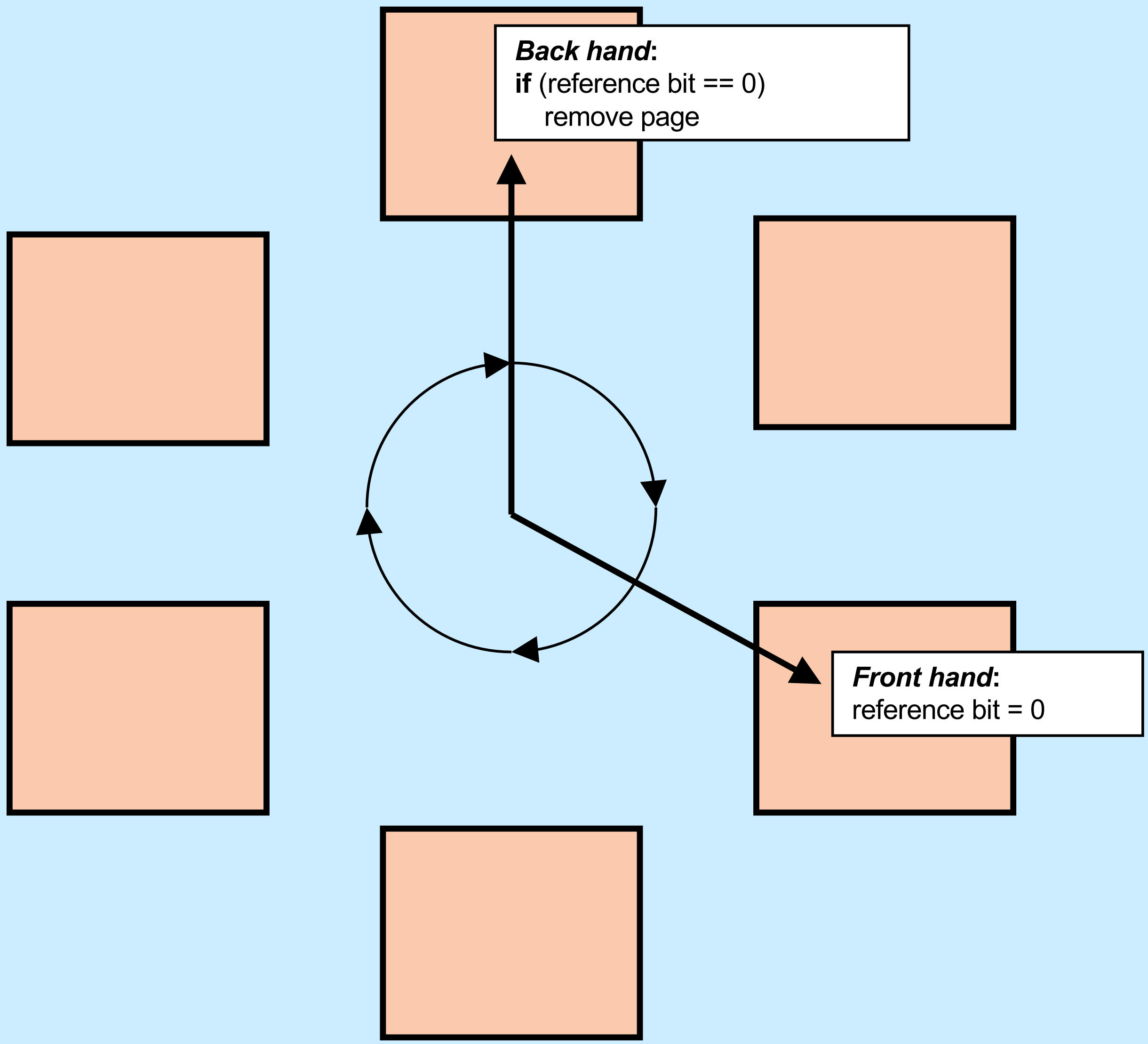
The "Pageout Daemon"



Managing Page Frames

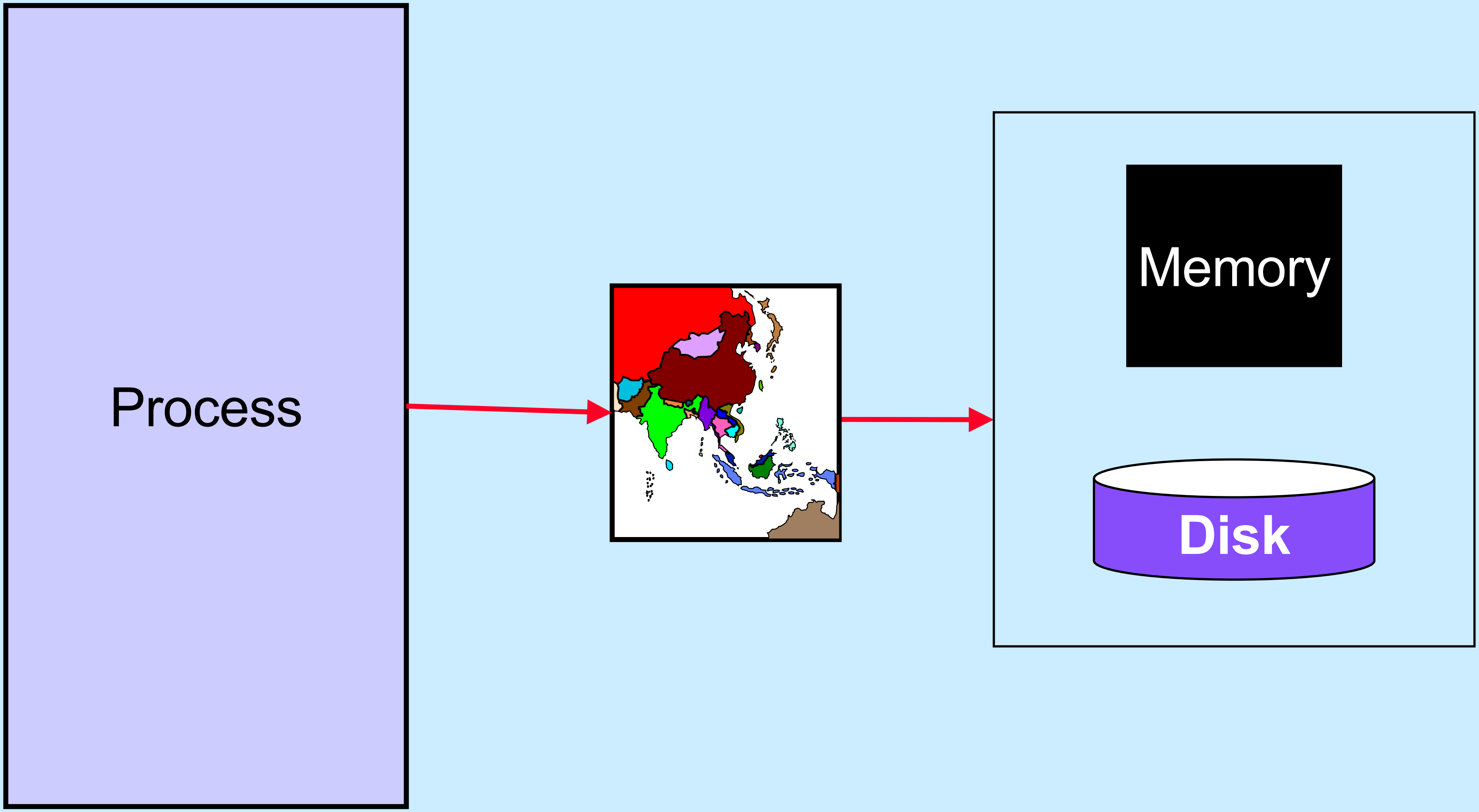


Clock Algorithm

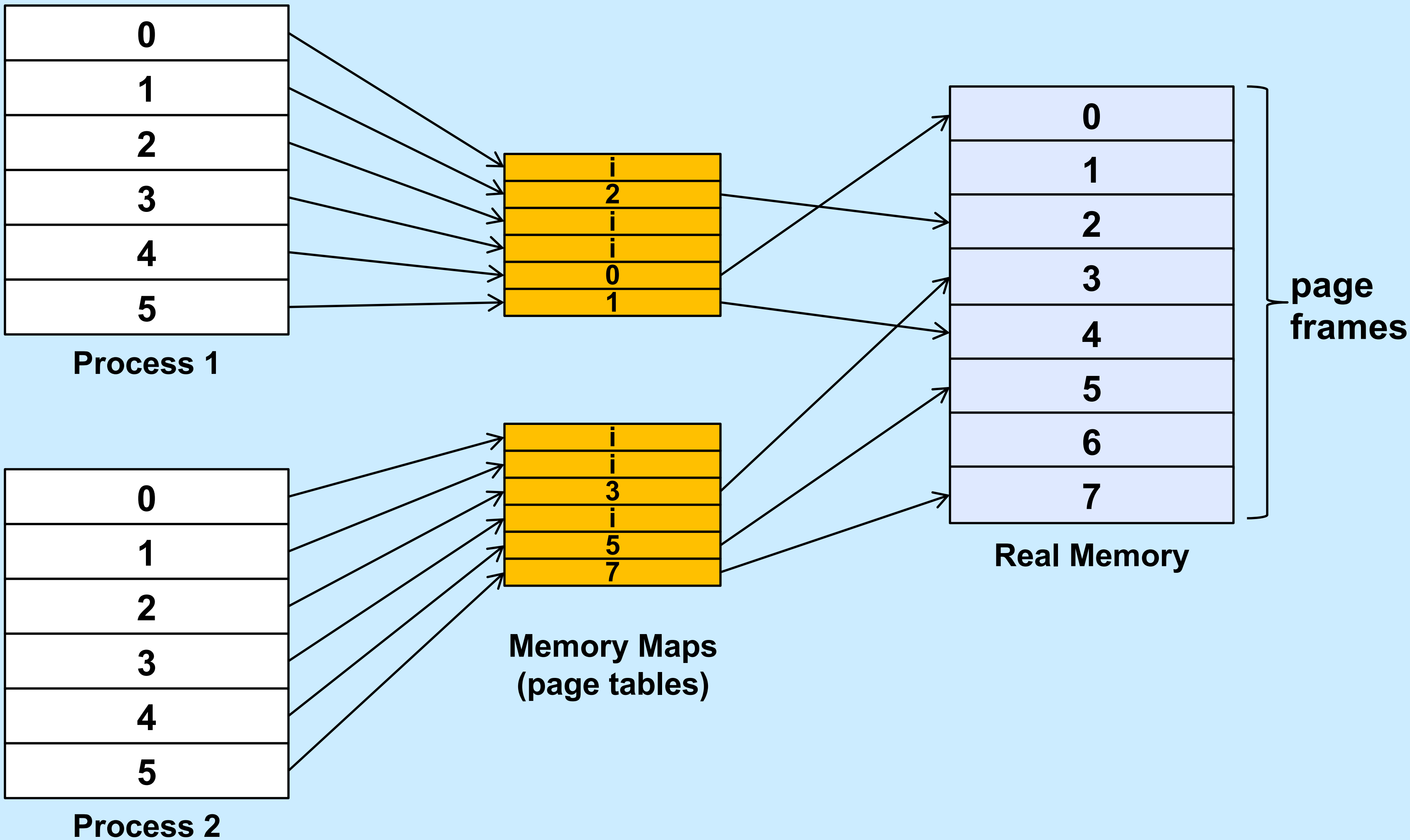


Why is virtual memory used?

More VM than RM

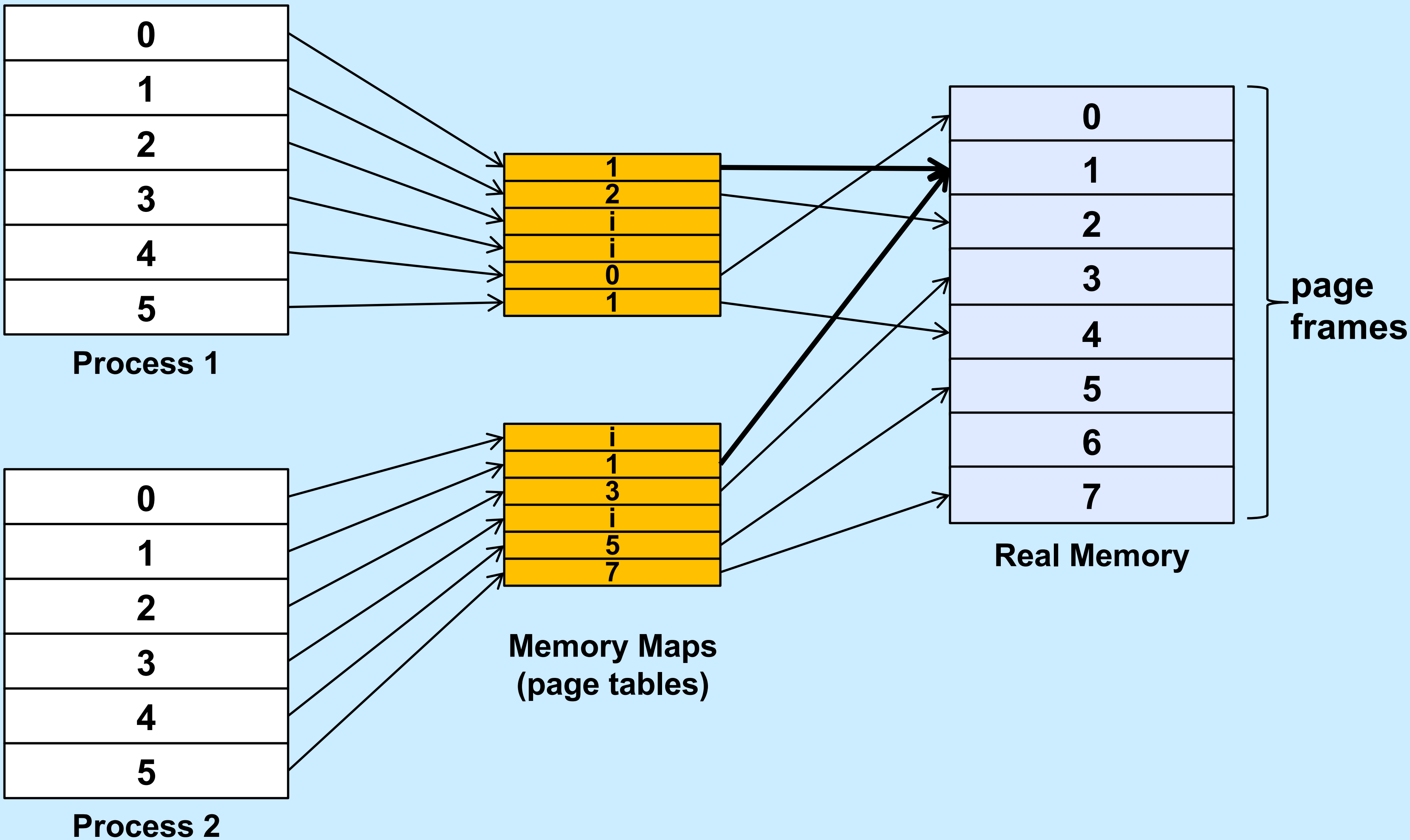


Isolation



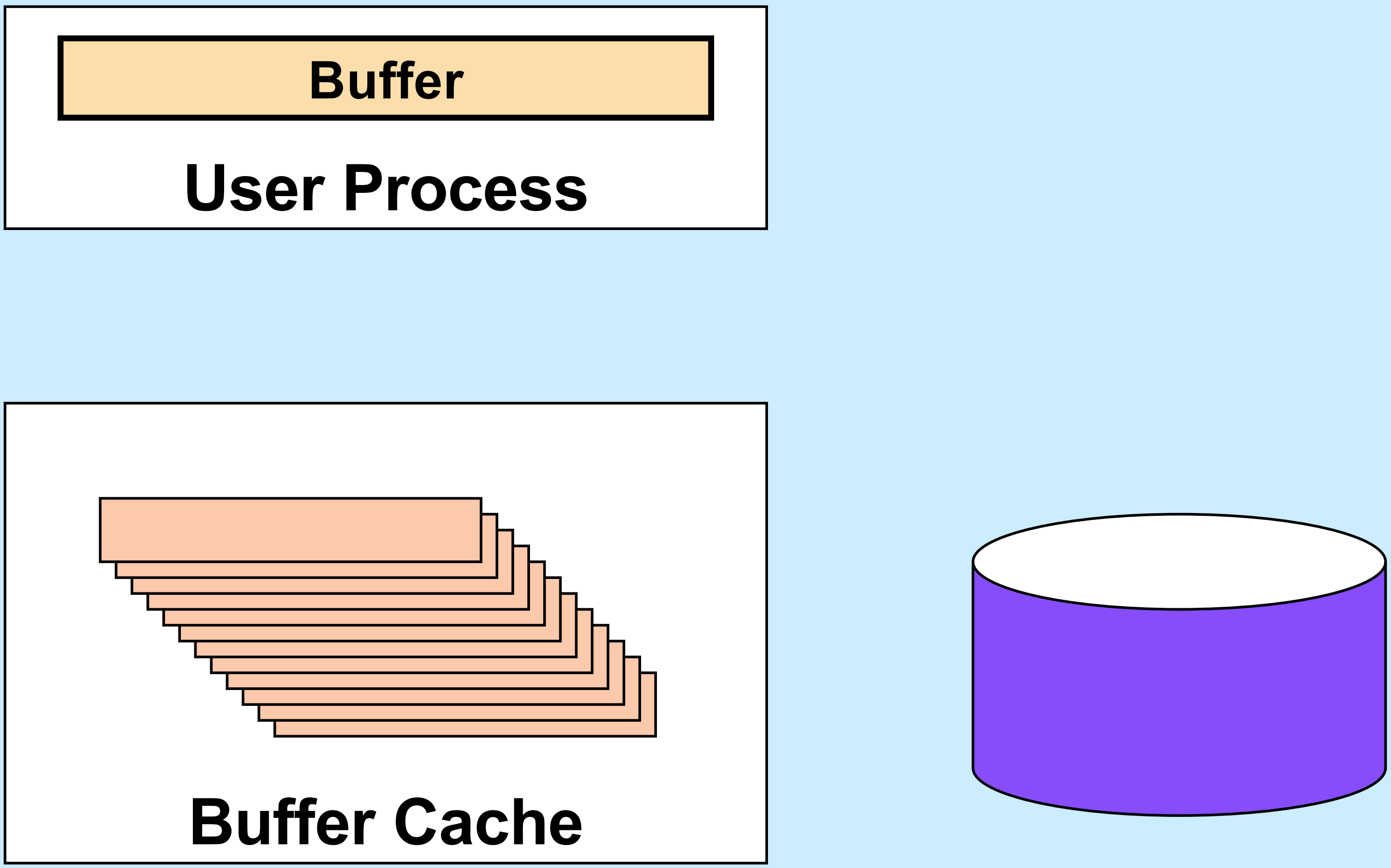
0
1
2
3
4
5
Process 1
i
2
i
i
0
1
0
1
2
3
4
5
6
7
page frames
0
1
2
3
4
5
Process 2
i
i
3
i
5
7
Memory Maps (page tables)
Real Memory

Sharing

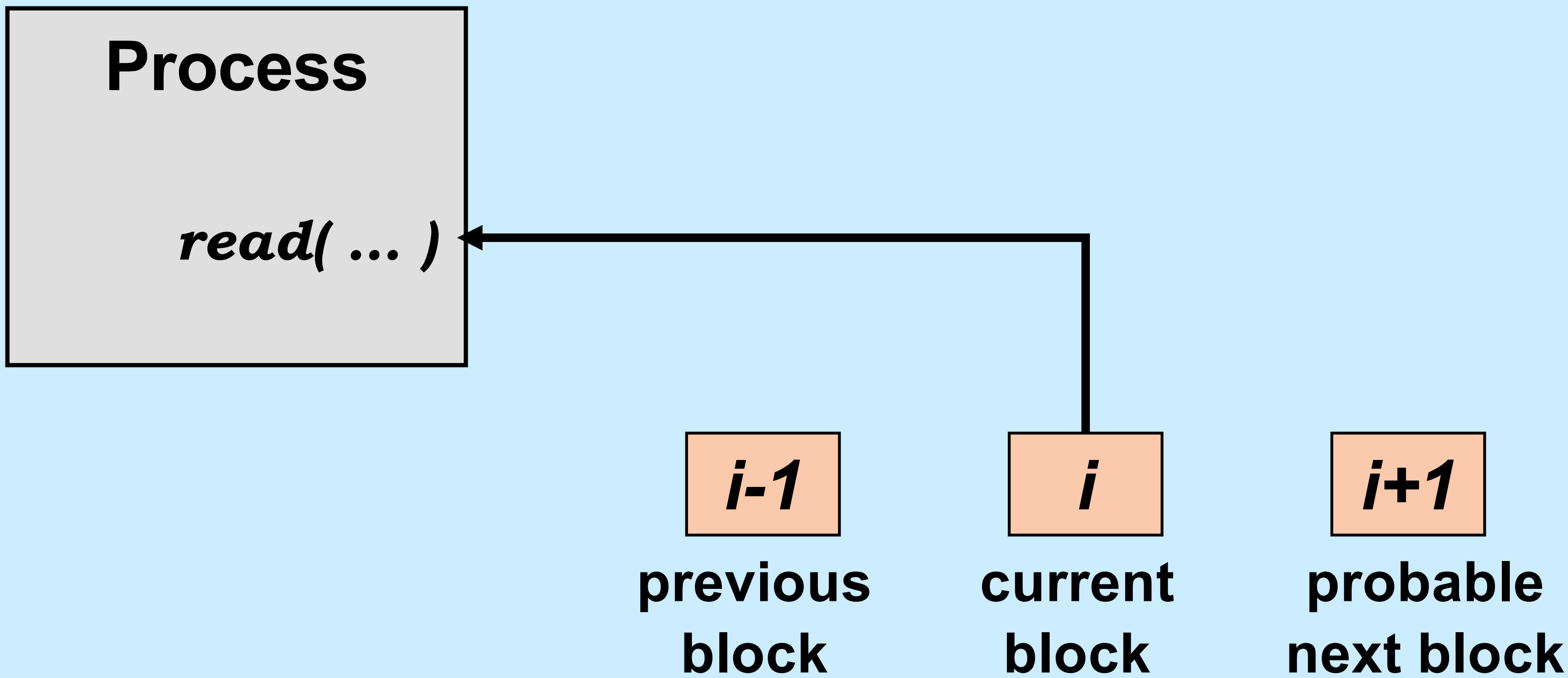


0
1
2
3
4
5
Process 1
1
2
i
i
0
1
i
1
3
i
5
7
Memory Maps
(page tables)
0
1
2
3
4
5
Process 2
0
1
2
3
4
5
6
7
Real Memory
page frames

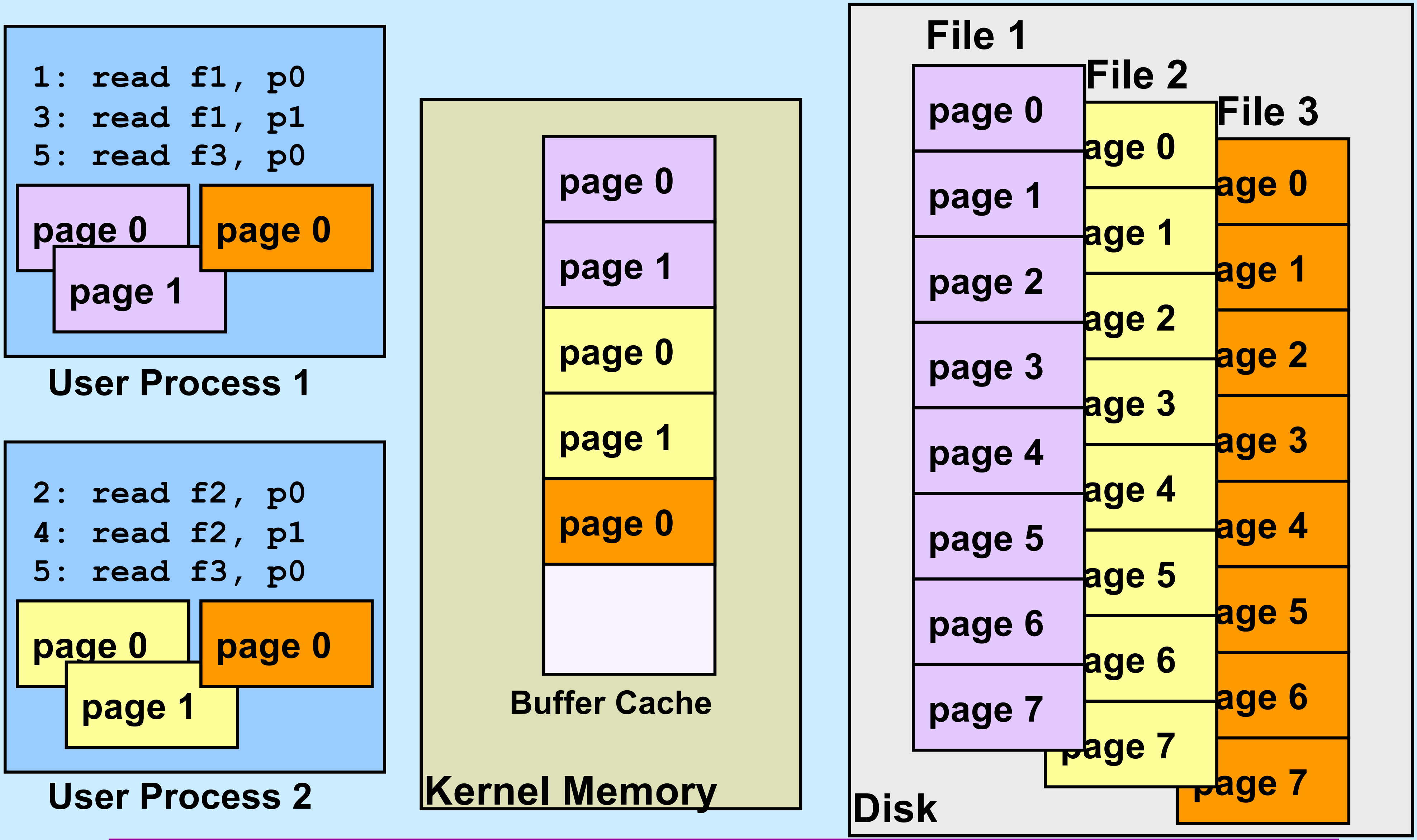
File I/O



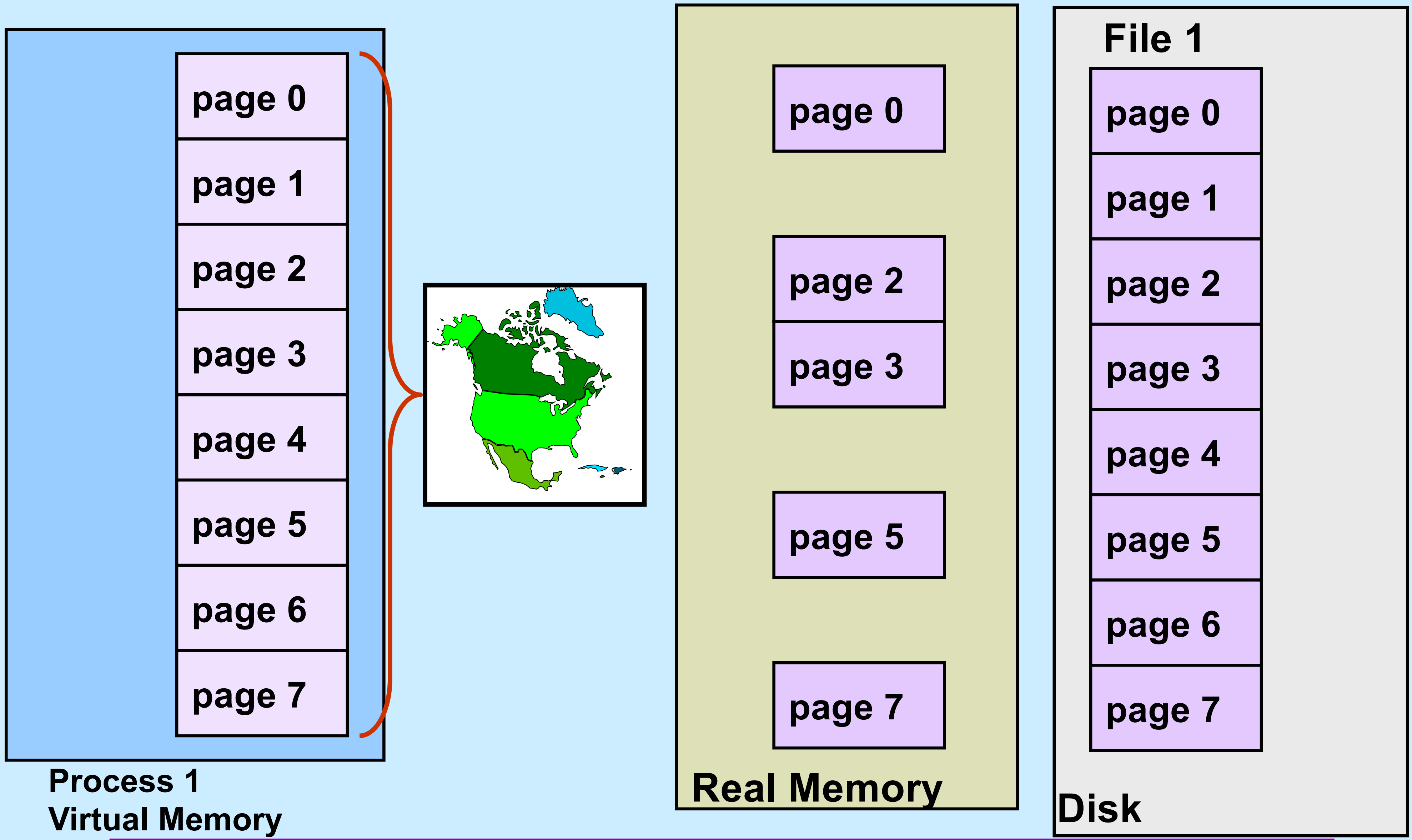
Multi-Buffered I/O



Traditional I/O



Mapped File I/O



Multi-Process Mapped File I/O