Memory

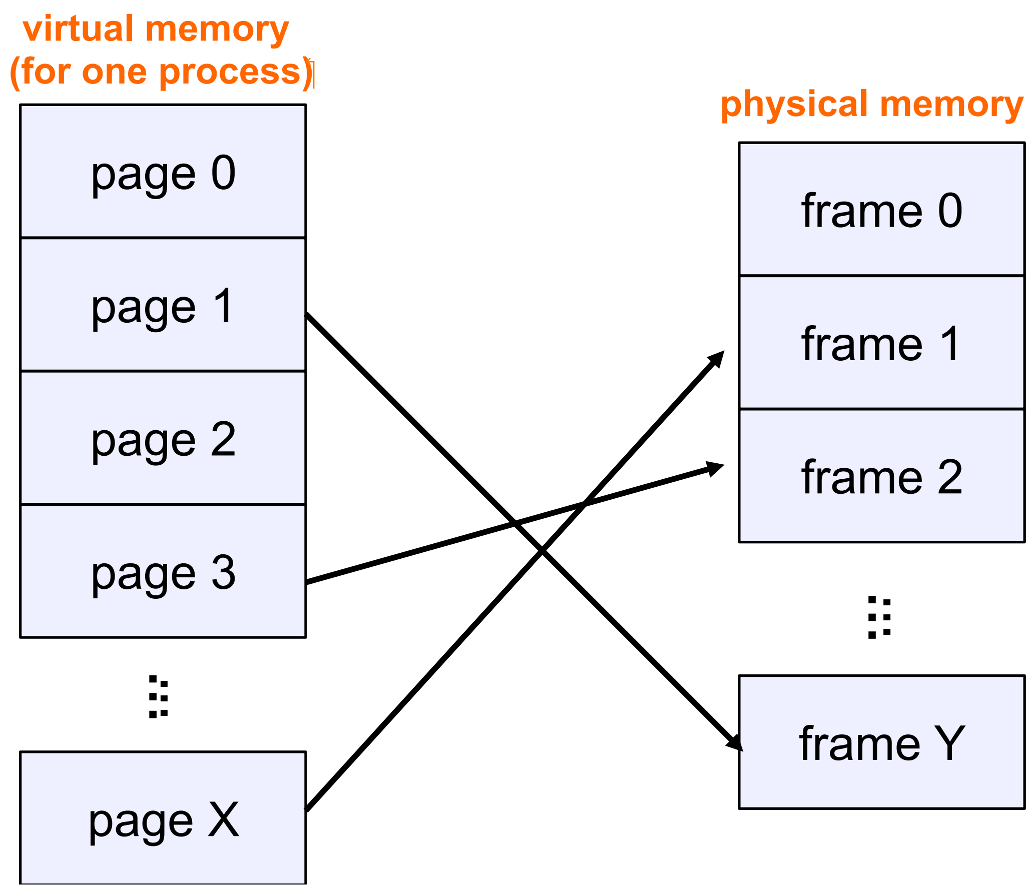
CS 241

February 1, 2012

Slides adapted in part from material by
Matt Welsh, Harvard U.

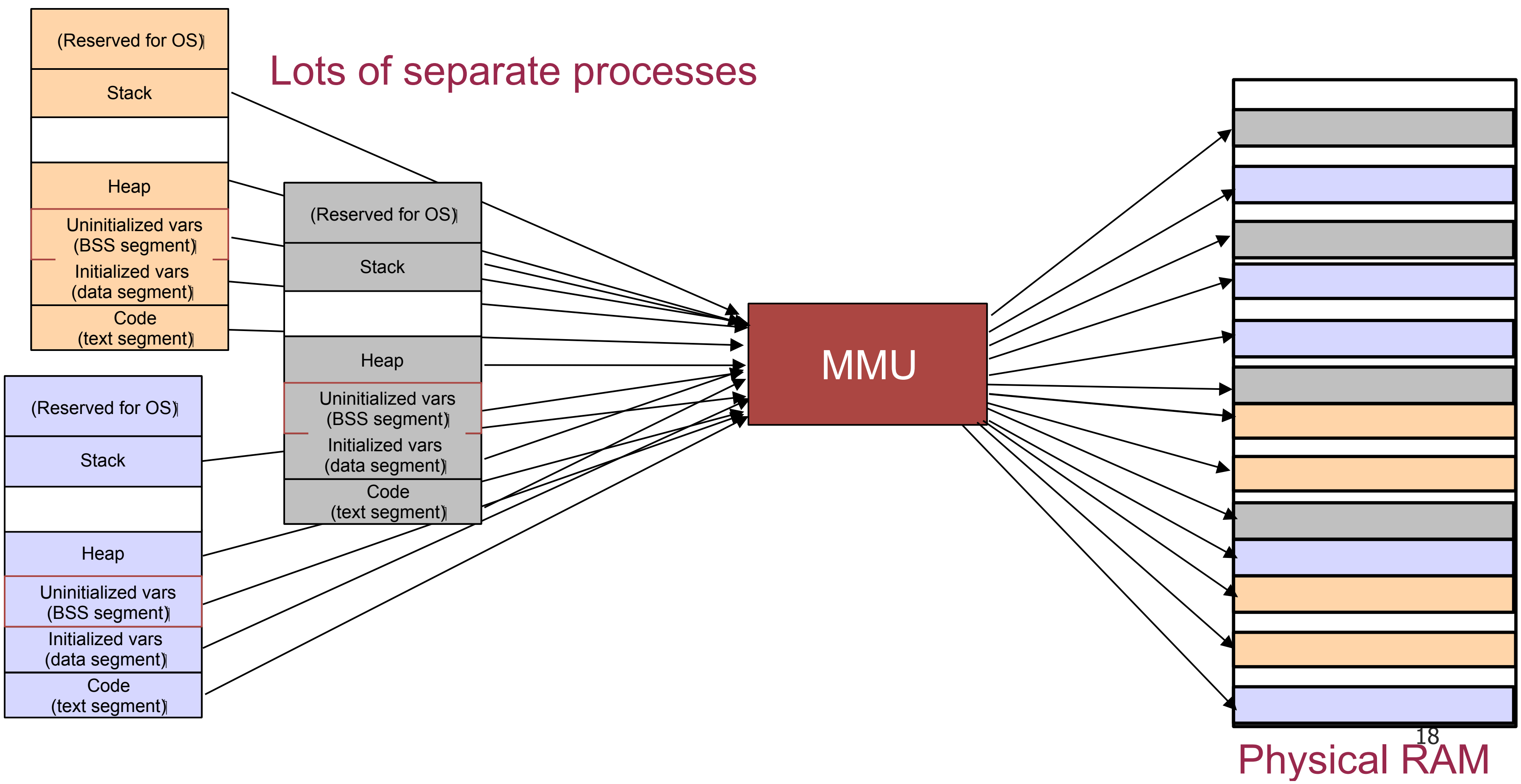
Paging

- Solve the external fragmentation problem by using **fixed-size chunks** of virtual and physical memory
  - Virtual memory unit called a **page**
  - Physical memory unit called a **frame** (or sometimes **page frame**)



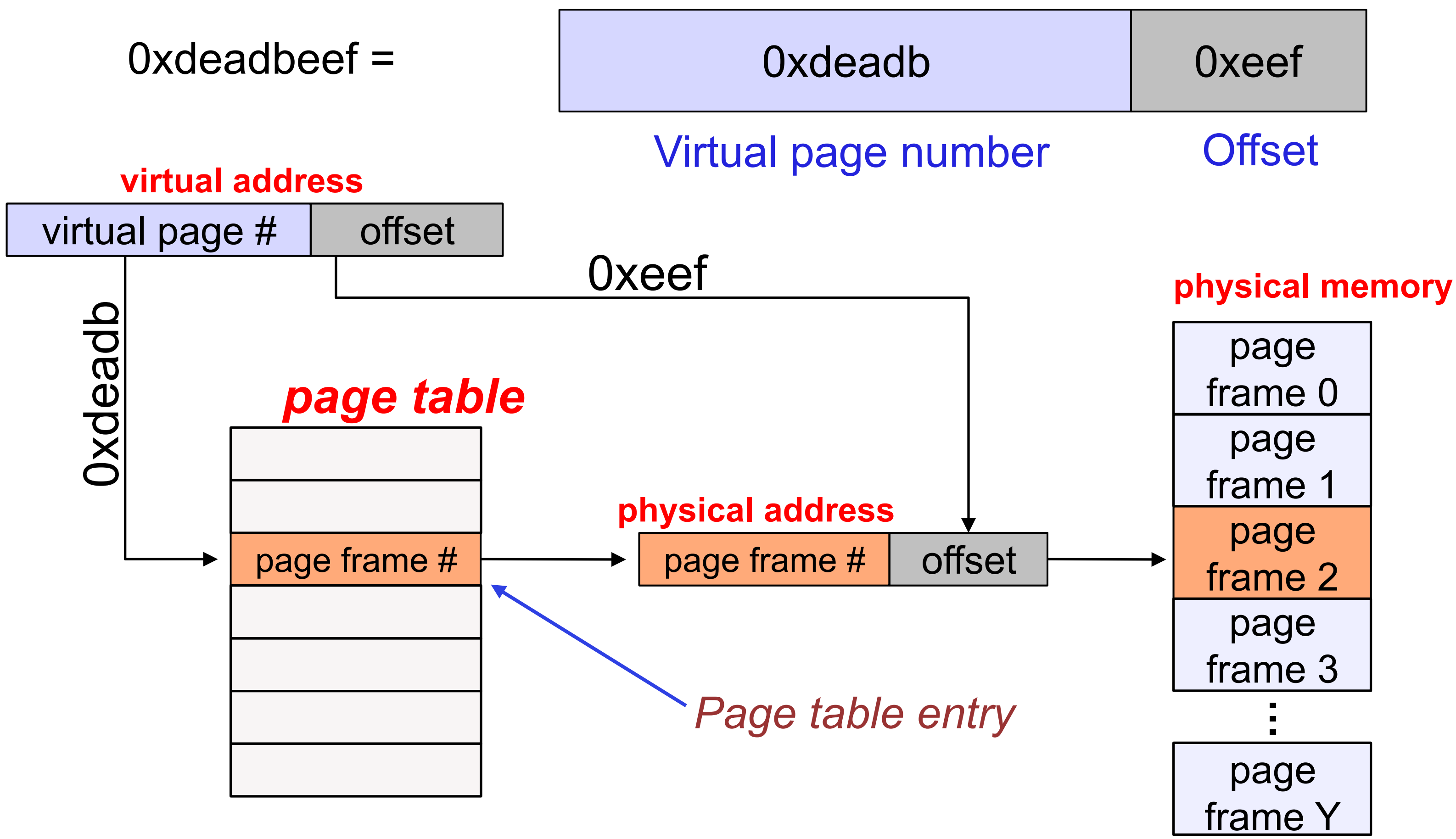
Application Perspective

- Application believes it has a single, contiguous address space ranging from 0 to 2P – 1 bytes
  - Where P is the number of bits in a pointer (e.g., 32 bits)
- In reality, virtual pages are scattered across physical memory
  - This mapping is invisible to the program, and not even under it's control!



Translation process

- Virtual-to-physical address translation performed by MMU
  - Virtual address is broken into a *virtual page number* and an *offset*
  - Mapping from virtual page to physical frame provided by a *page table* (which is stored in memory)



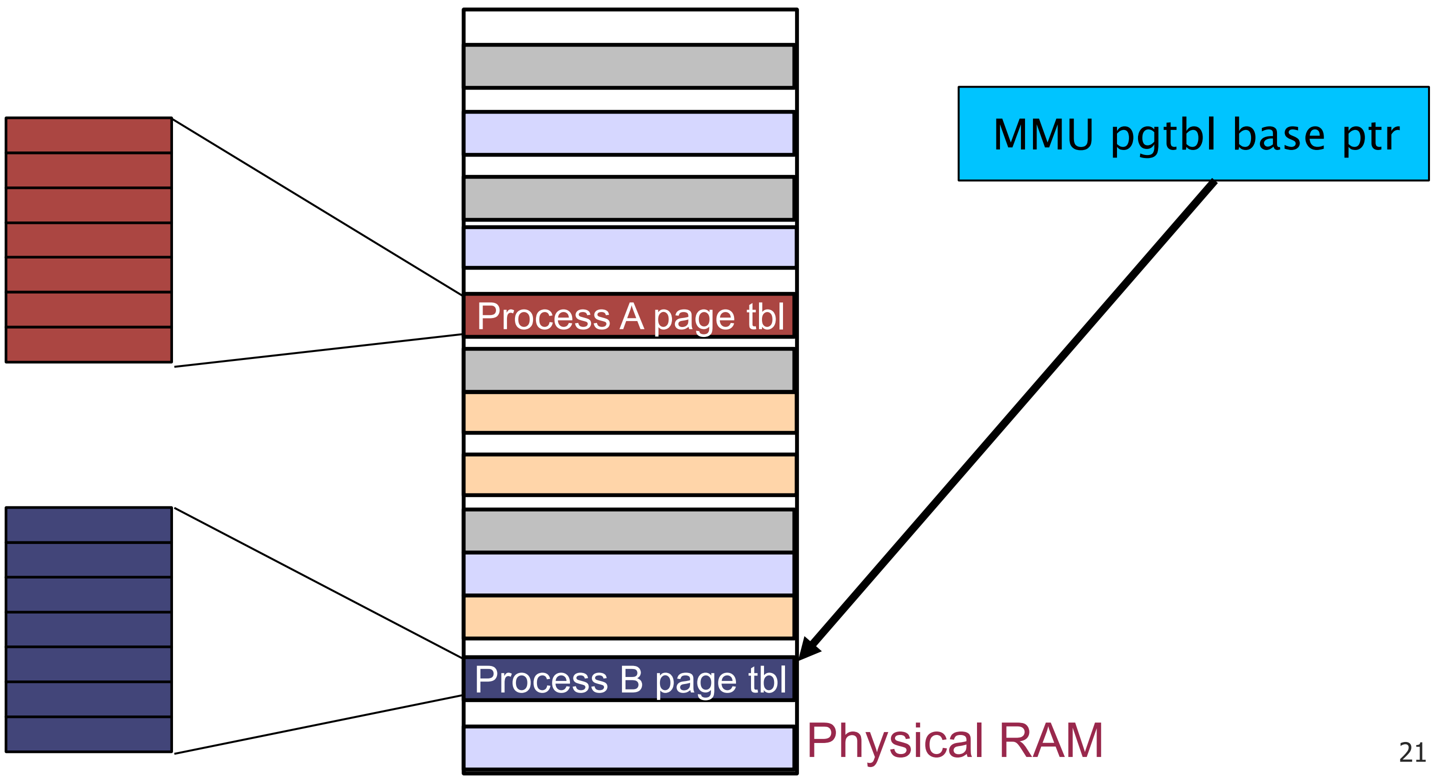
Translation process

```
if (virtual page is invalid or non-resident or protected)
    trap to OS fault handler
else
    physical frame # = pageTable[virtpage#].physPageNum
```

- Each virtual page can be in physical memory or swapped out to disk (called “paged out” or just “paged”)
- What must change on a context switch?
  - Could copy entire contents of table, but this will be slow
  - Instead use an extra layer of indirection: Keep pointer to current page table and just change pointer

Where is the page table?

- Page Tables store the virtual-to-physical address mappings.
- Where are they located? *In memory!*
- OK, then. How does the MMU access them?
  - The MMU has a special register called the *page table base pointer*.
  - This points to the *physical memory address* of the top of the page table for the currently-running process.



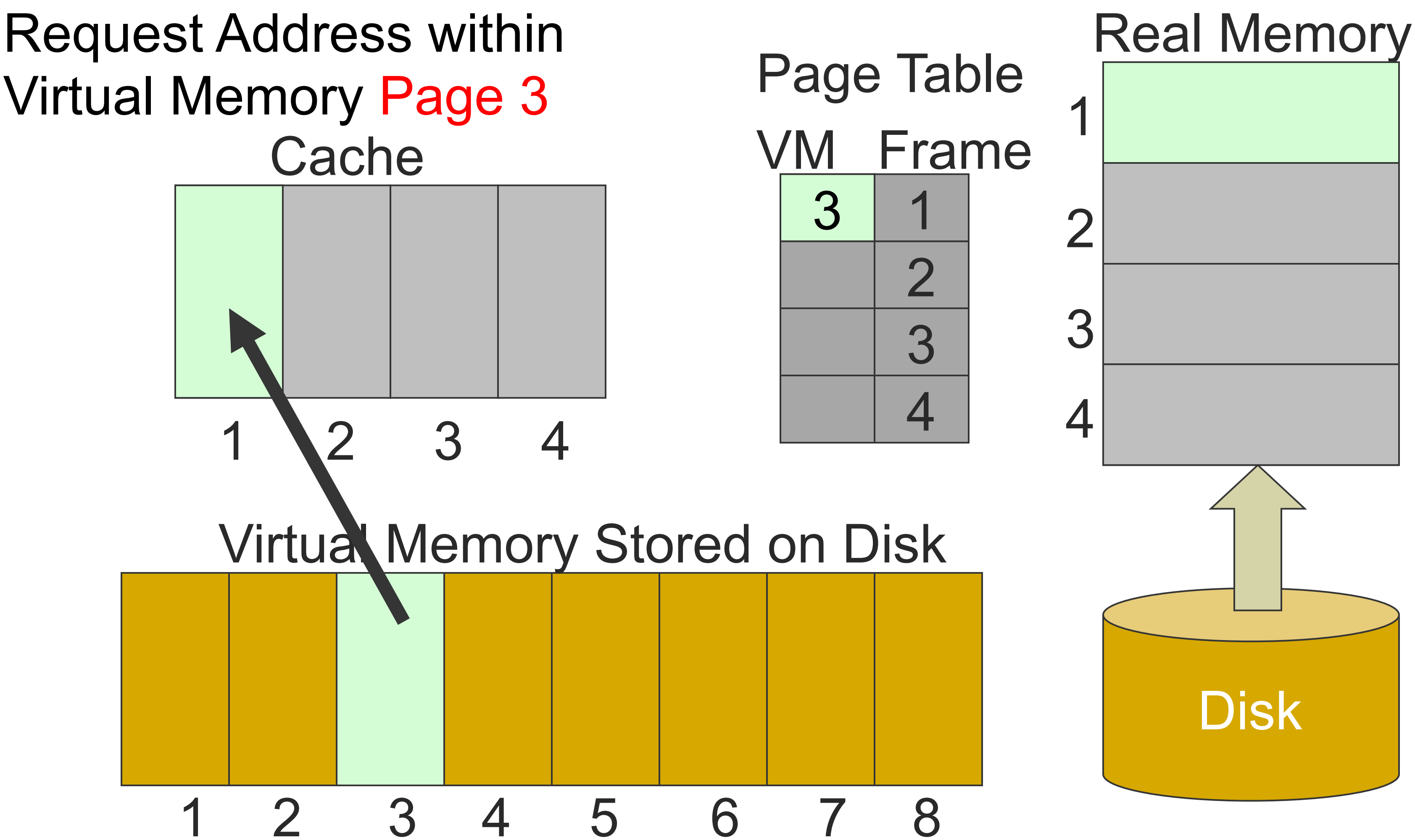
Page Faults

- What happens when a program accesses a virtual page that is not mapped into any physical page?
  - Hardware triggers a page fault
- Page fault handler
  - Find any available free physical page
  - If none, evict some resident page to disk
  - Allocate a free physical page
  - Load the faulted virtual page to the prepared physical page
  - Modify the page table

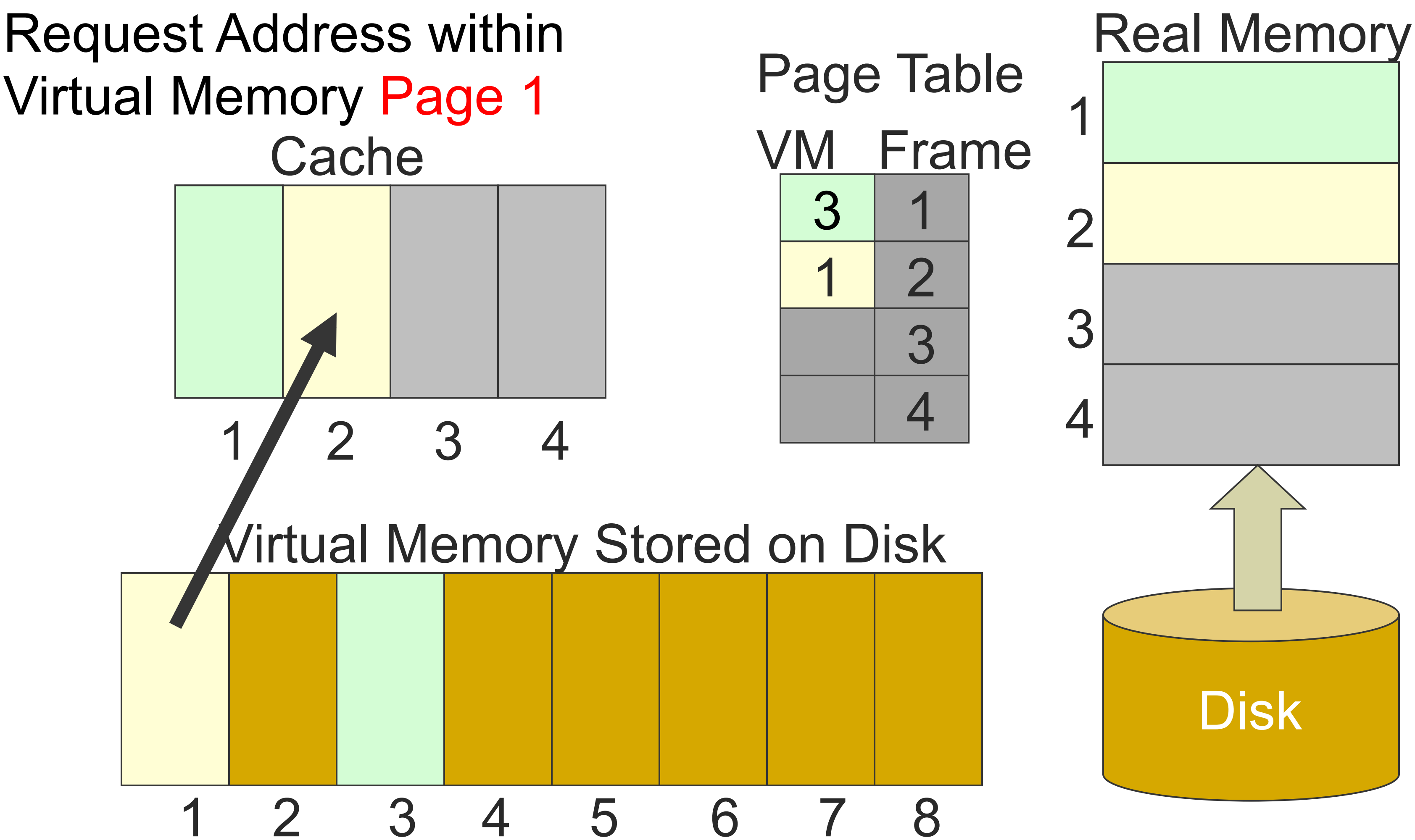
Advantages of Paging

- Simplifies physical memory management
  - OS maintains a free list of physical page frames
  - To allocate a physical page, just remove an entry from this list
- No external fragmentation!
  - Virtual pages from different processes can be interspersed in physical memory
  - No need to allocate pages in a contiguous fashion
- Allocation of memory can be performed at a (relatively) fine granularity
  - Only allocate physical memory to those parts of the address space that require it
  - Can swap unused pages out to disk when physical memory is running low
  - Idle programs won't use up a lot of memory (even if their address space is huge!)

Paging Example

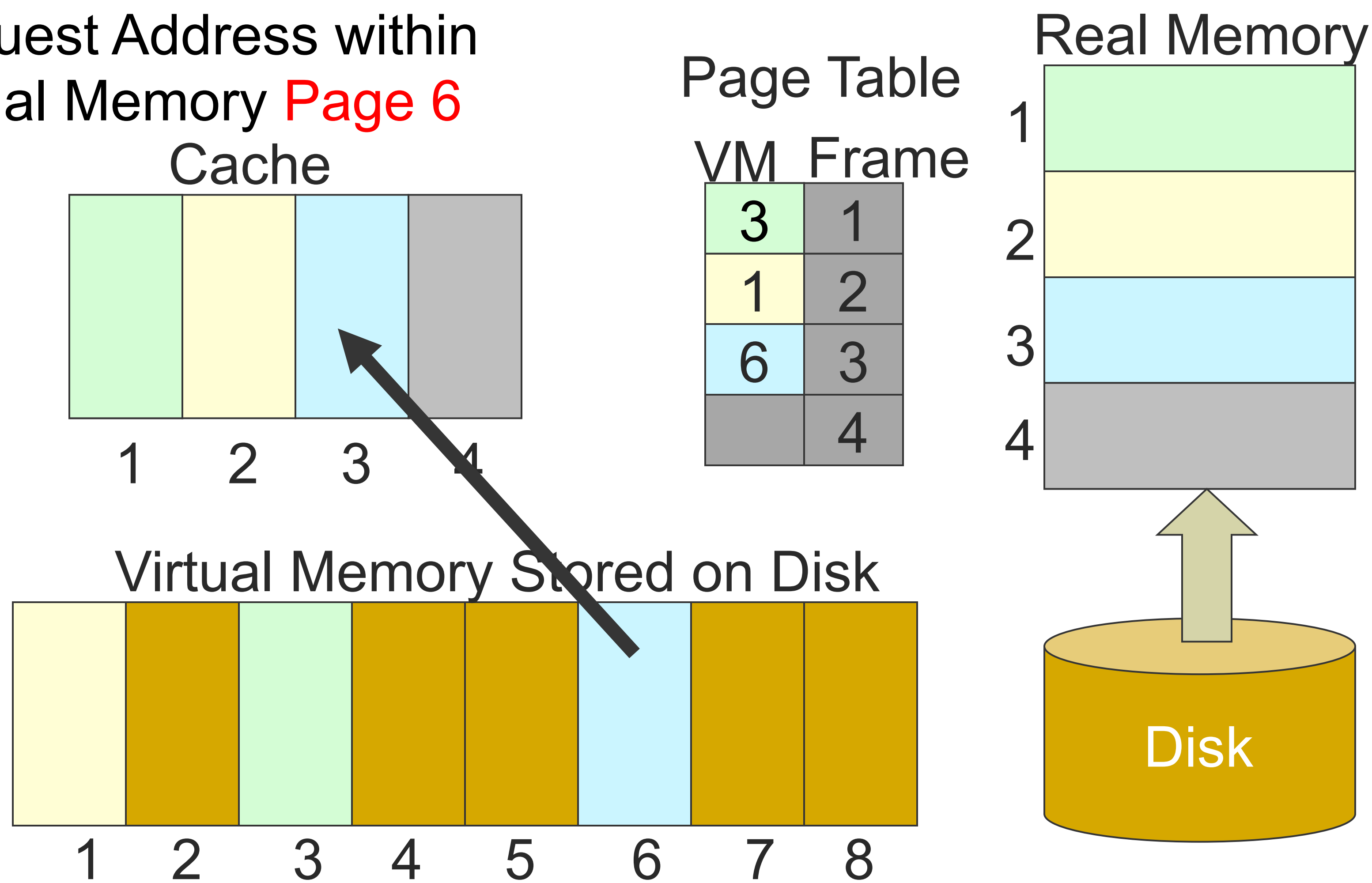


Paging Example



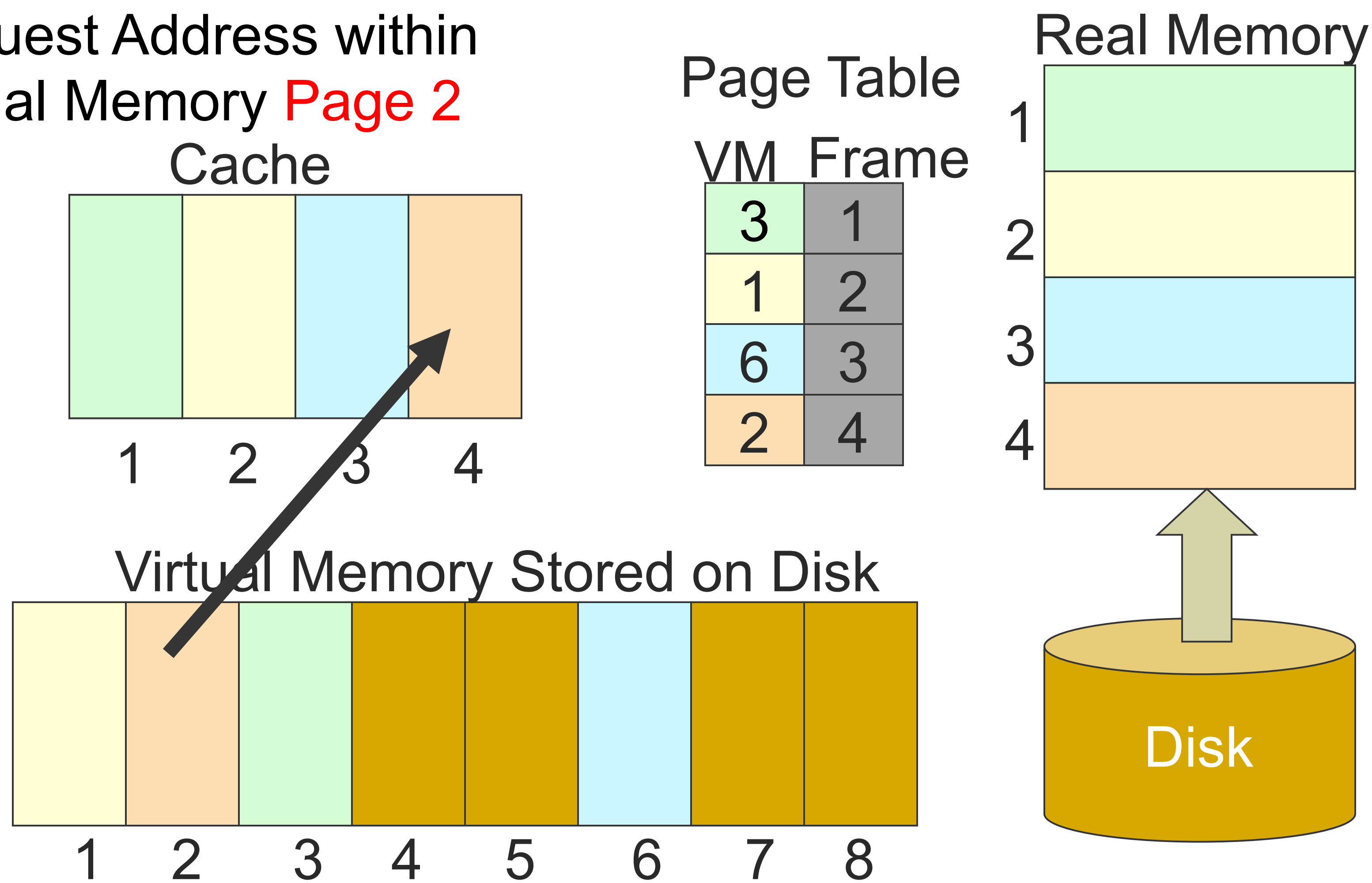
Paging Example

Request Address within Virtual Memory Page 6



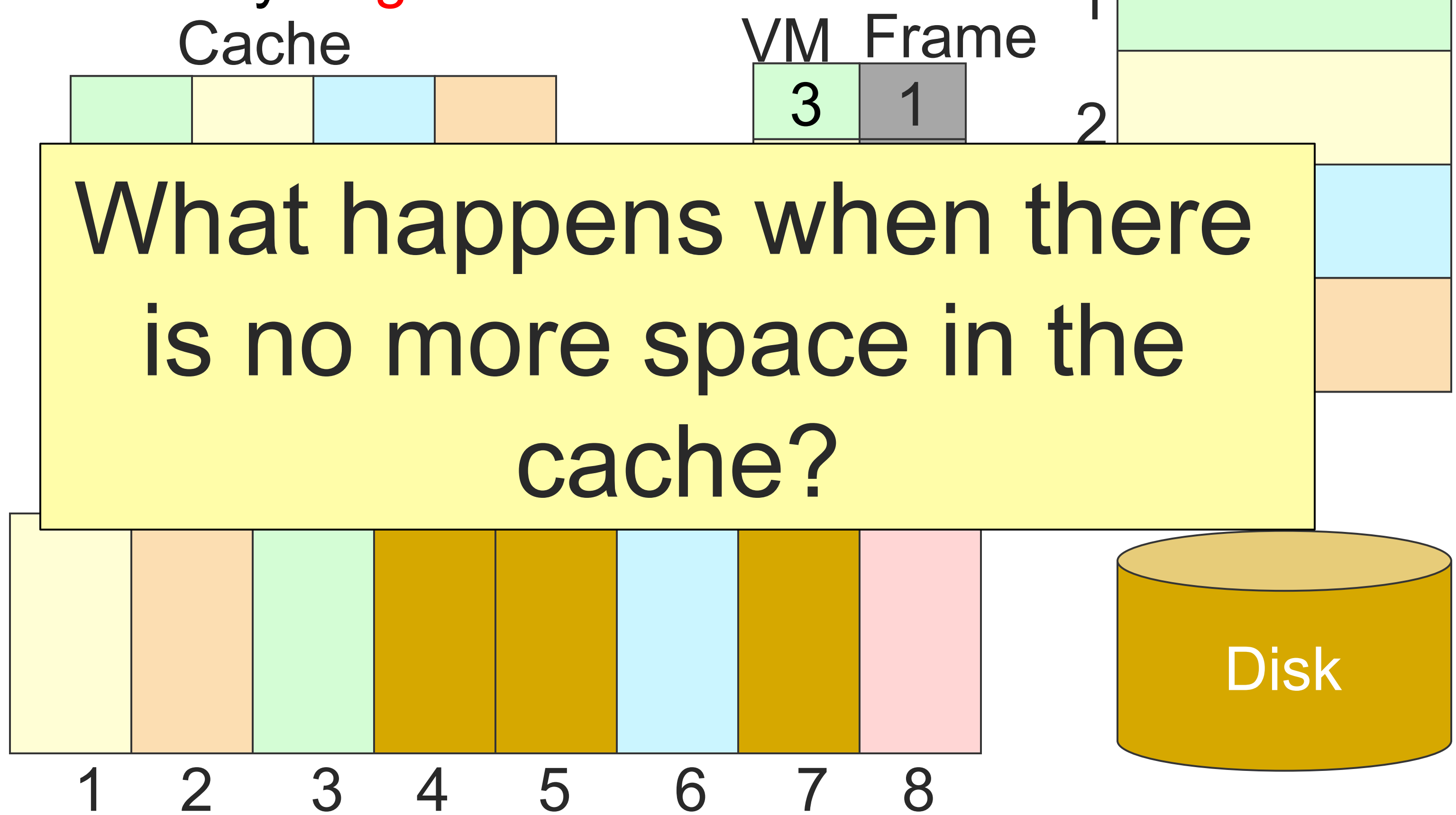
Paging Example

Request Address within
Virtual Memory Page 2



Paging Example

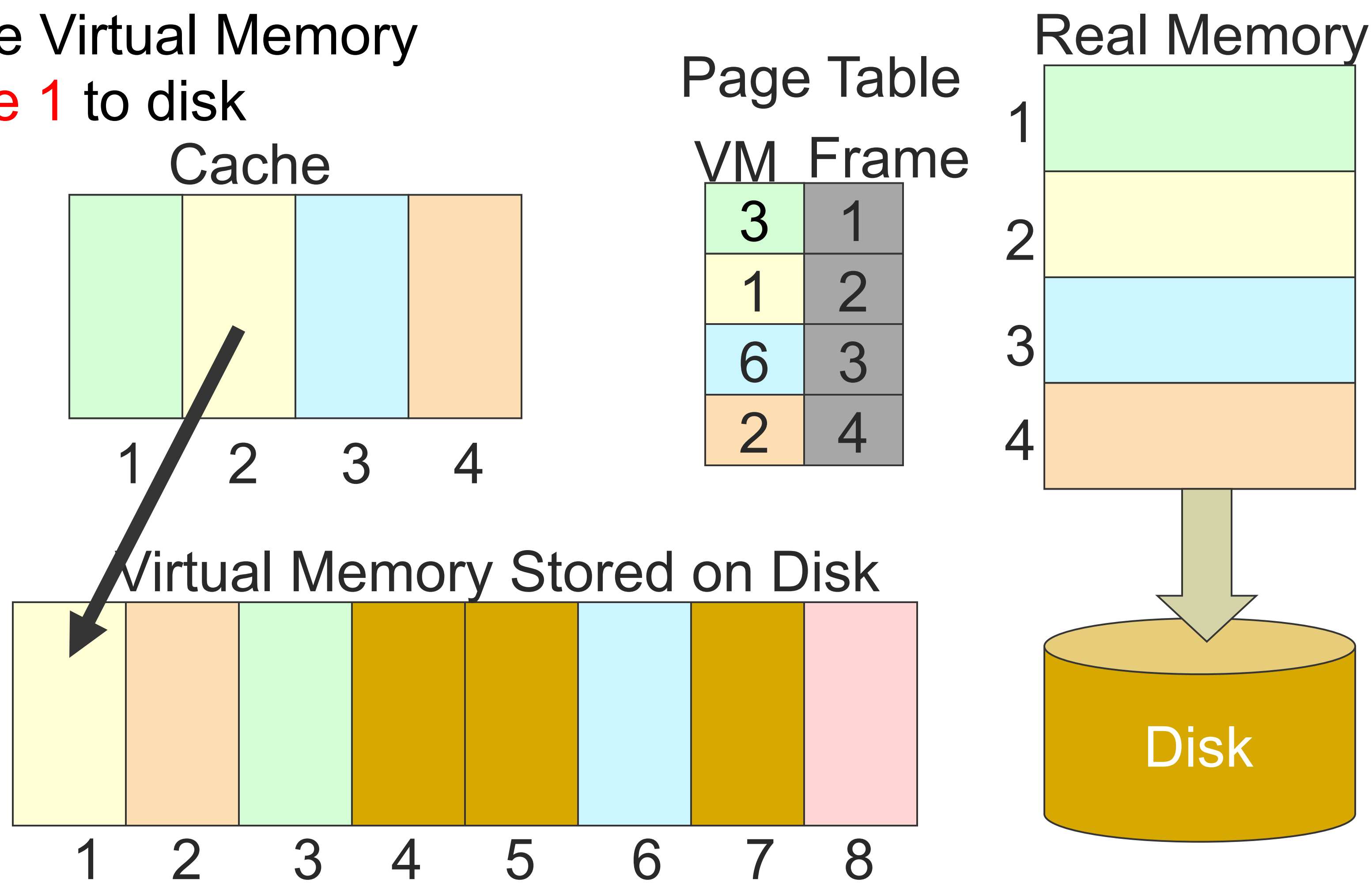
Request Address within Virtual Memory Page 8



What happens when there is no more space in the cache?

Paging Example

Store Virtual Memory
Page 1 to disk

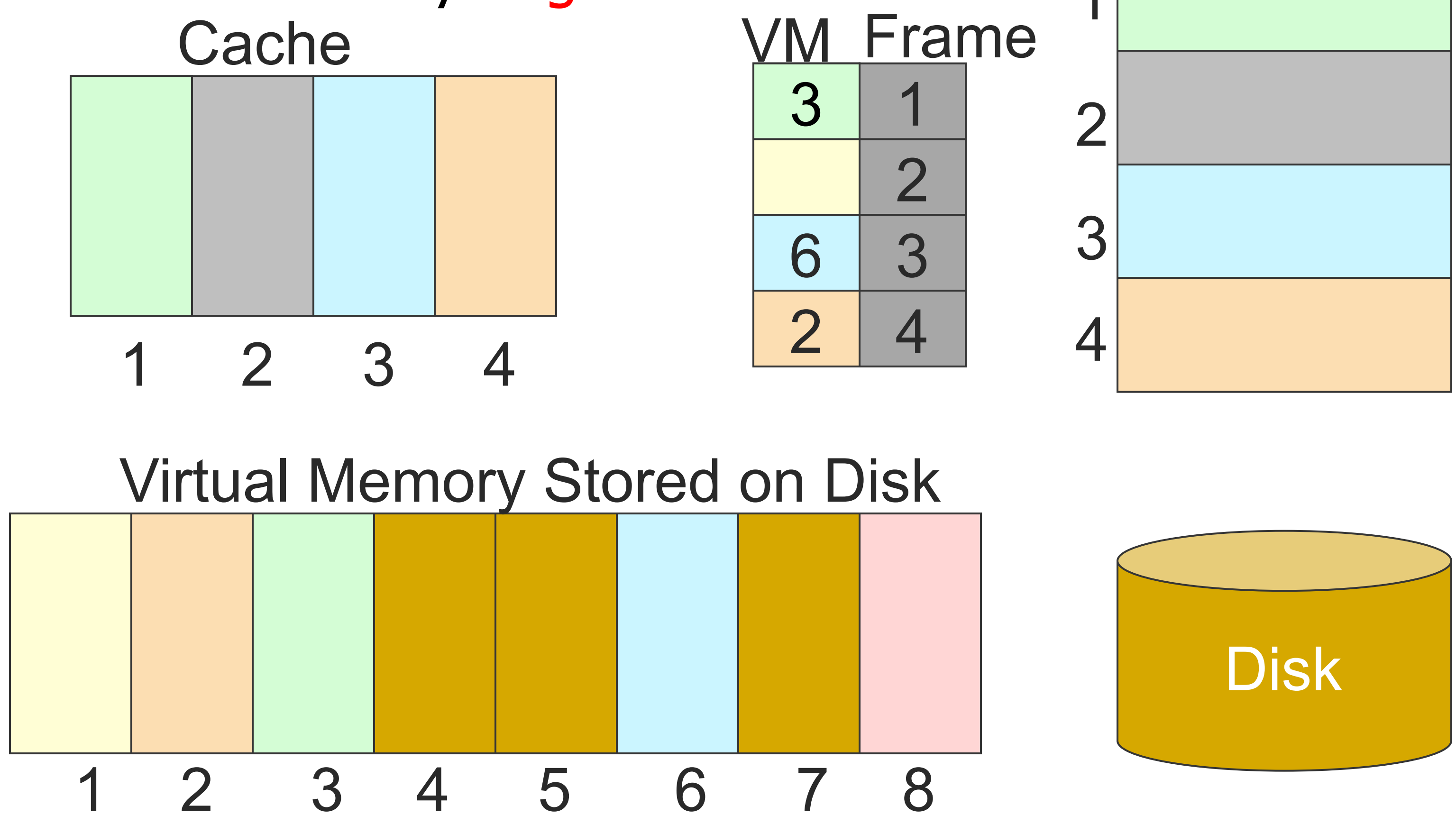


Page Table

VM	Frame
3	1
1	2
6	3
2	4

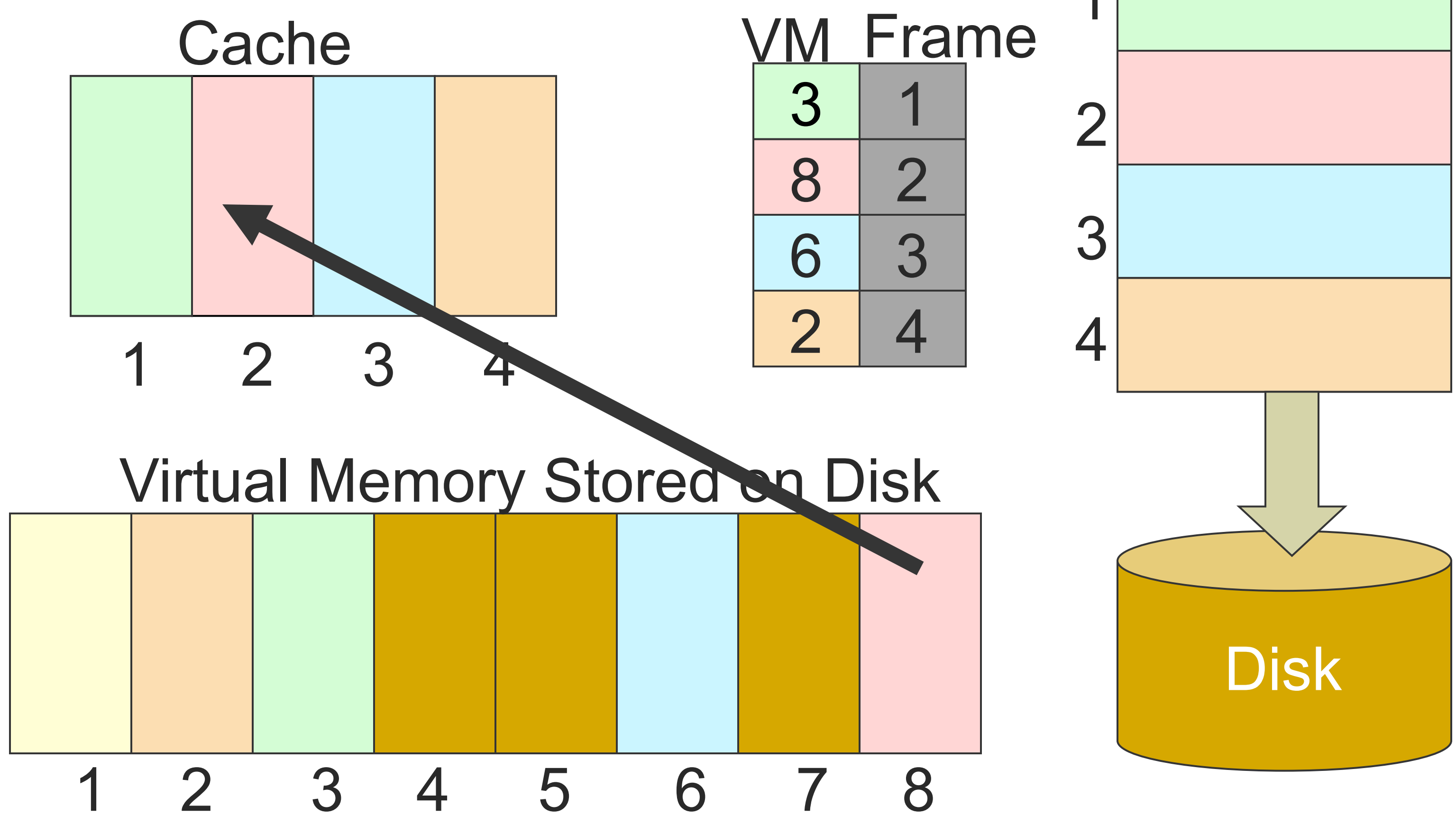
Paging Example

Process request for Address within Virtual Memory Page 8



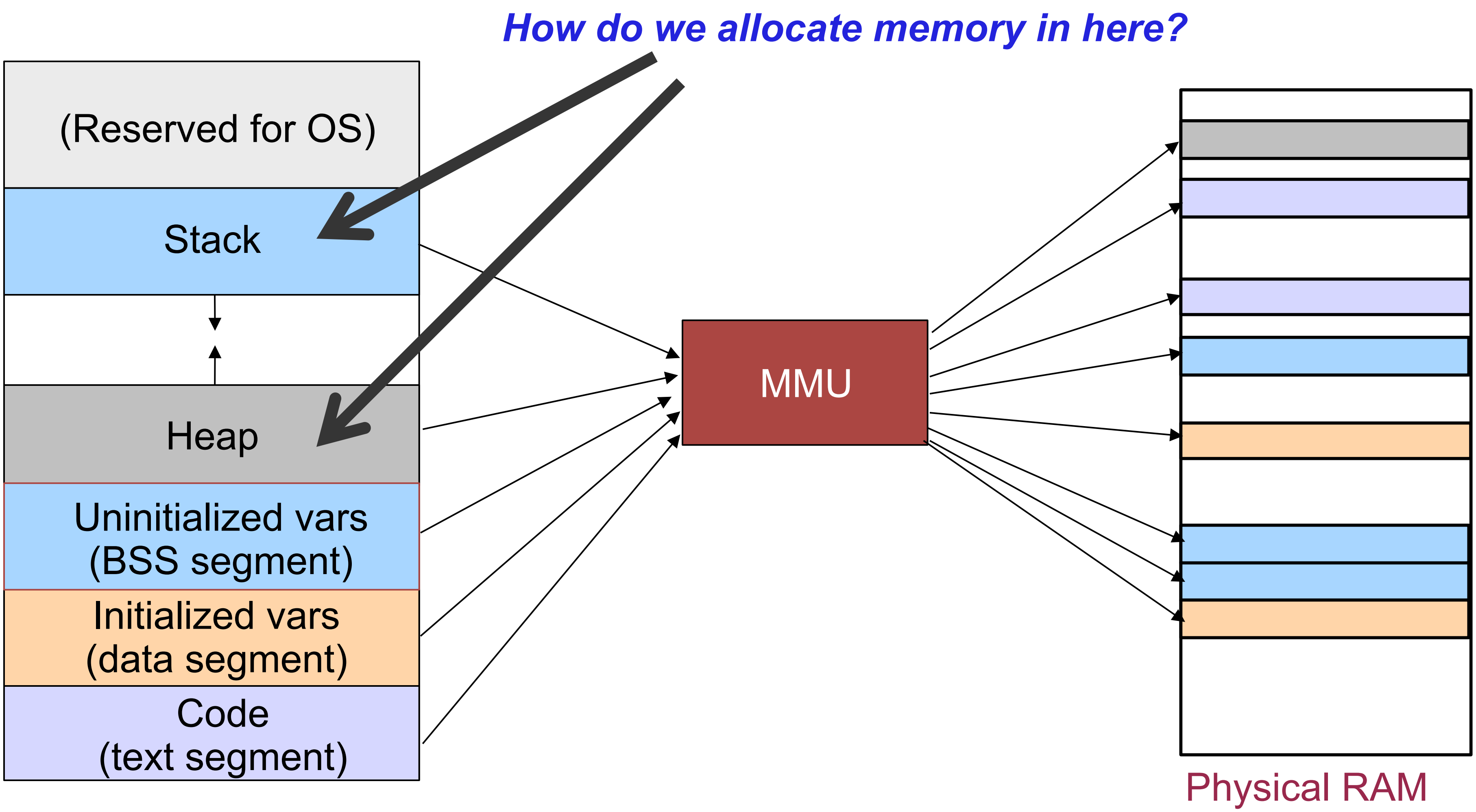
Paging Example

Load Virtual Memory
Page 8 to cache



VM	Frame
3	1
8	2
6	3
2	4

Is paging enough?



Memory allocation w/in a process

- What happens when you declare a variable?
  - Allocating a page for every variable wouldn't be efficient
  - Allocations within a process are much smaller
  - Need to allocate on a finer granularity
- Solution (stack): stack data structure (duh)
  - Function calls follow LIFO semantics
  - So we can use a stack data structure to represent the process's stack – no fragmentation!
- Solution (heap): **malloc**
  - This is a much harder problem
  - Need to deal with fragmentation