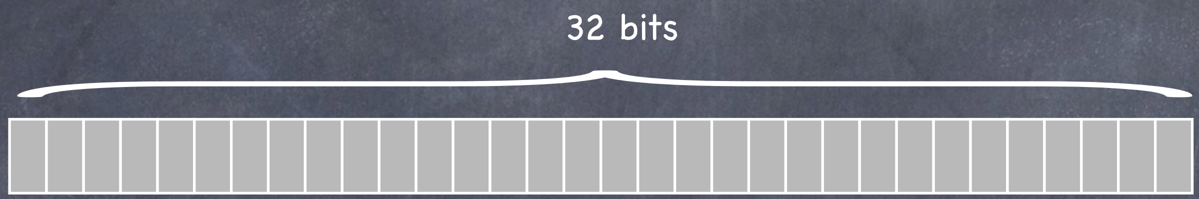
Eliminating External Fragmentation: Paging

- Allocate VA & PA memory in chunks of the same, fixed size (pages and frames, respectively)
- Adjacent pages in VA (say, within the stack) need not map to contiguous frames in PA!
  - Free frames can be tracked using a simple bitmap
    - 0011111001111011110000 one bit/frame
  - No more external fragmentation!
  - But now internal fragmentation (you just can't win...)
    - when memory needs are not a multiple of a page
    - typical size of page/frame: 4KB to 16KB

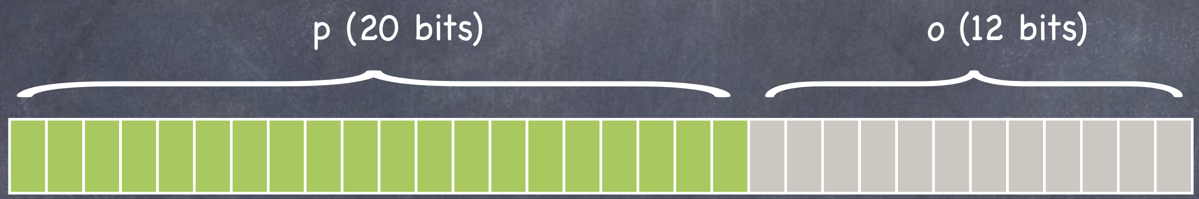
How can I reference a byte in VA space?

Virtual address



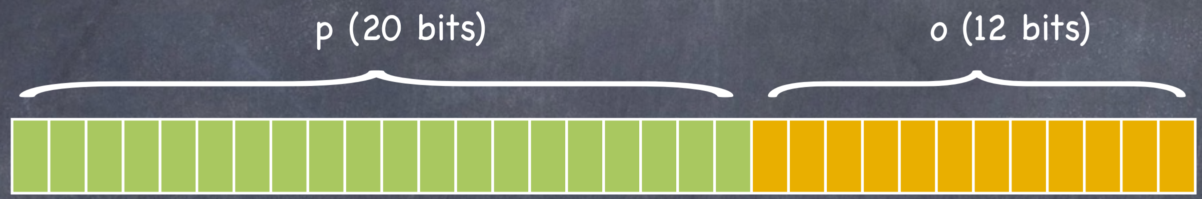
- Interpret VA as comprised of two components
  - page: which page?
  - offset: which byte within that page?

Virtual address



- Interpret VA as comprised of two components
  - page: which page?
    - no. of bits specifies no. of pages are in the VA space
  - offset: which byte within that page?

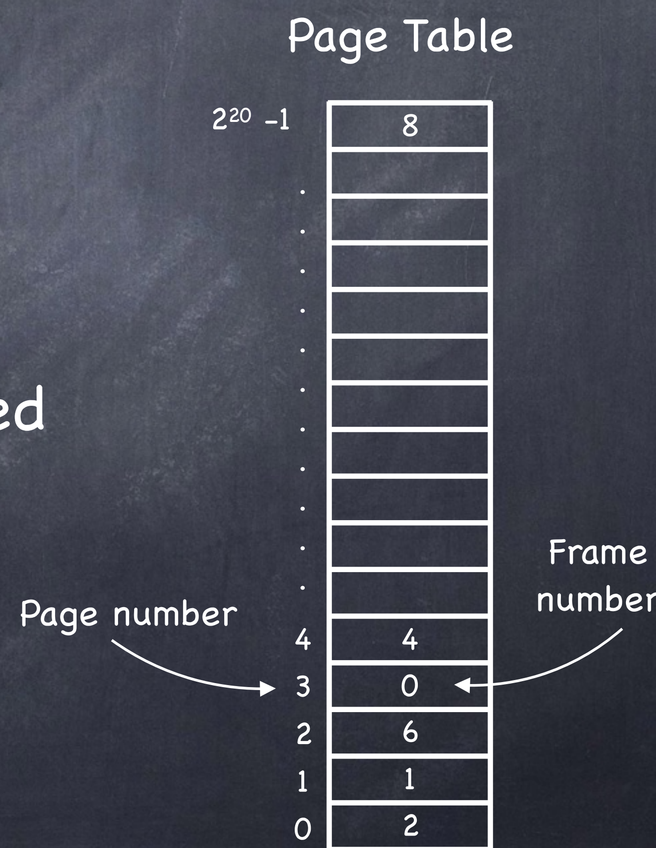
Virtual address



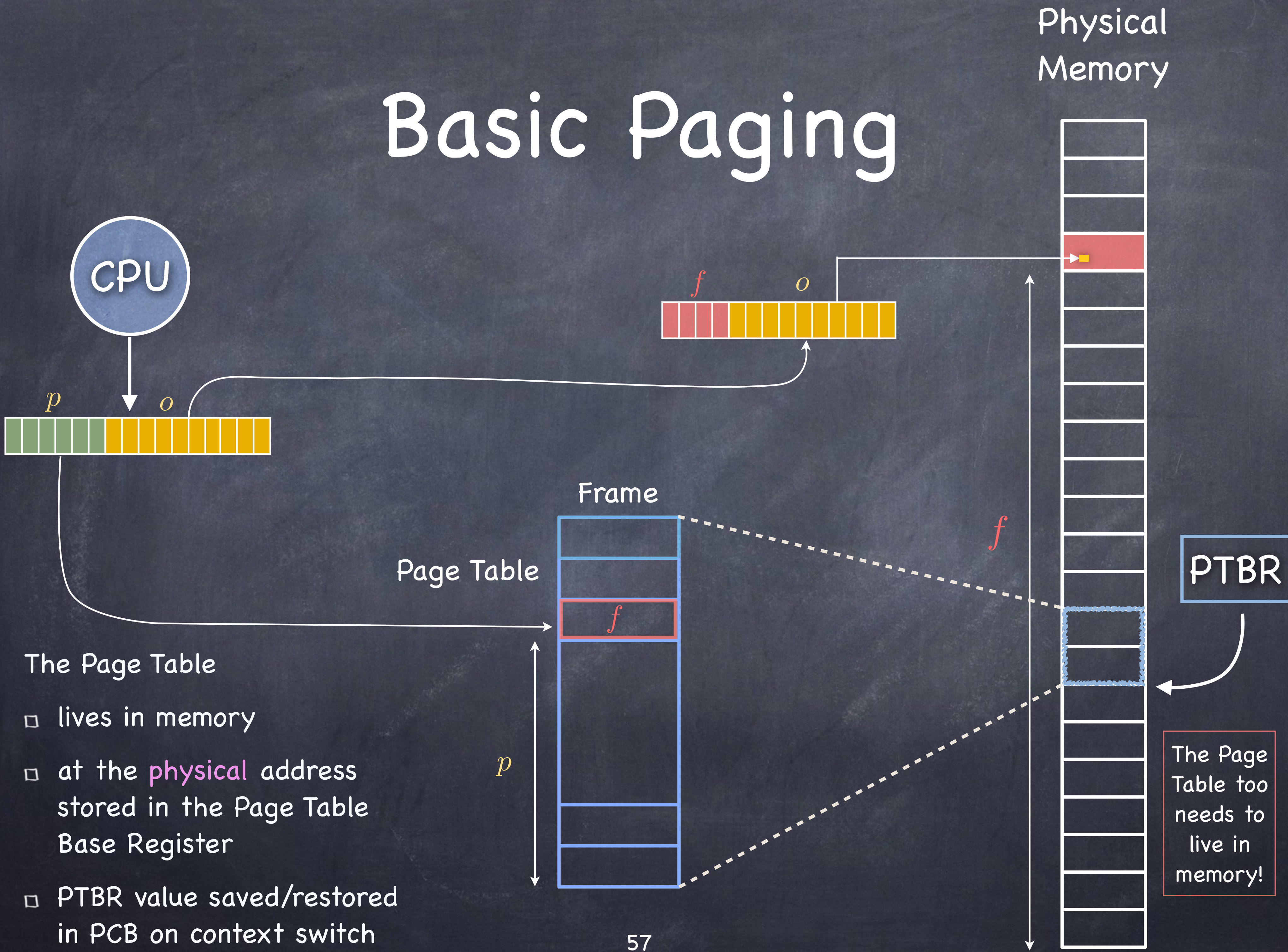
- Interpret VA as comprised of two components
  - page: which page?
    - no. of bits specifies no. of pages are in the VA space
  - offset: which byte within that page?
    - no. of bits specifies size of page/frame

Virtual address

- To access a byte
  - extract page number
  - map that page number into a frame number using a page table
    - Note: not all pages may be mapped to frames
  - extract offset
  - access byte at offset in frame



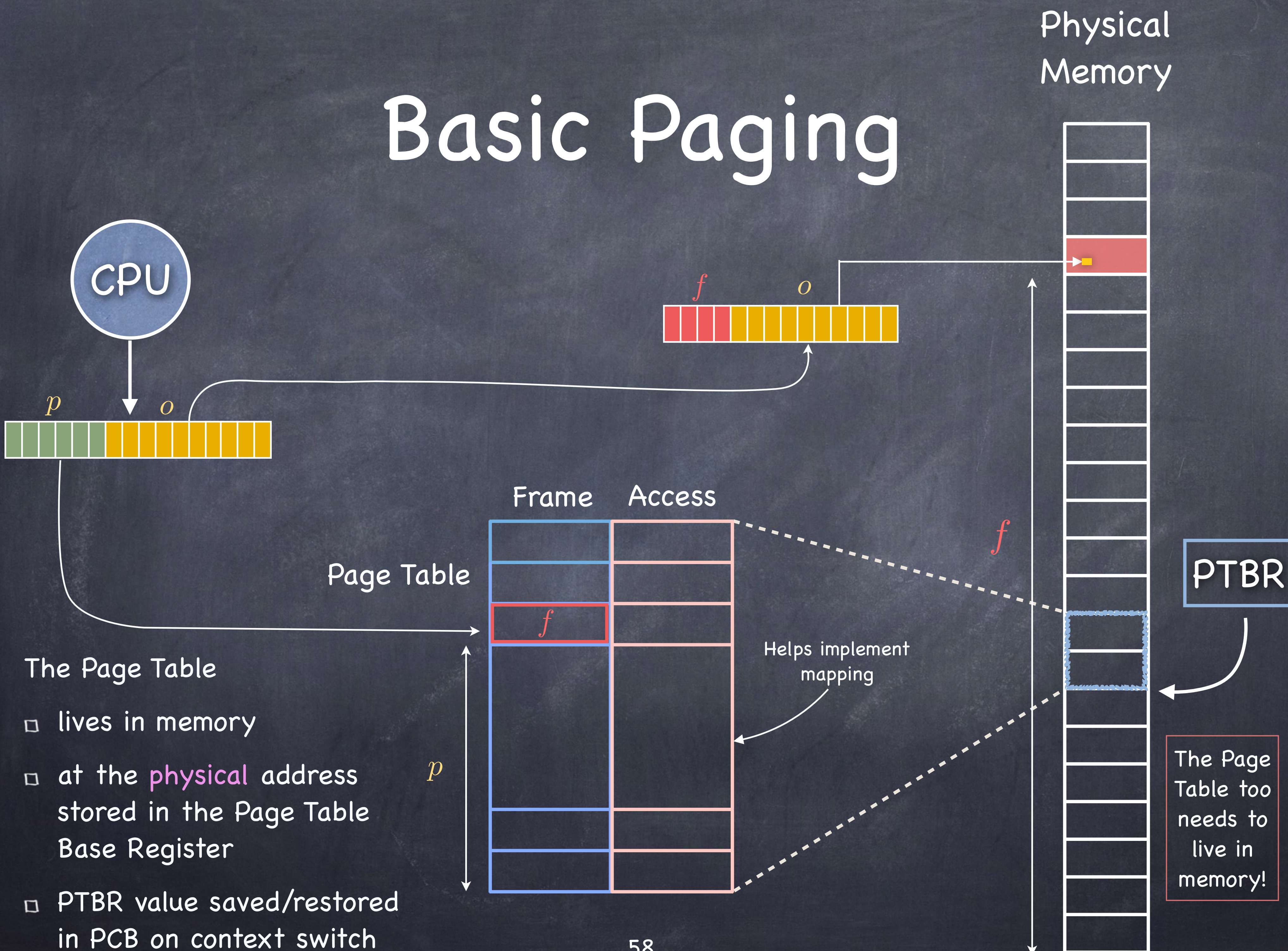
Basic Paging



The Page Table

- lives in memory
- at the physical address stored in the Page Table Base Register
- PTBR value saved/restored in PCB on context switch

Basic Paging

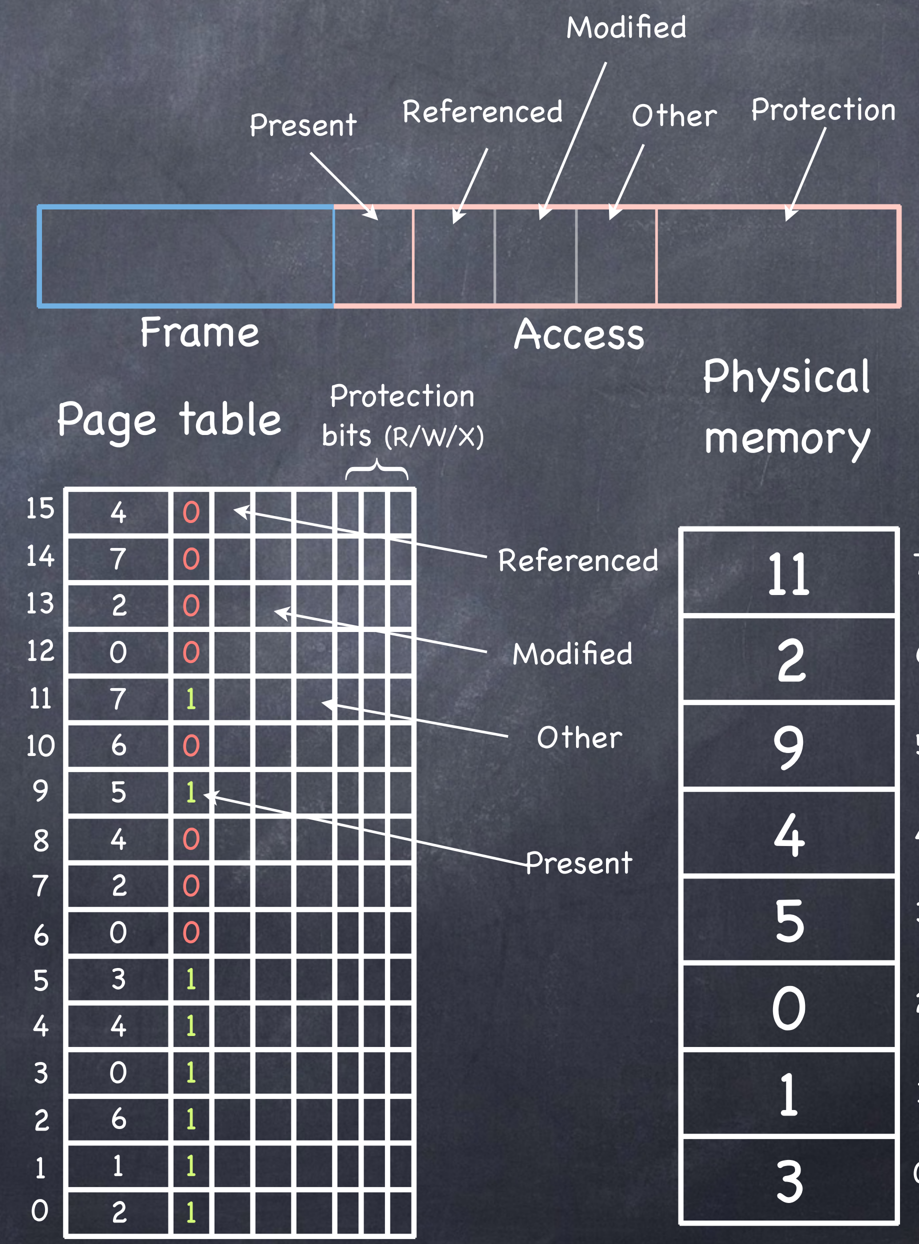


The Page Table

- lives in memory
- at the physical address stored in the Page Table Base Register
- PTBR value saved/restored in PCB on context switch

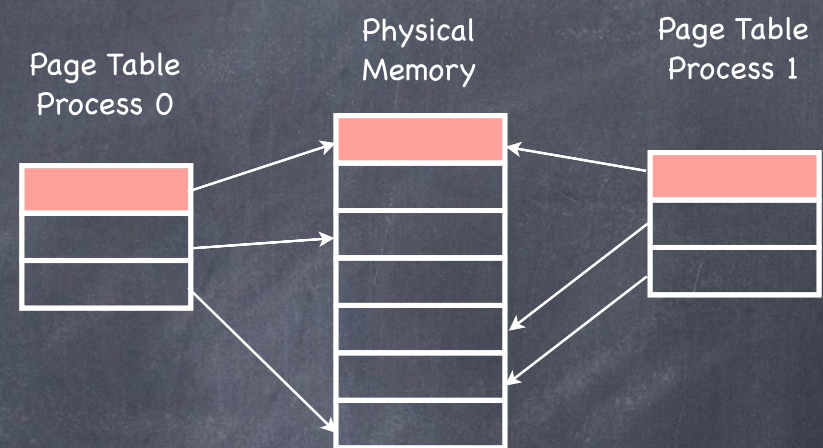
Page Table Entries

- Frame number
- Present (Valid/Invalid) bit
  - Set if entry stores a valid mapping. If not, and accessed, page fault
- Referenced bit
  - Set if page has been referenced
- Modified (dirty) bit
  - Set if page has been modified
- Protection bits (R/W/X)



Sharing

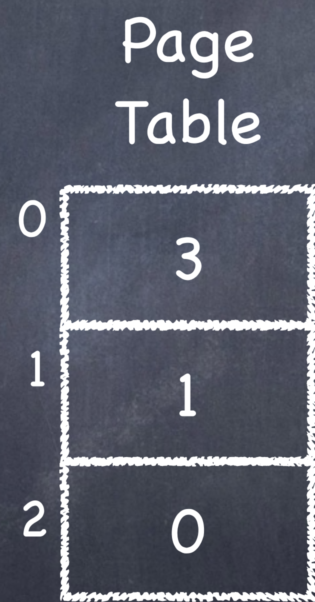
- Processes share a page by each mapping a page of their own virtual address space to the same frame
  - Fine tuning using protection bits (RWX)

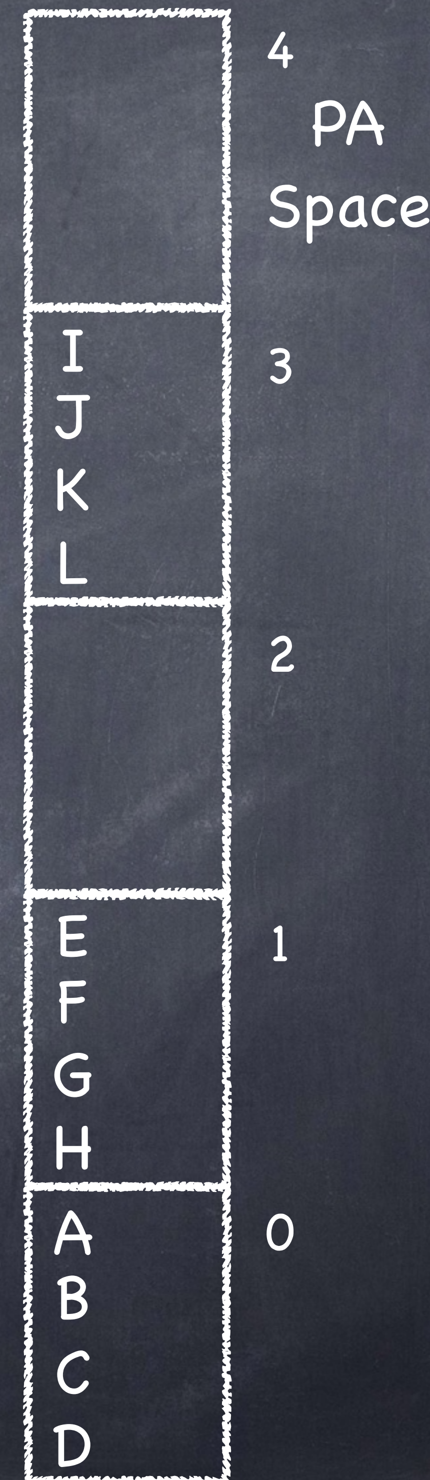


- We can refine COW to operate at the granularity of pages
  - on fork(), mark all pages in page table Read only
  - on write:
    - page fault
    - allocate new frame
    - copy page
    - mark both pages R/W

Example

Page size: 4 bytes





Space overhead

- Two sources, in tension:
  - data structure overhead (the Page Table itself)
  - fragmentation
    - How large should a page be?

Overhead for paging: sets of contiguous pages

(#entries x sizeofEntry) + (#"segments" x pageSize/2) =

= ((VA_Size/pagesize) x sizeofEntry) + (#"segments" x pageSize/2)

- What determines sizeofEntry?
  - enough bits to identify physical page ($\log_2$ (PA_Size / page size))
  - should include control bits (present, dirty, referenced, etc)
  - usually word or byte aligned

Computing paging overhead

- 1 MB maximum VA, 1 KB page, 3 segments (program, stack, heap)
  - $((2^{20} / 2^{10}) \times sizeofEntry) + (3 \times 2^{9})$
  - If I know PA is 64 KB then sizeofEntry = sizeofFrameNo + #ofAccessBits = 6 (since we have 2^{6} frames) + #ofAccessBits
    - if 7 access bits, byte aligned size of entry: 16 bits

What's not to love?

- Space overhead
  - With a 64-bit address space, size of page table can be huge!
- Time overhead
  - Accessing data now requires <u>two</u> memory accesses
    - must also access page table, to find mapped frame

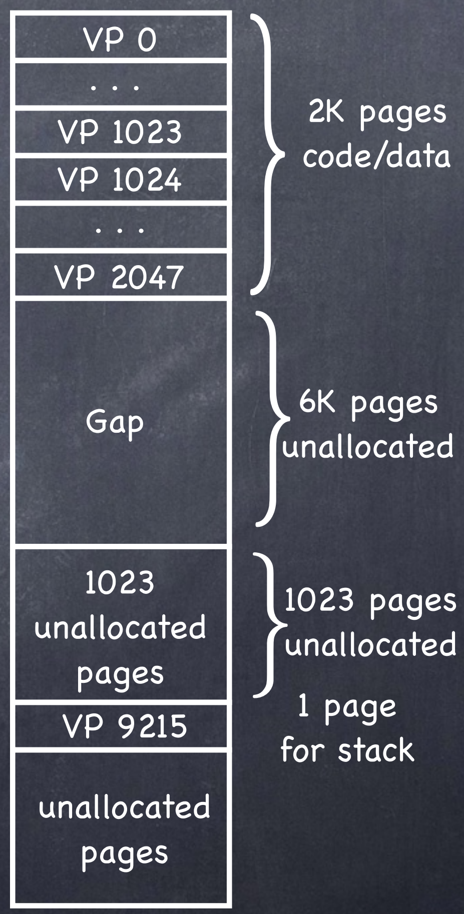
...and, like most times, space and time are in tension...

Reducing the Storage Overhead of Page Tables

- Size of the page table for a machine with 64-bit addresses and a page size of 4KB?
  - an array of 2^{52} entries!
- Good news
  - most space is unused
- Use a better data structure to express the Page Table
  - a tree!

Example

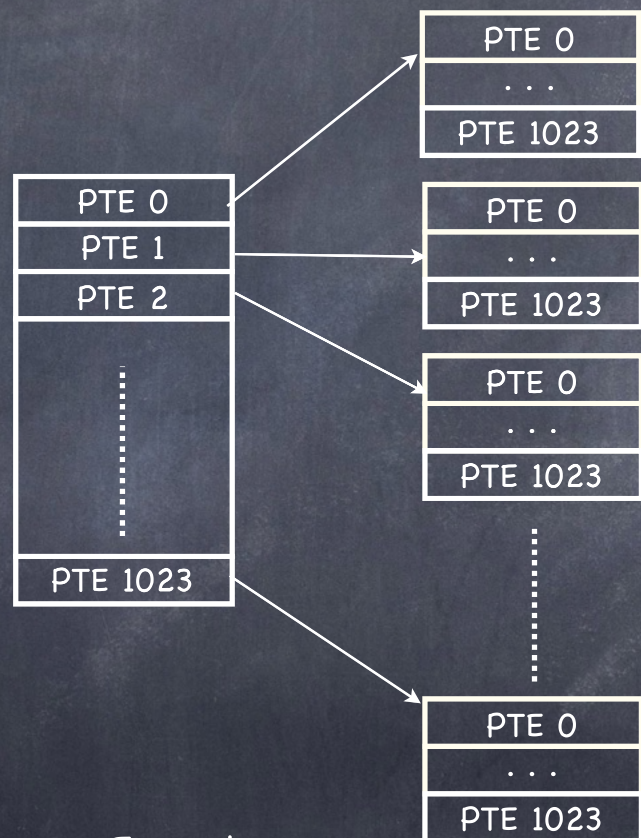
- 32 bit address space
- 4Kb pages
- 4 bytes PTE



Page Table

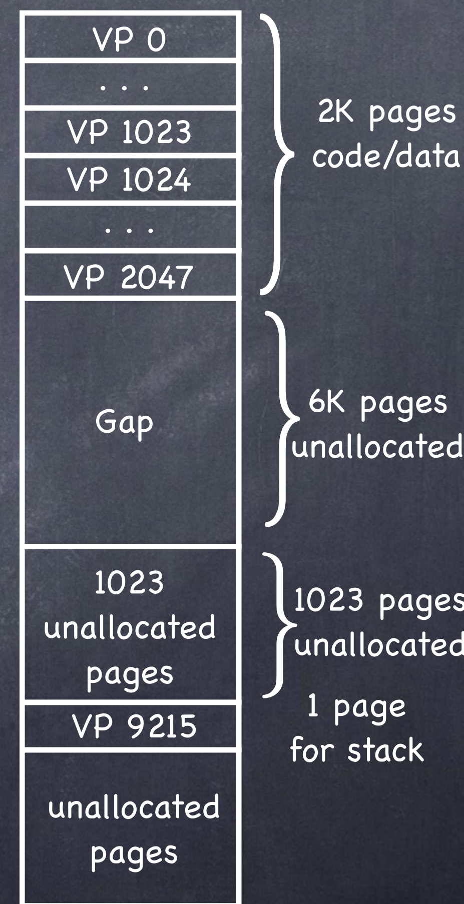
Reducing the Storage Overhead of Page Tables

- Size of the page table for a machine with 64-bit addresses and a page size of 4KB?
  - an array of 2^{52} entries!
- Good news
  - most space is unused
- Use a better data structure to express the Page Table
  - a tree!



Example

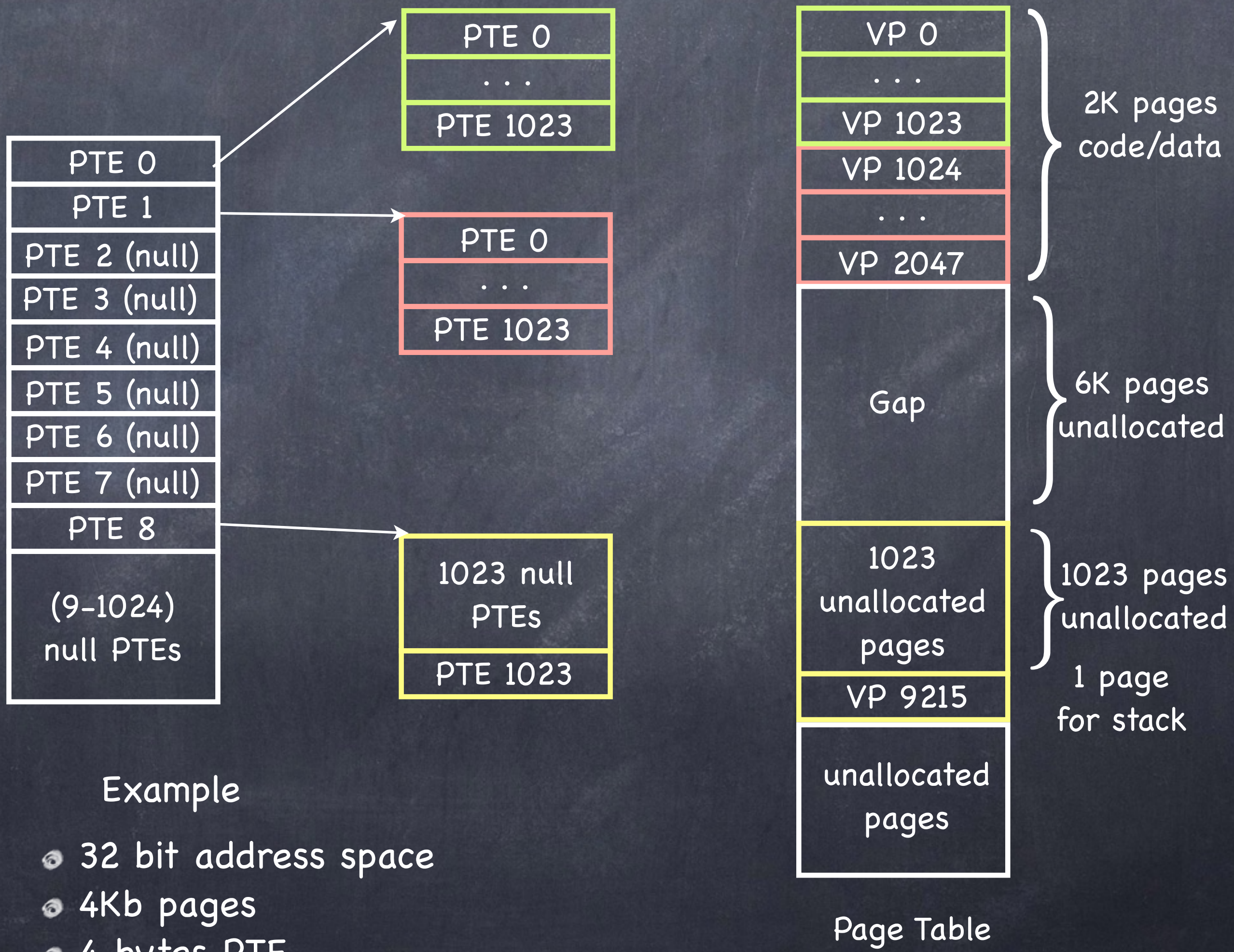
- 32 bit address space
- 4Kb pages
- 4 bytes PTE



Page Table

Reducing the Storage Overhead of Page Tables

- Size of the page table for a machine with 64-bit addresses and a page size of 4KB?
  - an array of 2^{52} entries!
- Good news
  - most space is unused
- Use a better data structure to express the Page Table
  - a tree!



Example

- 32 bit address space
- 4Kb pages
- 4 bytes PTE