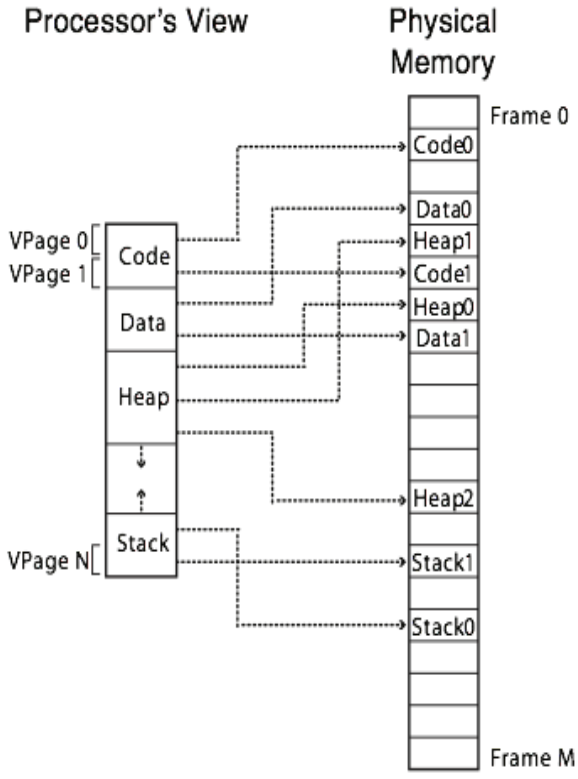
Topic 14: Paging

Reading: 8.2.2 - 8.3
Next reading: 9.1 - 9.5, 9.7

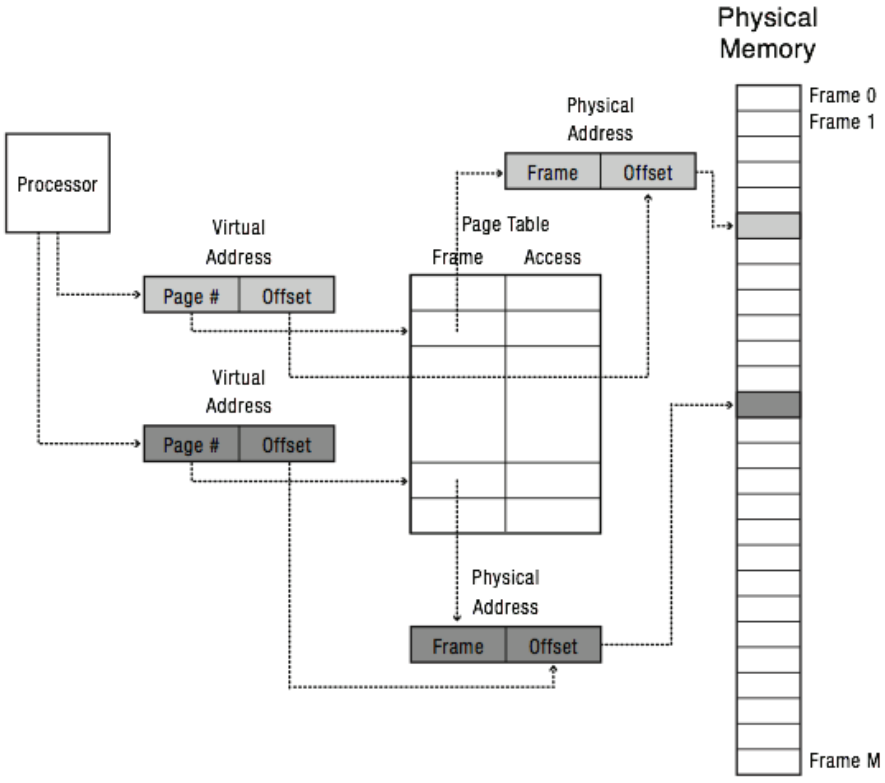
- Goal: make allocation and swapping easier than in segmentation, and reduce memory fragmentation.
- Make all chunks of memory the same size (power of two), call them *pages* (*page frames* or just *frames* in physical memory). Typical sizes range from 512-8k bytes. (Figure 8.5)



- For each process, a *page table* contains the frame for each page along with access (read/write/execute) and existence bits.

- There is one row in the table for each page in the VAS.
- Each row is a *page table entry (PTE)*.

- Translation procedure: page number comes from the virtual address and is used to index page table. (Figure 8.6)



- Similar to how segment number is extracted from virtual addres
- Note that offset is appended to frame, not added as in segmentation. Frames cannot start at arbitrary addresses, they must be aligned.

- Easy to allocate: keep a free list of available pages and grab the first one. Easy to swap since everything is the same size.
- Can share memory by sharing page table entry (page frame). Note that the virtual addresses do not need to be the same.
- Example. 2-bit page numbers, 14-bit offsets, 2-bit frame numbers

logical physical

Page	Frame
0	3
1	1
2	0
3	2

  - What PA does VA 0x0000 map to?
  - What PA does VA 0x4123 map to?
  - What VA does PA 0xFFFF map to?
- Problems:
  - Efficiency of access: even small page tables are generally too large to be stored in the MMU. Instead, page tables are kept in main memory and the MMU has only the page table's base address. Thus there is one memory access of overhead for every real memory access.

- Table space: if pages are small, the table size could be substantial. In fact, this is a problem even for normal page sizes: consider a 32-bit addresses space with 4k pages.
  - How big must the page table be?
  - Would larger pages help?
- *Internal fragmentation*: page size doesn't match up with information size. The larger the page, the worse this is.

Paged Segmentation: combine segmentation and paging to make page tables manageable.

- Each segment contains one or more pages.
- Segments correspond to logical units: code, data, heap and stack.
  - Segments vary in size and are often large.
- Pages are for the use of the OS; they are fixed-size to make it easy to manage memory.
- Going from paging to P+S is like going from single segment to multiple segments, except at a higher level. Instead of having a page table per process, have a page table per segment. (Figure 8.7)