Paging

- Divide main memory into set of fixed size blocks called frames, and divide process into equal sized blocks called pages.

- When process needs to run, bring pages of process into frames of main memory.

- Memory manager has to do following:
    - decide # of pages in a process
    - Have enough empty frames in main memory
    - Load program's pages into empty frames.

- Efficient use of memory
- No external fragmentation
- Almost no internal fragmentation.
- Can be loaded to frames that are non-contiguous.

- Increased overhead
- Internal fragmentation still exists
- If page size is ~~small~~ small → page tables are larger
- If page size is large → Internal fragmentation increases,

- page table is used to map pages of each process to frames.
- Basically, for each page, it it has been already loaded to a frame, we keep track of the starting memory address of that frame.
- logical address idea, can be used here.
- Suppose, we have $n+m$ bit logical address, first n bits can be used to keep track of page #. Next m bits can be used to keep track of the offset ~~of~~ ~~the~~ from the ~~beggin~~ begining of the page.
- CPU can decode this address by looking at the first n-bits, then looking at the page table and substitute frame address along with the offset.

Segmentation

- Divide process into logical segments.
- May vary in length.
- There is a maximum length.
- logical address contain 2 parts.
  - segment #
  - An offset
- Similar to Dynamic partitioning
- Eliminates Internal fragmentation.
- Usually visible to programmer/compiler.
- Segment Table for each process.

- Address Translation
  - Segment # → starting physical address of the segment
  - Displacement/offset → offset + starting Add. is compared against the length of seg.
  - Start. Add + offset is physical Add.