CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

Introduction to Virtual Memory

Ch 10.1 – 10.2

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Xiannong Meng, Fall 2021.

Problems with Memory System

- The total memory needed by running processes may exceed the amount of physical memory.
  - In one example we saw, there were 400+ processes active at one time on *linuxremote.*
- At the same time, only about 10 percent of the code is really needed (executed) in a program life time, according to the 90/10 rule.
  - https://softwareengineering.stackexchange.com/questions/334528/what-is-the-meaning-of-the-90-10-rule-of-program-optimization

Solutions?

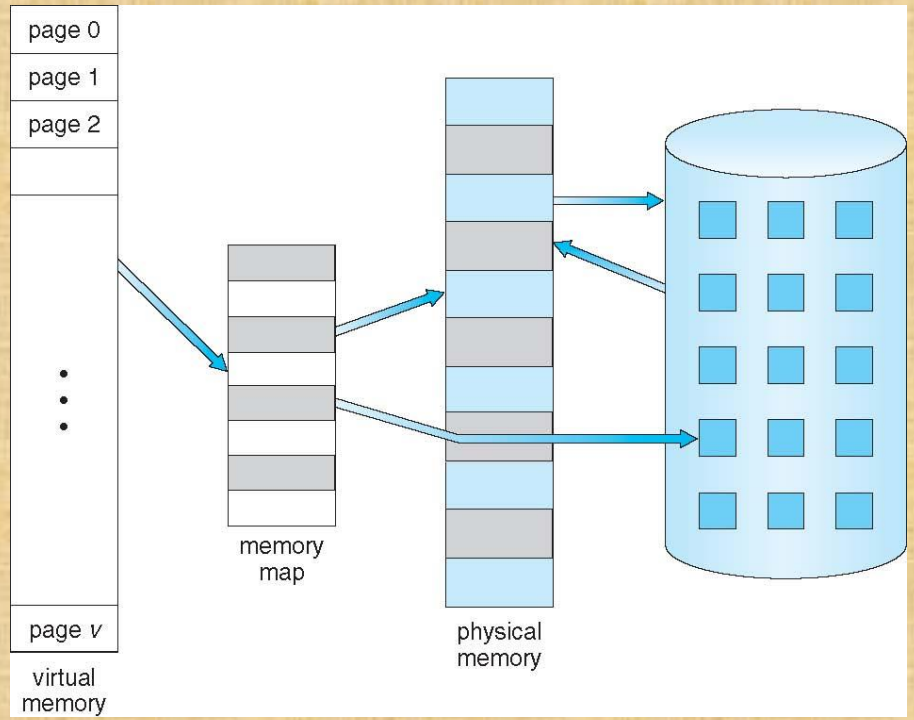
- We keep only the active part of the programs in memory as needed.
- Other less active part of the programs are stored on secondary storage. They are brought into the memory only when being called.
- Doing so maximizes the number of programs (processes) can be in memory at any time.
- **Virtual memory** is one such solution.

Virtual Memory

- **Virtual memory** – separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution.
  - Logical address space can therefore be much larger than physical address space.
  - Allows address spaces to be shared by several processes.
  - Allows for more efficient process creation.

- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

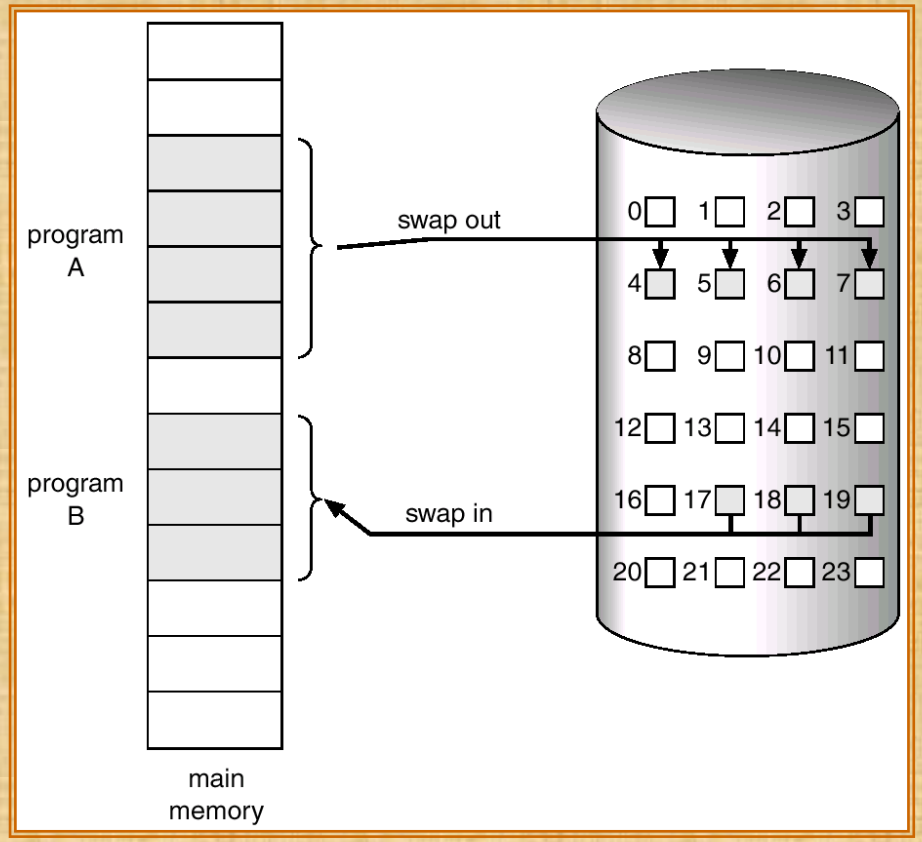
Virtual Memory That is Larger Than Physical Memory



Demand Paging

- Bring a page into memory only when it is needed.
  - Less I/O needed.
  - Less memory needed.
  - Faster response.
  - More users.
- When a page is needed (there is a reference to it):
  - invalid reference ⇒ abort.
  - not-in-memory ⇒ bring it into to memory.
- **Lazy swapper** – never swap a page into memory unless page will be needed.

Mapping of a Paged Memory to Contiguous Disk Space



Note: contiguous space on disk...

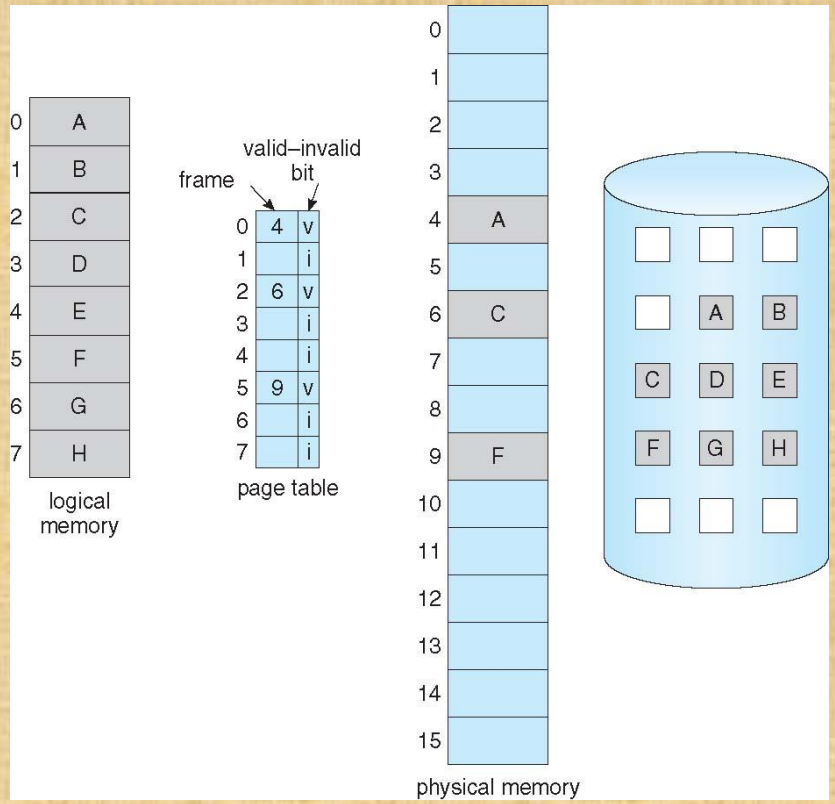
Valid-Invalid Bit

- With each page table entry a valid bit is associated
  (1 $\Rightarrow$ in-memory, 0 $\Rightarrow$ not-in-memory)
- Initially valid bit is set to 0 on all entries.
- During address translation, if valid bit in page table entry is 0 $\Rightarrow$ page fault.
- Example of a page table snapshot $\Rightarrow$

Frame #	valid
	1
	1
	1
	1
	0
⋮	
	0
	0

sample page table

Page Table When Some Pages Are Not in Main Memory



Page Fault

When a page that is needed is not in memory, we say a **page fault** occurs.

Page Fault and Its Handling

1. For any reference to a page, the very first reference will trap to OS $\Rightarrow$ page fault.

2. OS looks at page table and page limit table to decide:
   - If it was an invalid reference (address out of bound) $\Rightarrow$ abort.
   - If it was a reference to a page that is not in memory, continue.

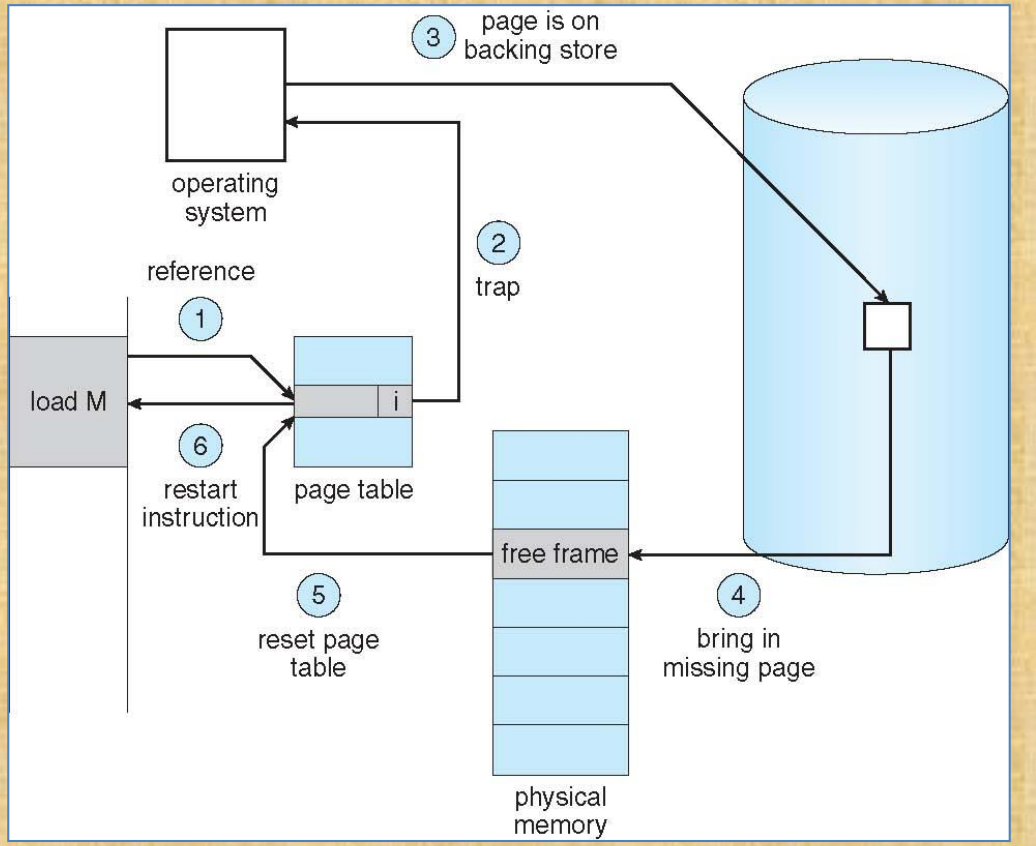
3. **Get an empty frame from the free-list.**

4. Bring the page content from disk into frame.

5. Update the page table and make valid bit = 1.

6. Restart the instruction that caused the page fault.

Steps in Handling a Page Fault

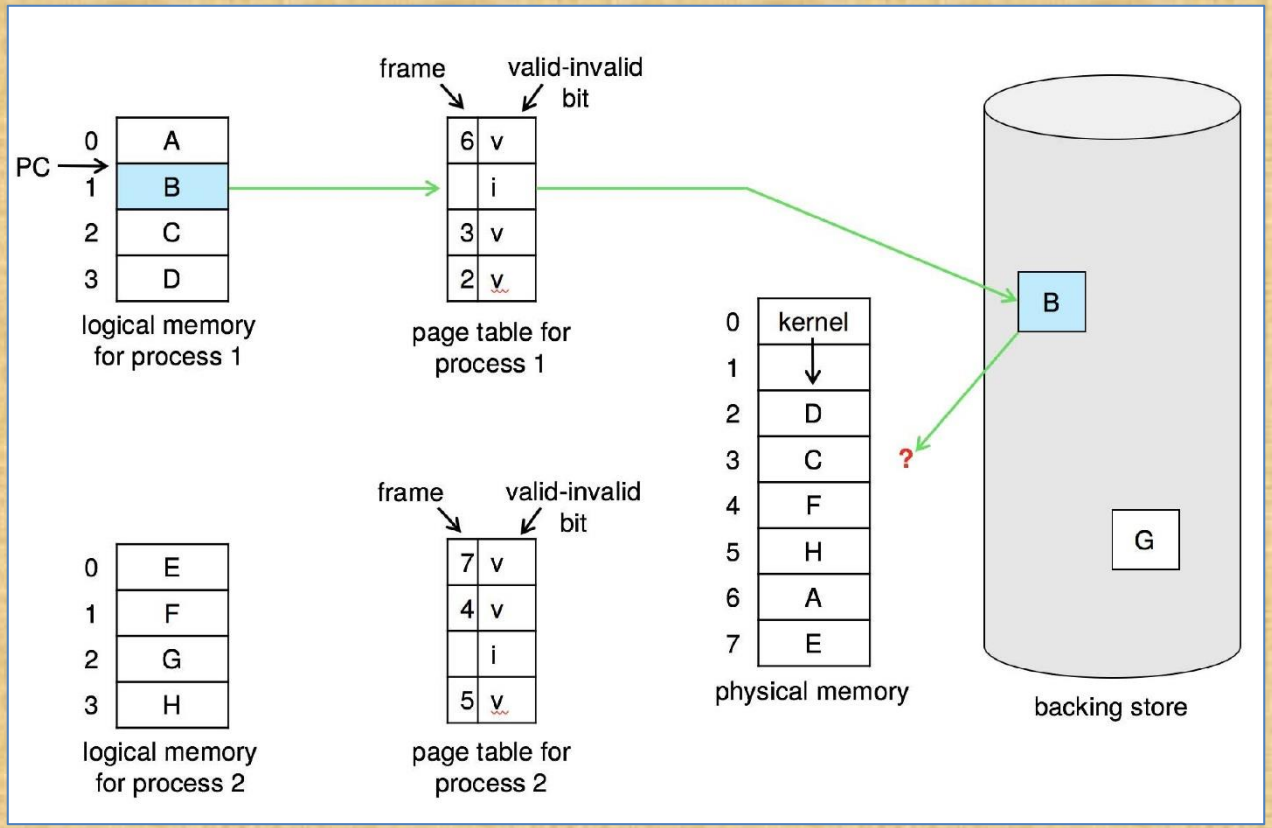


No free frame: now what?

- **Page replacement:** Are all those pages in memory being referenced? Choose one to swap back out to disk and make room to load a new page.
  - **Algorithm:** How you choose a victim.
  - **Performance:** Want an algorithm that will result in **minimum** number of page faults.

- Side effect: The same page may be brought in and out of memory several times.

Need For Page Replacement



Basic Page Replacement

1. Find the location of the desired page on disk
2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim frame**
   - Write victim frame to disk if dirty
3. Bring the desired page into the (newly) free frame; update the page and frame tables
4. Continue the process by restarting the instruction that caused the trap

Note now potentially 2 page transfers for page fault – increasing EAT

Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement.
- Use ***modify*** **(*dirty*)** ***bit*** to reduce overhead of page transfers – only modified pages are written to disk.
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory.