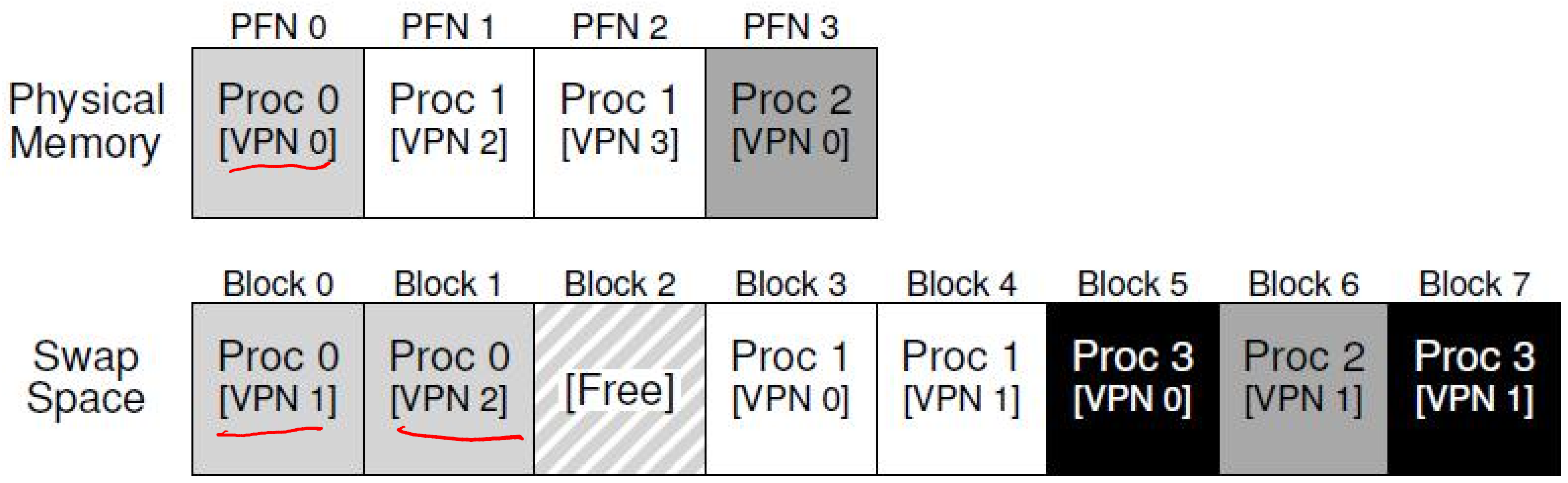
Lecture 10: Demand Paging

Mythili Vutukuru

IIT Bombay

Is main memory always enough?

- Are all pages of all active processes always in main memory?
  - Not necessary, with large address spaces
- OS uses a part of disk (swap space) to store pages that are not in active use



Page fault

- Present bit in page table entry: indicates if a page of a process resides in memory or not
- When translating VA to PA, MMU reads present bit
- If page present in memory, directly accessed
- If page not in memory, MMU raises a trap to the OS – page fault

Page fault handling

- Page fault traps OS and moves CPU to kernel mode
- OS fetches disk address of page and issues read to disk
  - OS keeps track of disk address (say, in page table)
- OS context switches to another process
  - Current process is blocked and cannot run
- When disk read completes, OS updates page table of process, and marks it as ready
- When process scheduled again, OS restarts the instruction that caused page fault

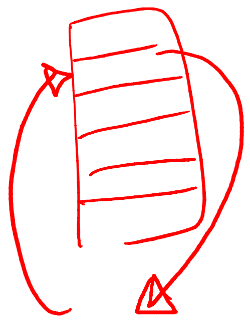
Summary: what happens on memory access

$$\begin{tikzcd}
\text{CPU} \arrow[r] & \text{cache} \arrow[d] \\
& \text{MMU} \arrow[d] \\
& \text{MEM}
\end{tikzcd}$$

- CPU issues load to a VA for code or data
  - Checks CPU cache first
  - Goes to main memory in case of cache miss
- MMU looks up TLB for VA
  - If TLB hit, obtains PA, fetches memory location and returns to CPU (via CPU caches)
  - If TLB miss, MMU accesses memory, walks page table, and obtains page table entry
    - If present bit set in PTE, accesses memory
    - If not present but valid, raises page fault. OS handles page fault and restarts the CPU load instruction
    - If invalid page access, trap to OS for illegal access

More complications in a page fault

- When servicing page fault, what if OS finds that there is no free page to swap in the faulting page?
- OS must swap out an existing page (if it has been modified, i.e., dirty) and then swap in the faulting page – too much work!
- OS may proactively swap out pages to keep list of free pages handy
- Which pages to swap out? Decided by page replacement policy.

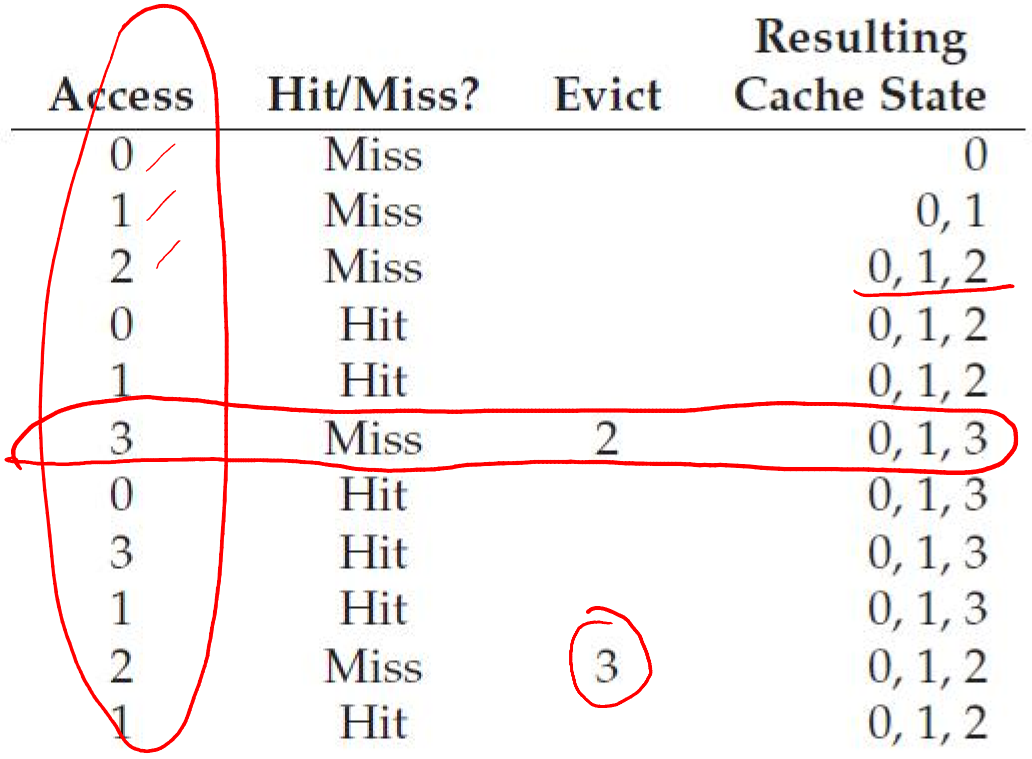


Page replacement policies

- Optimal: replace page not needed for longest time in future (not practical!)
- FIFO: replace page that was brought into memory earliest (may be a popular page!)
- LRU/LFU: replace the page that was least recently (or frequently) used in the past

Example: Optimal policy

- Example: 3 frames for 4 pages (0,1,2,3)
- First few accesses are cold (compulsory) misses



Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0, 1
2	Miss		0, 1, 2
0	Hit		0, 1, 2
1	Hit		0, 1, 2
3	Miss	2	0, 1, 3
0	Hit		0, 1, 3
3	Hit		0, 1, 3
1	Hit		0, 1, 3
2	Miss	3	0, 1, 2
1	Hit		0, 1, 2

Figure 22.1: **Tracing The Optimal Policy**

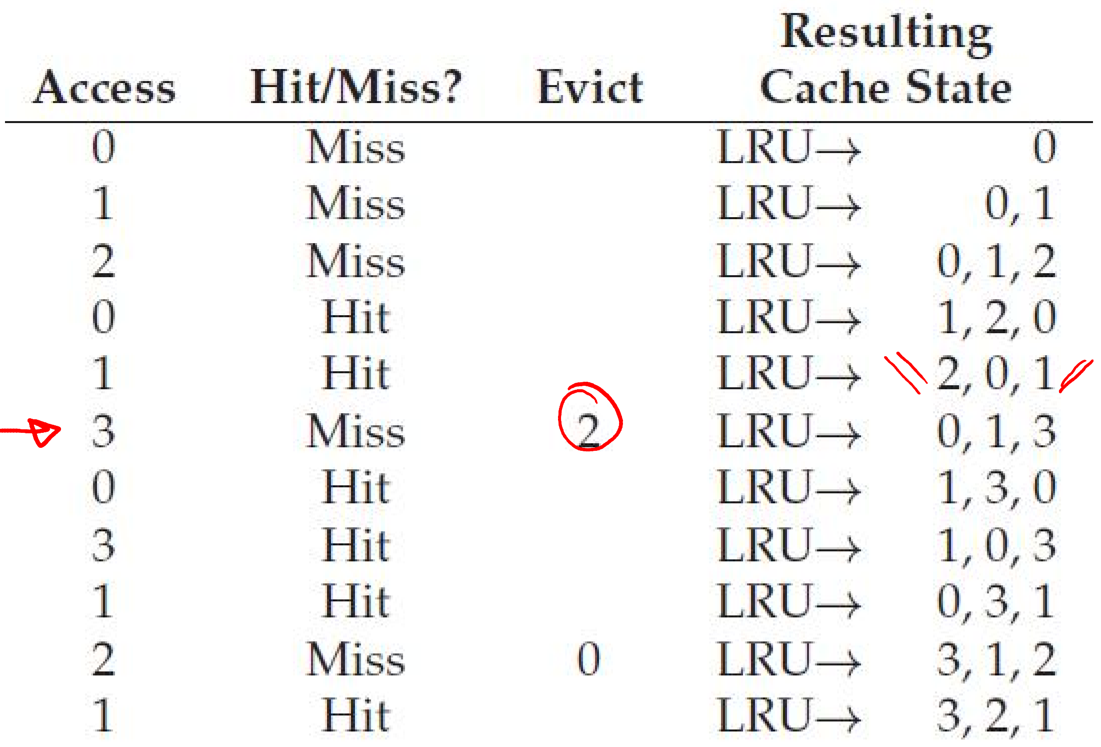
Example: FIFO

- Usually worse than optimal
- Belady's anomaly: performance may get worse when memory size increases!

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		First-in→ 0
1	Miss		First-in→ 0, 1
2	Miss		First-in→ 0, 1, 2
0	Hit		First-in→ 0, 1, 2
1	Hit		First-in→ ~~0~~, 1, 2
3	Miss	0	First-in→ 1, 2, 3
0	Miss	1	First-in→ 2, 3, 0
3	Hit		First-in→ ~~2~~, 3, 0
1	Miss	2	First-in→ 3, 0, 1
2	Miss	3	First-in→ 0, 1, 2
1	Hit		First-in→ 0, 1, 2

Example: LRU

- Equivalent to optimal in this simple example
- Works well due to locality of references



Access	Hit/Miss?	Evict	Resulting Cache State	
0	Miss		LRU→	0
1	Miss		LRU→	0, 1
2	Miss		LRU→	0, 1, 2
0	Hit		LRU→	1, 2, 0
1	Hit		LRU→	2, 0, 1
3	Miss	2	LRU→	0, 1, 3
0	Hit		LRU→	1, 3, 0
3	Hit		LRU→	1, 0, 3
1	Hit		LRU→	0, 3, 1
2	Miss	0	LRU→	3, 1, 2
1	Hit		LRU→	3, 2, 1

Figure 22.5: **Tracing The LRU Policy**

How is LRU implemented?

- OS is not involved in every memory access – how does it know which page is LRU?
- Hardware help and some approximations
- MMU sets a bit in PTE ("accessed" bit) when a page is accessed
- OS periodically looks at this bit to estimate pages that are active and inactive
- To replace, OS tries to find a page that does not have access bit set
  - May also look for page with dirty bit not set (to avoid swapping out to disk)