What happens if we don't have infinite supply of memory?

- Note that cpu can address data residing in memory - cpu cannot access data from disk or other storage devices. So, data has to be brought to memory before accessing them
- Physical memory (RAM) size is limited
- Not all processes content could be kept in memory
  - Do we limit the number of process running in the system - But this would limit the extent of multiprogramming.
  - Small address space - Programming becomes hard.

Swapping

- When memory is insufficient, use some storage space (disk space) to extend memory space
- We can’t just throw away data present in memory to make space (This would mean losing data). So, some physical pages are (probably pages that are not needed for now) to disk to make some space in memory
- If the content of the pages (content prior to swapping) are accessed sometime in the future, they are brought back to memory
- The storage space reserved for this process is called as swap space

Mechanism/Policy

- Similar to the idea of context switch and scheduling, swapping refers the process of moving physical page(s) contents from memory to swap space (or vice versa) is the mechanism and the algorithm that decides the pages to swap-out is the policy.

Mechanism

- Assume a page has been chosen to be swapped by the policy.
- After removing the page (page contents are moved to swap space), the PTE entry pointing to the page should not be valid (since the page will not reside in memory)
- OS marks the present bit in the PTE as '0'
- During address translation, hardware notices that the present bit is not set and raises a trap to OS that is usually referred as page fault
- OS is now responsible for bringing in the needed page from swap area and edit the PTE entry (point to new PFN) and update the present bit to '1'. The instruction that caused the page fault is re-executed.

OS Background process

- If swap process is carried out when the memory is full, it could result in performance degradation (Application is stopped, pages are moved to swap space to create some free pages and application is resumed)
- Separate process exists in OS called 'swap daemon' (different names in different OS) that performs swapping when amount of free pages in the memory goes below a certain threshold

Memory as cache

- The physical memory can be visualized as a caching layer for disk (TLB is cache for page table in memory). Going to disk for every access would be really really slow.
As for any cache in general, maximizing the cache hit rate is the ultimate goal of any cache replacement policy. Policy to swap pages from memory to swap area is nothing but a cache replacement policy.

**Ideal cache replacement policy**

- The best policy should move pages that will not be used anytime sooner from the pages and make room for data that will be accessed soon.
- But it is really hard to make futuristic prediction as to which page will be accessed now or later.
- Assume we have a memory trace and limited physical memory, an ideal policy could always be used as a baseline to compare how well other replacement policies. It should be noted that any replacement policy can perform lower or equal to the ideal policy and not better than that.

**Replacement policies**

- **FIFO**: If a page has to be swapped out, choose the page that was allocated first.
- **Random**: Choose any page in memory to be swapped out - Contents of randomly chosen page is swapped out and the page is now free.
- **LRU**: The page that was not recently accessed is picked for swapping

**Implementing a policy**

- **FIFO**: Track through a list the order in which physical pages were assigned. When swapping has to happen, choose the first page in the list.
- **LRU**: This could be done only if accesses to page are tracked (since LRU depends on the page that was not recently accessed).
  - However, data is accessed directly from hardware and there is no way for the software to know if a page was accessed or not.
  - To alleviate this issue, hardware (cpu) support is provided. CPU sets an access bit (or use bit or reference bit) when a page is accessed (read or write) by the cpu. This bit is present in the PTE along with valid and present bit.
  - Scanning through the entire memory to identify pages with access bit set is zero is really hard (takes huge amount of time).
  - Clock Algorithm (Approximate LRU)

More paging related concepts

- Demand Paging; Thrashing

Questions

- What happens when swap area is full?
- Which are the possible valid and present bit combination? \(<0, 0>\); \(<1,1>\); \(<1, 0>\); \(<0, 1>\)
- After page fault is serviced, the instruction that caused the fault (or page miss) gets re-executed - Why?
- What are the advantages of background processes like swap daemon?
- What are the different ways to improve cache hit ratio?
- Is it possible to implement LRU policy without hardware support?