CSE 3201

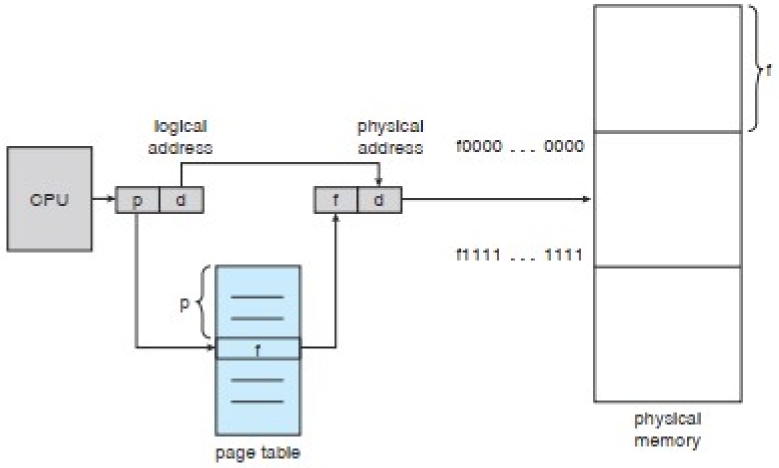
OPERATING SYSTEMS

Paging

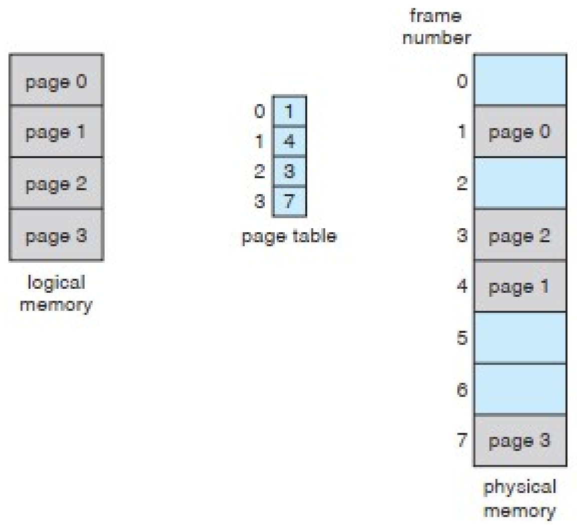
- Paging is a memory-management scheme that permits the physical address space of a process to be noncontiguous.
- Paging avoids external fragmentation and the need for compaction.

- When some code fragments or data residing in main memory need to be swapped out, space must be found on the backing store.
- The backing store has the same fragmentation problems discussed in connection with main memory

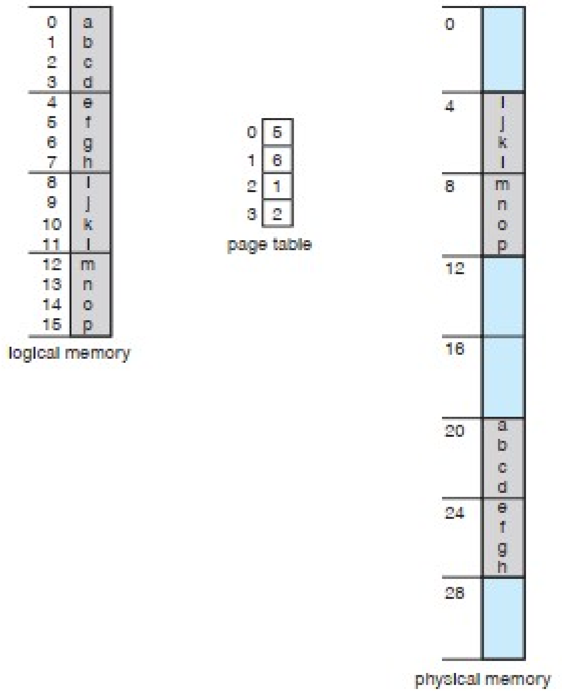
Paging Hardware



Paging mechanism



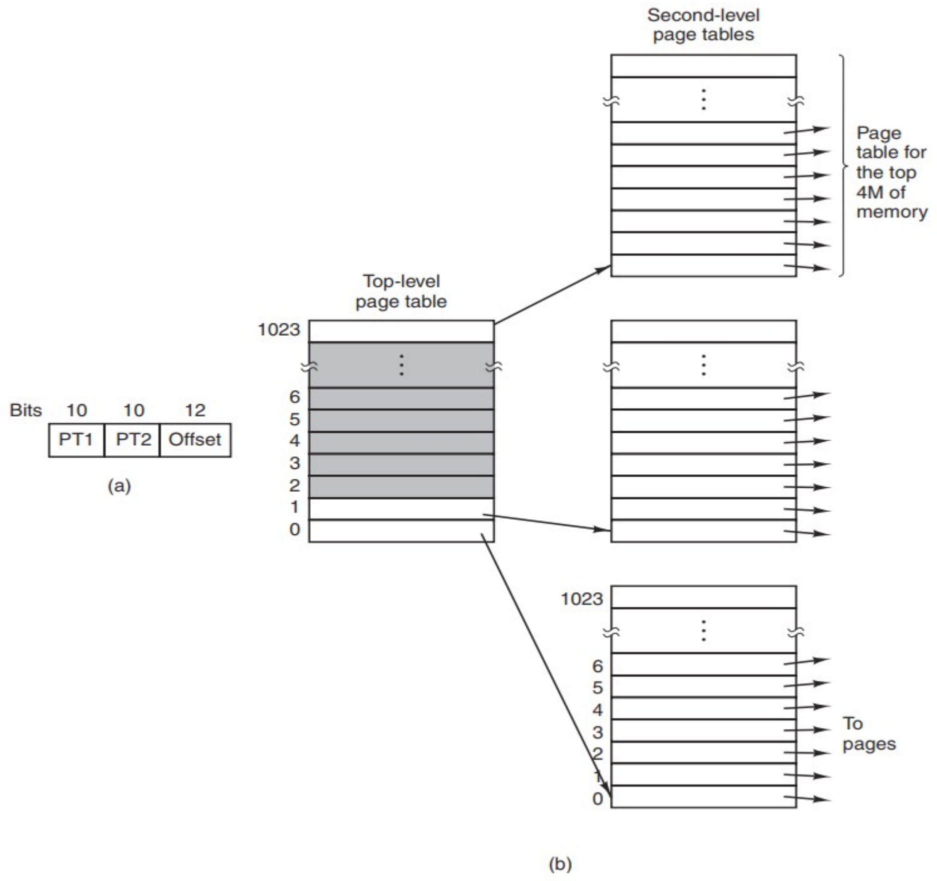
Paging Model of logical and Physical memory



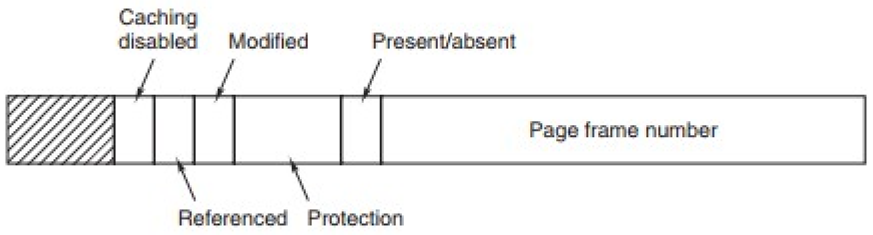
Paging with 32 byte memory with 4 byte paging

- There is no external fragmentation, but internal fragmentation exists.
- Expected fragmentation is average one-half page per process.
- Smaller page size is desirable but it has overhead.
- With larger page size disk IO is more efficient.

Multi level page table



Page Table entry structure



Present/absent –

1 – the entry is valid and can be used

0 – virtual page to which the entry belongs and is not currently in memory.

Protection –

1 – read only

0 – read/write

Modified and Referenced – keeps track of memory usage.

Caching –

Important for pages that map to register rather than memory.

Page Fault

- What happens if the program references an unmapped address?
  - The MMU notices that the page is unmapped and causes the CPU to trap to the operating system. This trap is called a page fault.
- The operating system picks a little-used page frame and writes its contents back to the disk (if it is not already there).
- It then fetches (also from the disk) the page that was just referenced into the page frame just freed, changes the map, and restarts the trapped instruction.

The Optimal Page Replacement Algorithm

- The page with the highest label should be removed.
- Suppose, one page will not be used for **8 million instructions** and another will not be used for **6 million instructions**.
- This algorithm is not realizable as we don’t know which page is going to be used next.

The Not Recently Used Page Replacement Algorithm

- The **Referenced** and **Modified** bit are used to decide the Not Recently Used Page.
- R is set whenever the page is referenced (read or written). M is set when the page is written to (i.e., modified).
- When a process is started up, all of its page table entries are marked as not in memory. As soon as any page is referenced, a page fault will occur.
- The operating system then sets the R bit (in its internal tables), changes the page table entry to point to the correct page, with mode READ ONLY, and restarts the instruction.
- If the page is subsequently modified, another page fault will occur, allowing the operating system to set the M bit and change the page’s mode to READ/WRITE.
- Periodically (e.g., on each clock interrupt), the R bit is cleared, to distinguish pages that have not been referenced recently from those that have been.

The Not Recently Used Page Replacement Algorithm

- When a page fault occurs, the operating system inspects all the pages and divides them into four categories based on the current values of their R and M bits:
  - Class 0: not referenced, not modified.
  - Class 1: not referenced, modified.
  - Class 2: referenced, not modified.
  - Class 3: referenced, modified.
- The NRU (Not Recently Used) algorithm removes a page at random from the lowest-numbered nonempty class.

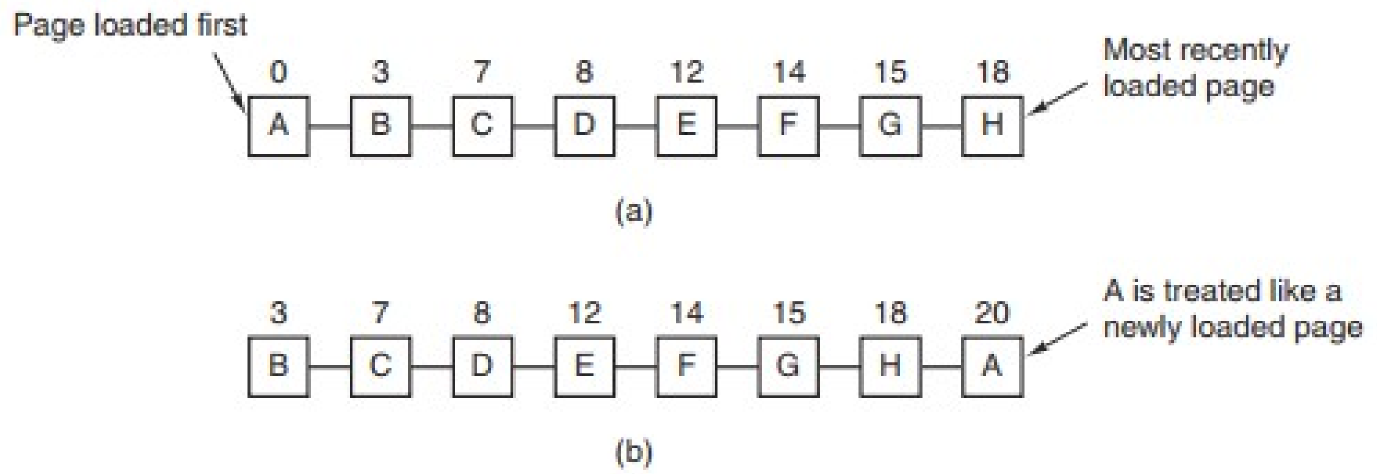
The First-In, First-Out (FIFO) Page Replacement Algorithm

- The operating system maintains a list of all pages currently in memory, with the most recent arrival at the tail and the least recent arrival at the head.
- On a page fault, the page at the head is removed and the new page added to the tail of the list.
- One problem exists. What it can be?

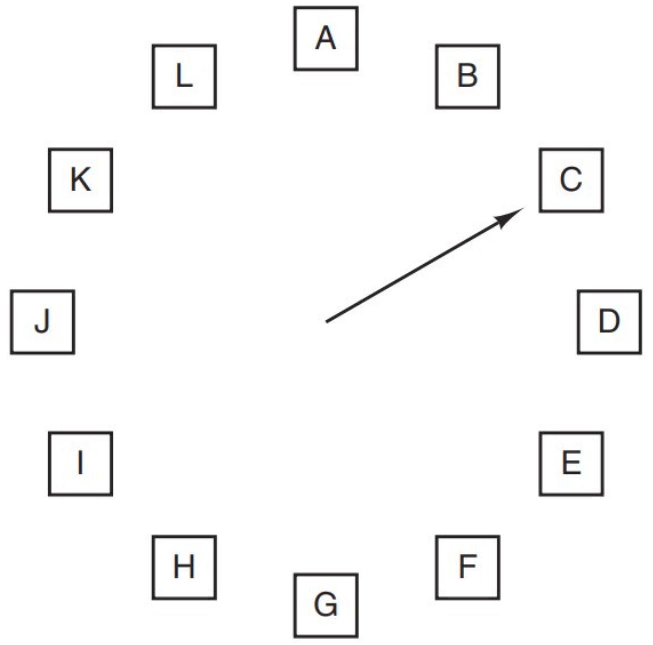
The Second-Chance Page Replacement Algorithm

- A simple modification to FIFO that avoids the problem of throwing out a heavily used page is to inspect the R bit of the oldest page.
- If it is 0, the page is both old and unused, so it is replaced immediately.
- If the R bit is 1, the bit is cleared, the page is put onto the end of the list of pages, and its load time is updated as though it had just arrived in memory. Then the search continues.

The Second-Chance Page Replacement Algorithm



The Clock Page Replacement Algorithm



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page
R = 1: Clear R and advance hand

The Least Recently Used (LRU) Page Replacement Algorithm

- Pages that have been heavily used in the last few instructions will probably be heavily used again soon.
- Pages that have not been used for ages will probably remain unused for a long time.
- When a page fault occurs, throw out the page that has been unused for the longest time.
- How can we find LRU pages?
  - One possible way is to implement hardware(!).
  - Maintain a counter for counting instructions. Add a special field to attach this number to each page table entry. What to do with this numbers?

Next

- More Page replacement algorithms