Topics covered –

1. Page Faults & Demand Paging
2. Copy On Write

Page Table Entry (PTE):

<table>
<thead>
<tr>
<th>D</th>
<th>U</th>
<th>R/W</th>
<th>P</th>
</tr>
</thead>
</table>

← 20 bit Physical Page No. →

Page Attributes:

Every page has a few attributes like the ones mentioned above. These bits are directly interpreted by the hardware.

U – User/Supervisor

R/W – Read Write

P – Present

D - Dirty

Page Fault: Page fault occurs in the middle of an instruction when the instruction refers to an unmapped address.

The page fault can be handled in the following ways:

1. Raise Segmentation Fault
2. Allocate memory for the page
3. Copy-on-Write
4. Go to disk.
**Allocate Memory for the page**

In case a process is trying to allocate memory and the current page doesn’t have as much free memory as requested, the Operating System would allocate one or more pages accordingly.

**Copy-on-write**

In case a process is trying to fork, the entire address space of the forking process need not be copied on to the child’s address space. Instead, all the read only pages would be shared. Whenever one of the processes modifies any shared page, another copy is created.
Go to disk

In case a page the process is requesting is not found on the physical memory, the OS would lookup the process’s page table to find if the page exists. And if it does, it would swap it in from the disk.

Demand Paging –

- Bring pages into memory as and when needed.
- If system is running low on memory, give used pages to other processes.
Page Replacement Algorithms:

1. FIFO (First In First Out)
2. LRU (Least Recently Used)
3. OPT (Optimal Page Trace)

**FIFO:** Pick the page which is oldest in terms of amount of time since the page was loaded in the memory and swap it.

**LRU:** Page is swapped on the basis of how long it has been since the page was accessed.

**OPT:** Swap out the page whose next use will occur farthest in future.

**Page Access Trace:** Keep track of all the memory accesses that the process did.

Consider the following example:

| Pages | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| FIFO  | 1 | 2 | 3 | 4 | 1 | 5 | 6 | 2 | 1 | 3 | 7 | 6 | 2 | 1 | 3 | 6 |
| (4 Hits) | (16 Misses) |
| LRU   | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 3 | 7 | 6 | 2 | 1 | 3 |
| (5 Hits) | (15 misses) |
| OPT   | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 1 | 3 | 6 | 2 | 1 | 3 | 6 |
| (9 Hits) | (11 Misses) |

Lecture error - 7 should have been evicted
Clock is a more efficient version of FIFO than Second-chance because pages don’t have to be constantly pushed to the back of the list, but it performs the same general function as Second-Chance. The clock algorithm keeps a circular list of pages in memory, with the "hand/head" pointing to the oldest page in the list. When a page fault occurs and no empty frames exist, then the R (referenced) bit is inspected at the hand’s location. If R is 0, the new page is put in place of the page the "hand" points to, otherwise the R bit is cleared. Then, the clock hand is incremented and the process is repeated until a page is replaced.