- Paging
- Segmentation

Real / Simple

All pages are of same size

All segments are of different size

p	d

logical address

search p in page table.

s	d

d → use → different values for each segment

check that 'd' is within the segment

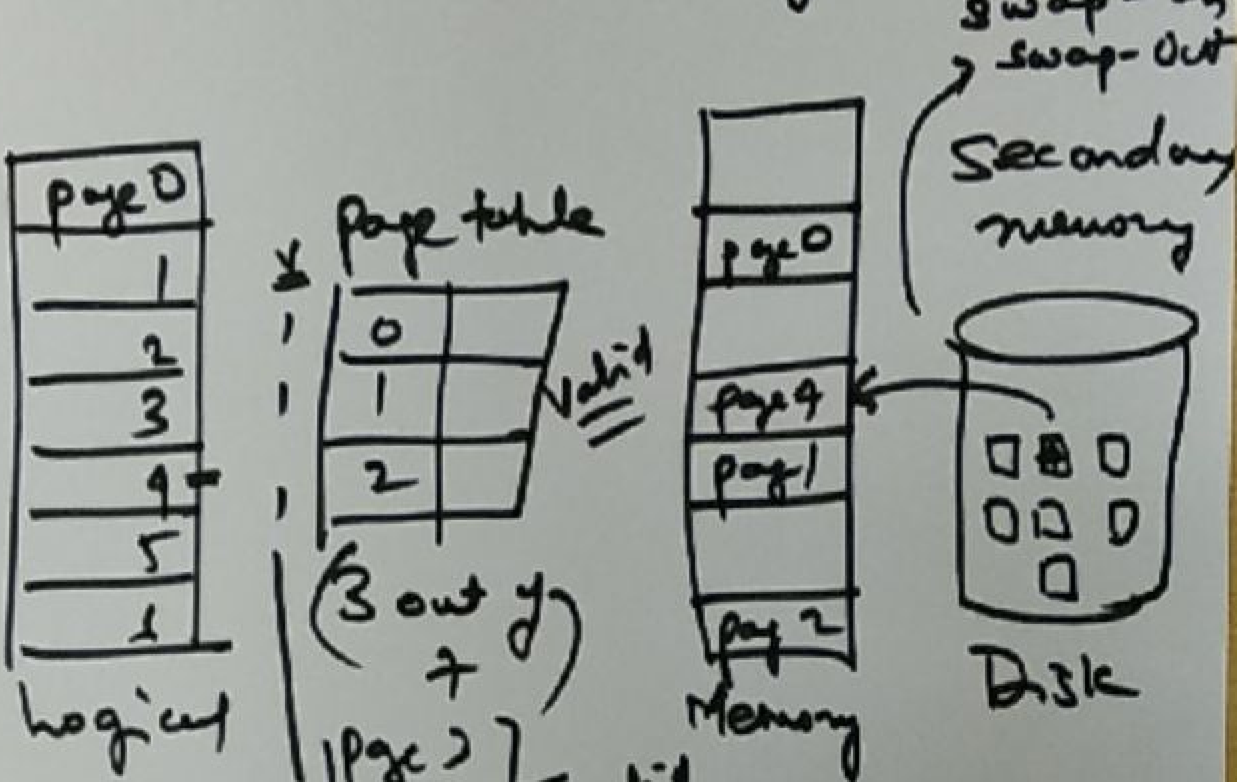
↓

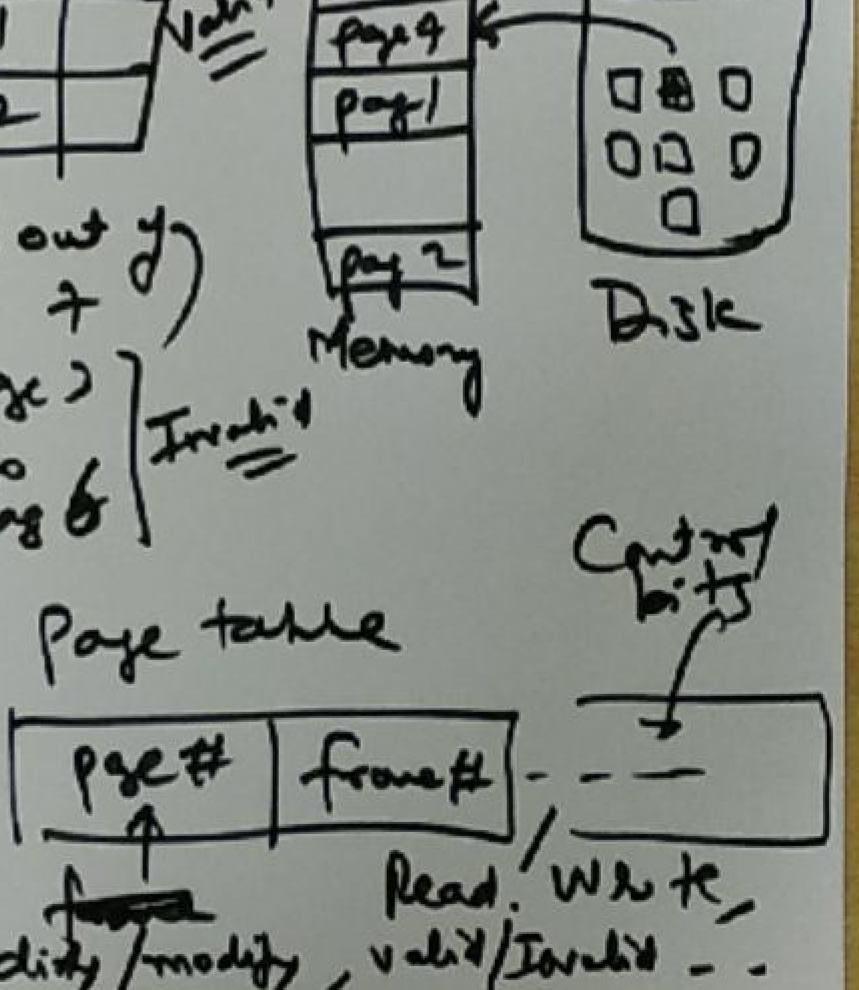
less than limit of segment

s → to search segment in segment table

base address + → physical address

- TLB / Associative memory
  - └ part of page table
- Virtual memory / Paging
  - \ Segmentation



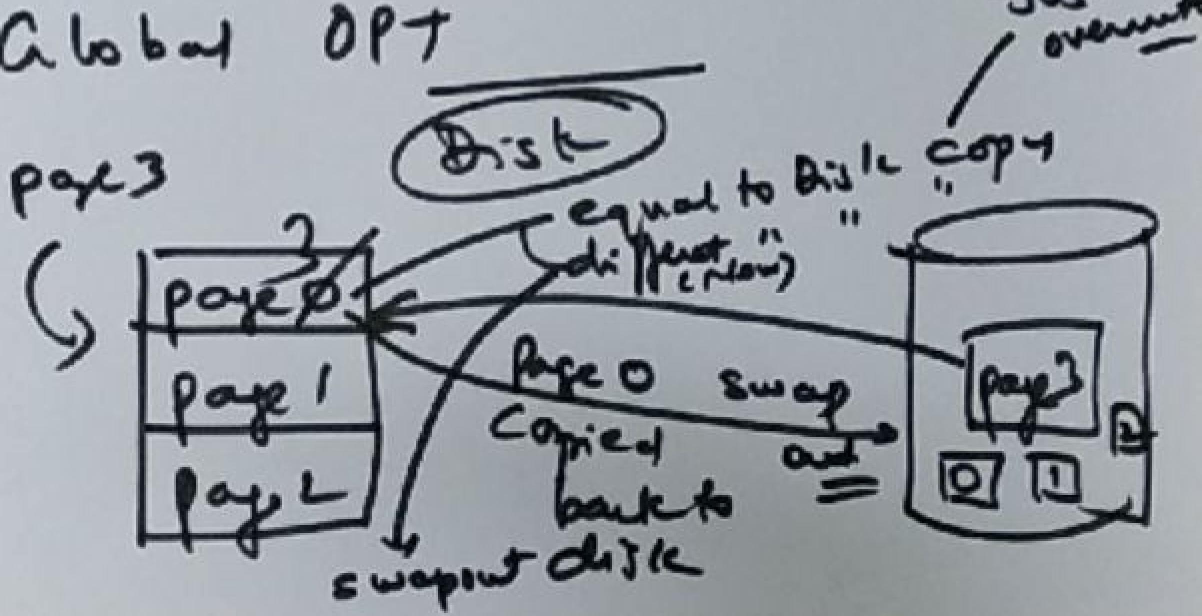


- New page that is to be loaded
  - Memory is available – ok
    - FFL has frames
  - Memory is not available
    - FFL = ∅

Page Replacement

- Global, Local
- +
- FIFO — First In First Out
- LRU — Least Recently Used
- OPT — optimal

Global FIFO
Local LRU
Global OPT



- Copy of page in memory has been written/modified after having loaded

  ↳ Modified page / Dirty page

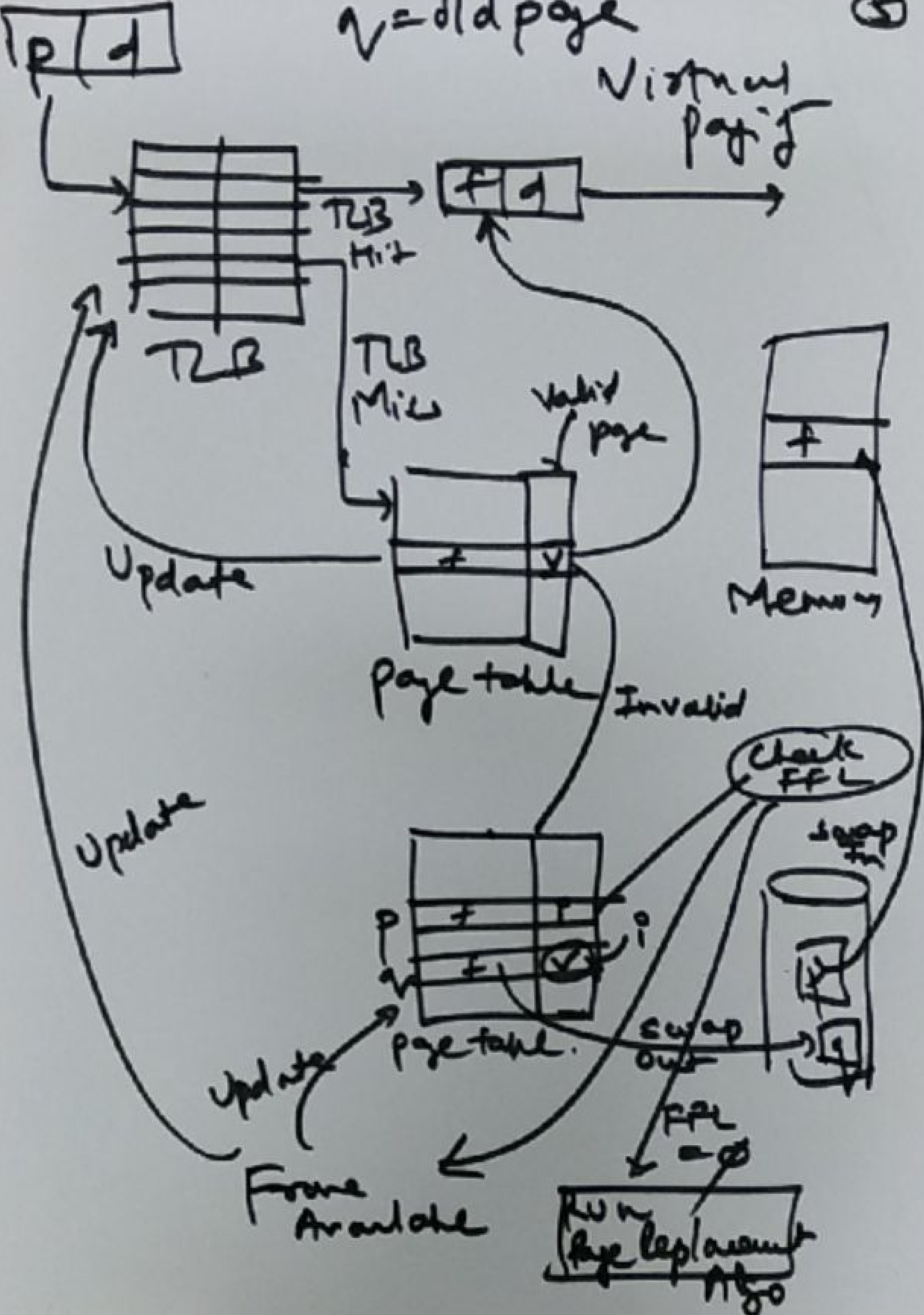
  ↳ copy old page back to disk (Swap out) ← Disk write

  ↳ bring/load the new page (Swap In) ← Disk read.

Summary

① If copy of page in memory is same —— Swap-In New page

② If copy of page in memory is different then on disk - Swap-Out, Swap-In


v = old page
p | d
Virtual page f
f | d
TLB Hit
TLB
TLB Miss
valid page
f
v
Update
Memory
Page table
Invalid
Check FFL
swap in
Update
p
f
i
q
f
v
i
page table.
swap out
Update
FFL = ∅
Frame Available
Run Page Replacement Algo

EAT = Hit + Miss

Real 100% —

↓

90% —— 10%

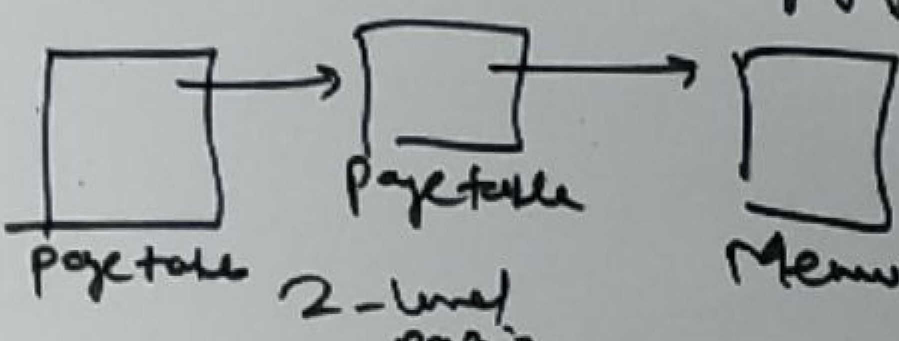
Virtual —— 80% —— 20%

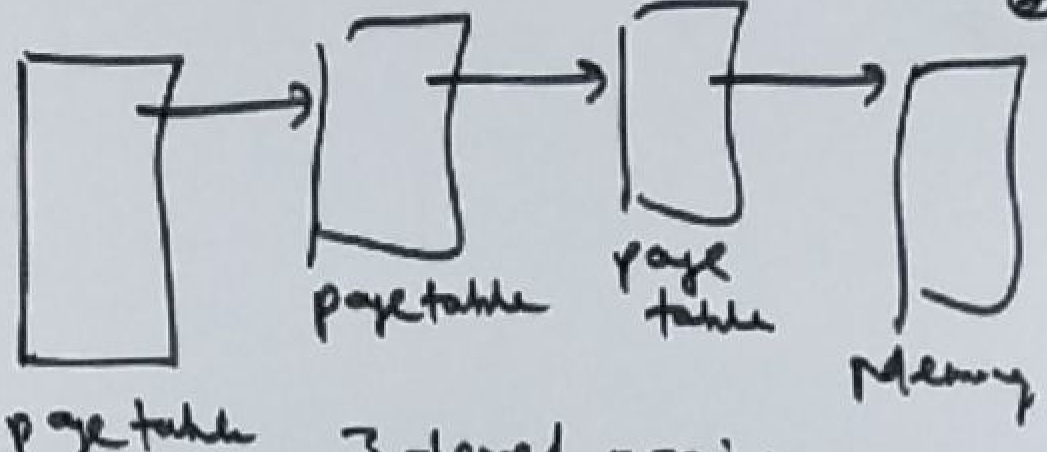
— Disk

- swapout, swapin
- x% — dirty page
- (1 − x) — swap-in

— Paging

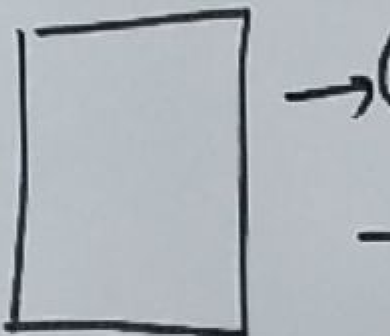
- Single-level paging
- Multi-level paging
  - k-level paging





3-level paging

- Very large page tables
  - decrease the scope of search

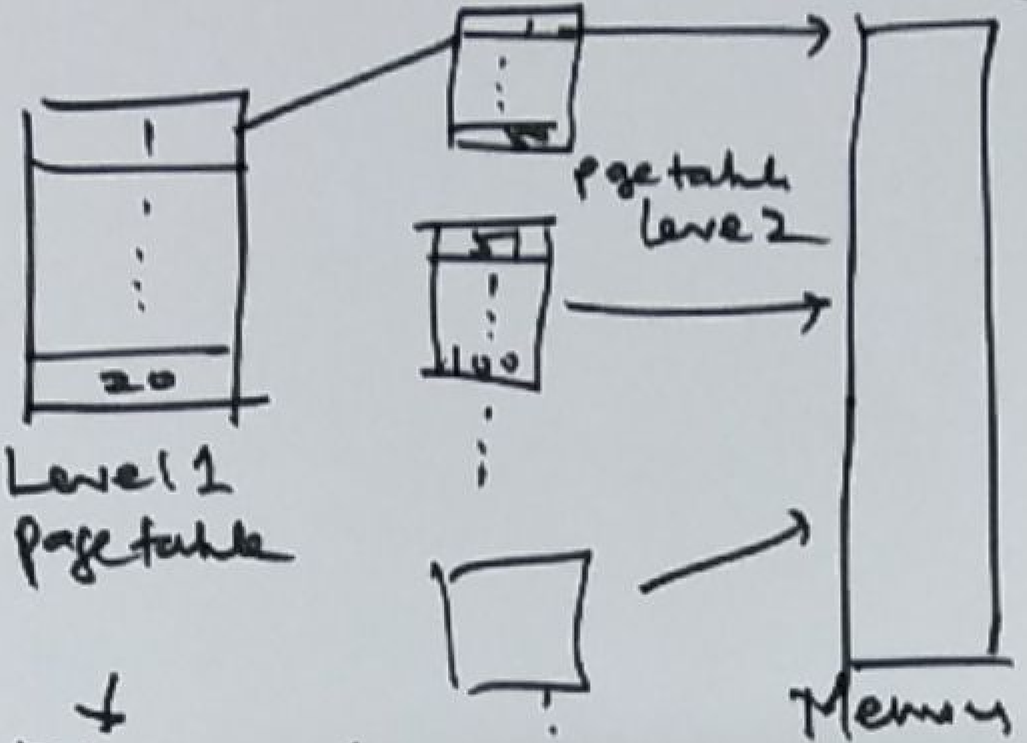


$\boxed{20}$ — points level 1

20 page table — 2-level paging

- 50 PTE each → level 2

PT (Single level) 1000 PTE



↓

EAT — (k+1) ma.

- Replacement Algos
  - Local — process specific
  - Global. — across all processes