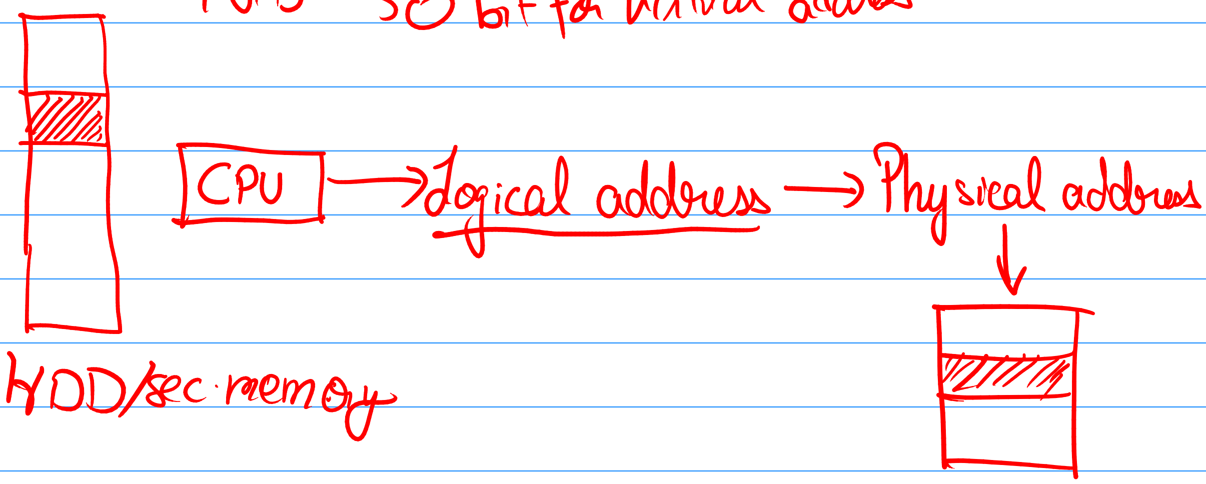
1GB 30 bit for virtual address

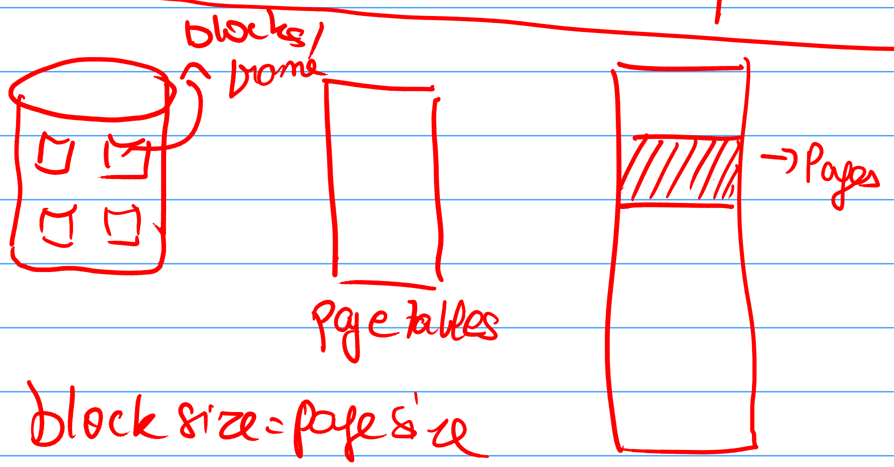


Virtual Memory

- Motivation: main memory acts as *cache for secondary storage*, e.g., magnetic disk
- *Virtual address space*, i.e., space addressable by a program is determined by ISA
  - e.g., 64-bit MIPS address space size is 2^{64} – recall jr instruction
  - typically: main memory size $\leq$ disk size $\leq$ virtual address space size
- Program can "pretend" it has main memory of the size of the disk – which is *smaller than* the*virtual memory* (= whole virtual address space), but *bigger than* the actual *physical memory* (=DRAM main memory)
  - *Page table* (as we shall see) transparently converts a virtual memory address to a physical memory address, *if the data is already in main*; *if not*, it issues call to OS to fetch the data from disk into main
- Virtual memory is organized in fixed-size (power of 2, typically at least 4 KB) blocks, called *pages*. Physical memory is also considered a collection of pages of the same size.
  - the unit of data transfer between disk and physical memory is a page

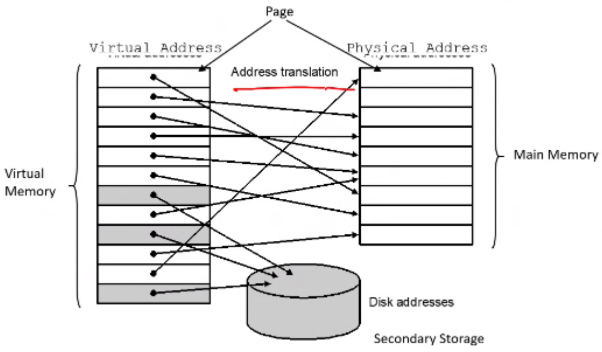
Demand paging – Bring in the page from SM whenever required!

logical to physical



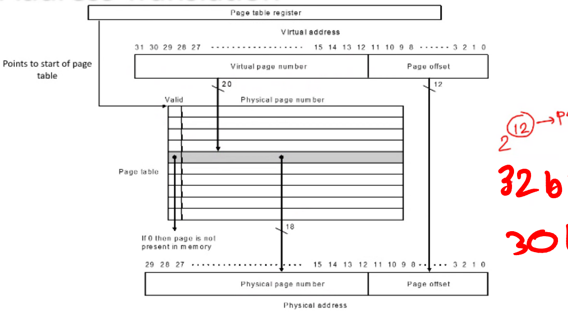
block size = page size

[block = frame]



Mapping of pages from a virtual address to a physical address or disk address

Page Table Implements Virtual to Physical Address Translation



Page table: page size 4 KB, virtual address space 4 GB, physical memory 1 GB

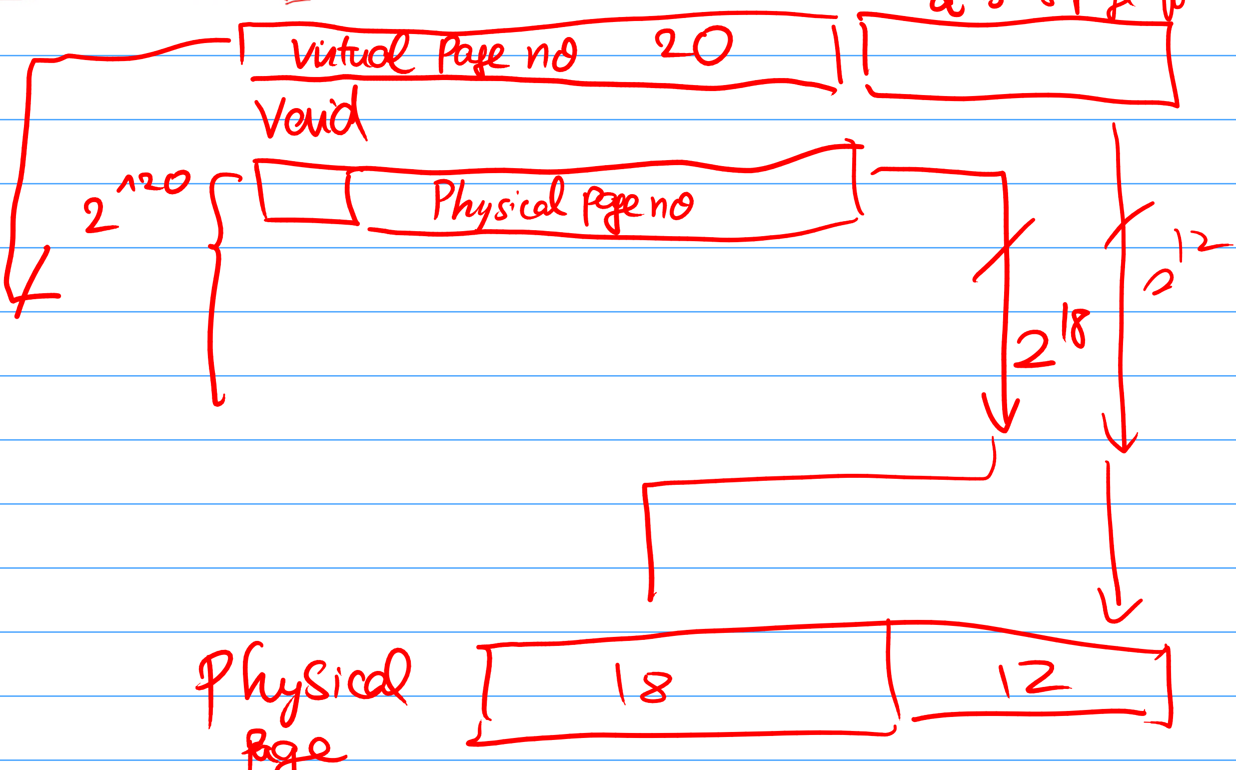
$2^{(12)} \rightarrow$ page offset

32 bits — VAS

30 bits → PAS

↳ Physical address space

12 bits page offset



Size $= 2^{20} \times (18+1)$

We prefer write back over write through
Since write back is better in terms of latencies

Write back may cause a problem in redundant disks

TLB

[Page table register]

① Common Page table (Too large) → requires vast contiguous space
② Separate page table for each process (lower access time)

32 bit virt mem
4kB Page size → 2^{12}
4 bytes page table entry
Find size

$2^{20} \times 4$
= 4MB page size

4 MB page size is huge

⇒ Each program has its own page table
& the page table register points to the start of the program's page table (PTR)
= Page table register

- No. of page table entries = address space size / page size

  $= 2^{32} / 2^{12} = 2^{20}$
- Size of page table = No. of entries × entry size

  $= 2^{20} \times 4$ bytes = 4 MB (*huge*!)

PTR

- Note, to avoid large page table size:
  - each program has its own page table
    - *page table register* points to start of program's page table
  - to reduce storage required per program page table
    - page table for a program covers the span of virtual memory containing its own code and data
    - other techniques, e.g., multiple-level page tables, hashing virtual address, etc.

Making Address Translation Fast with the Translation-lookaside Buffer

To find page is present

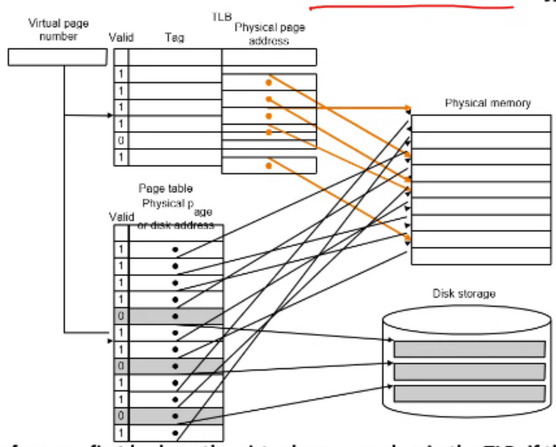
T_{MM} → memory Access

+

T_{MM} – Page table

2 Times memory access

- A *cache* for address translations – *translation-lookaside buffer (TLB)*:



On a page reference, first look up the virtual page number in the TLB; if there is a TLB miss look up the page table; if miss again then true page fault

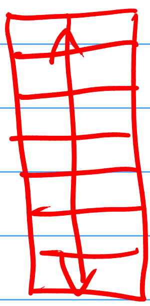
Page Faults → Page not in memory, retrieve bring it from disk

Enormous miss penalty ~10^6 cycles

⇒ 32 or 64 kB Page Size (large)

All filled

what to replace?



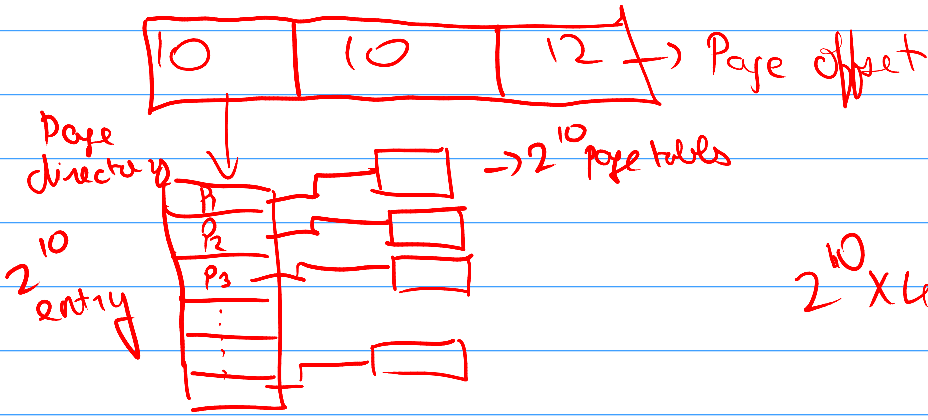
1. FCFS (FIFO)
2. LRU
3. Optimal page replacement

(theoretical, best performance in terms of less page misses)

6 is not used anytime soon, it's least used in the future
but we cannot see the future

$2, 4, 6, \overbrace{2, 5, 7, 2, 4, 5,}\ 8$

6 → replace



$2^{10} \times 4$ bytes (lets say)

we may require all 2^{10} entries in page directory but not all the entries in page tables all at once

→ reducing the no. of page tables in practice

eg: 4 processes will have 4 directory entries

⇒ 4×2^{10} page table entries

if 1 page table entry was 4 bytes long

$8 \times 2^{10} = 8KB$

otherwise $2^{20} \times 4 = 4MB$ (too much)

Each process will need its own page table

Page directory is common though, each pointing to a page table of 4kB size.