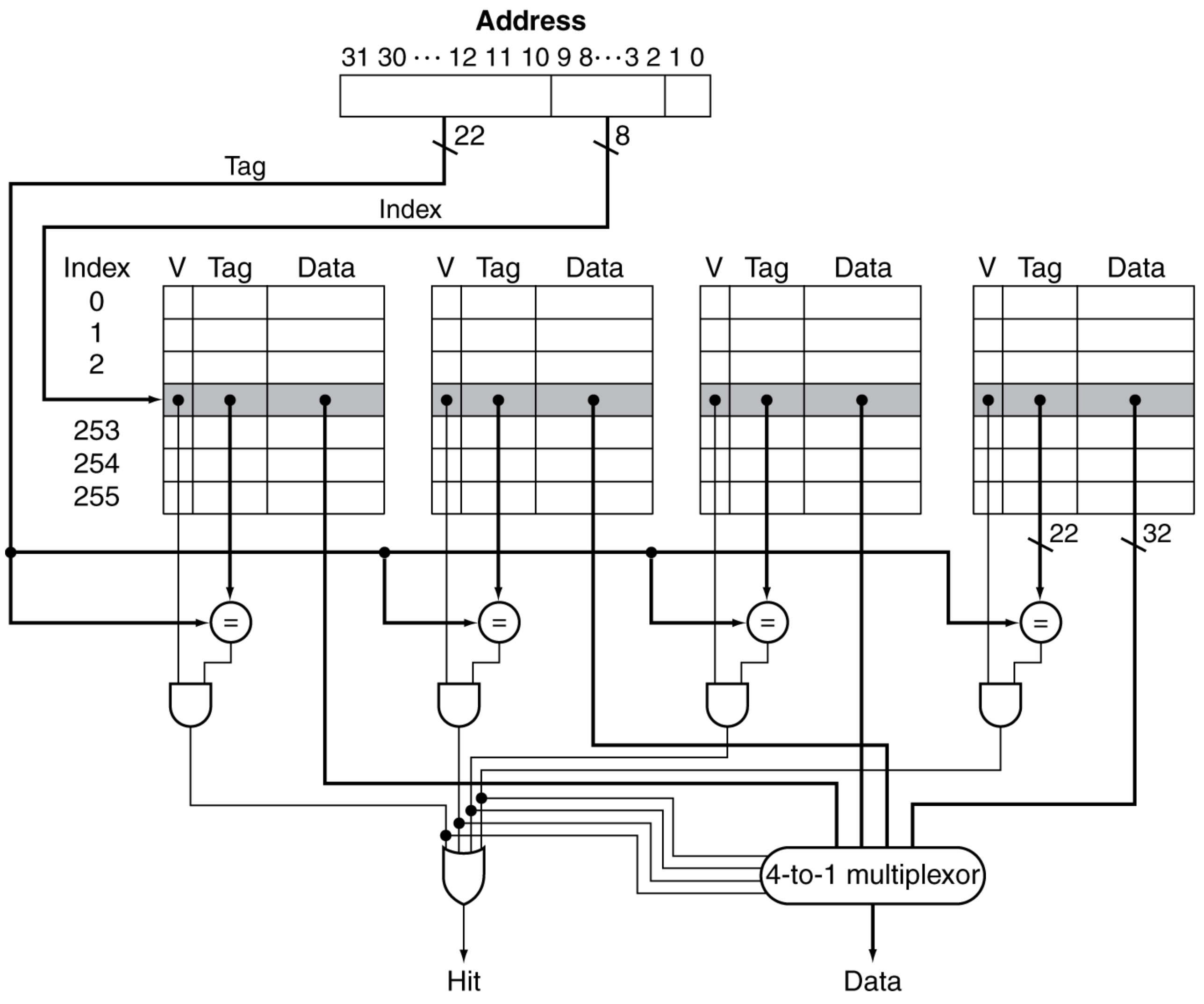
Memory Management

How Much Associativity

- Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000: (data miss rate)
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Set Associative Cache Organization



Associativity Implies Choices

- Direct-mapped
  - Compare addr. with only one tag
  - Location A can be stored in exactly one cache line
- N-way set-associative
  - Compare addr. with N tags simultaneously
  - Location A can be stored in exactly one set, but in any of the N cache lines belonging to that set
- Fully associative
  - Compare addr. with each tag simultaneously
  - Location A can be stored in any cache line

Replacement Policy

- Direct mapped: no choice

- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set

- Optimal policy: Replace the block that is accessed furthest in the future
  - Requires knowing the future

- Predict the future from looking at the past
  - If a block has not been used recently, it's often less likely to be accessed in the near future (a locality argument)

- Least-recently used (LRU)
  - Choose the one unused for the longest time

Write Policy

- Write-through:
  - CPU writes are cached, but also written to main memory immediately
  - Stalling the CPU until write is completed
  - Simple, slow
- Write-back:
  - CPU writes are cached, but not written to main memory until we replace the block
  - Commonly implemented in current systems
  - Fast, more complex

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

Write-Back with ‘Dirty’ Bits

- Add 1 bit per block to record whether block has been written to.
- Only write back dirty blocks

Virtual Memory

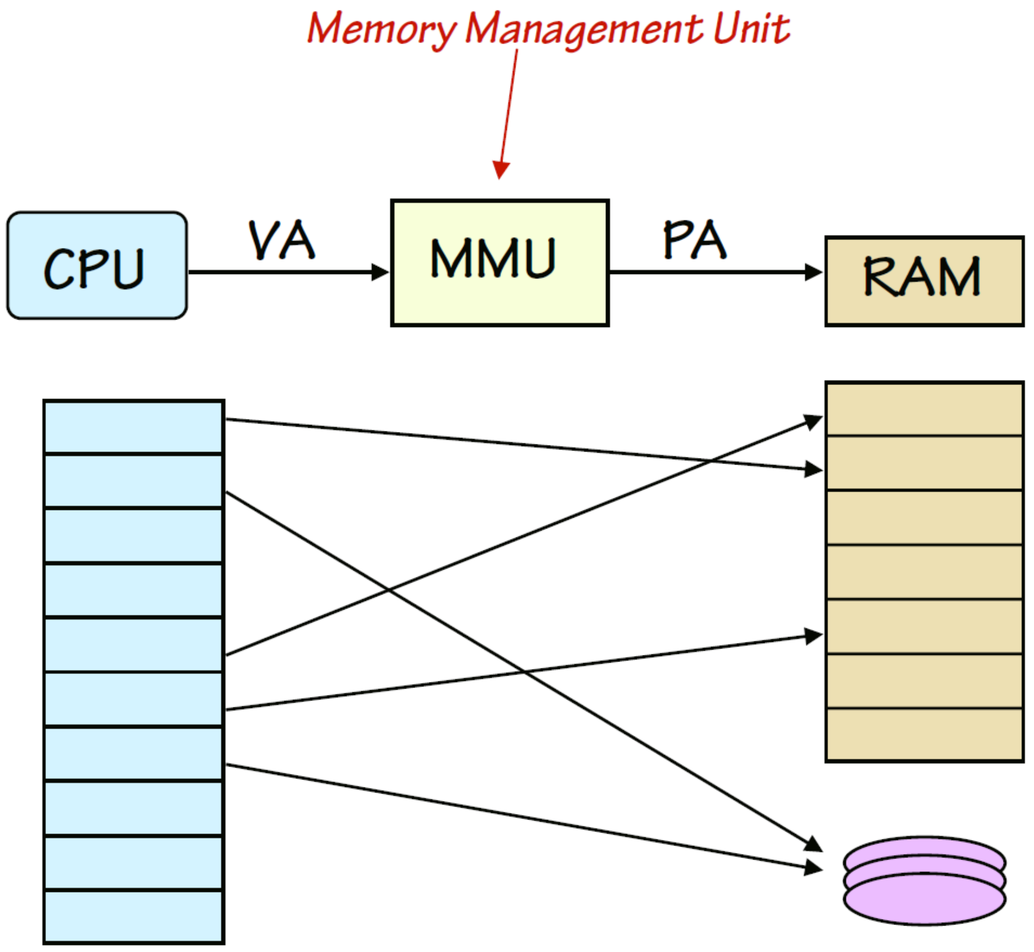
- Software caches:
- Same objective: fake large, fast, and cheap memory
- Conceptually similar
- Different implementations

Virtual Memory

- Use main memory as a "cache" for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

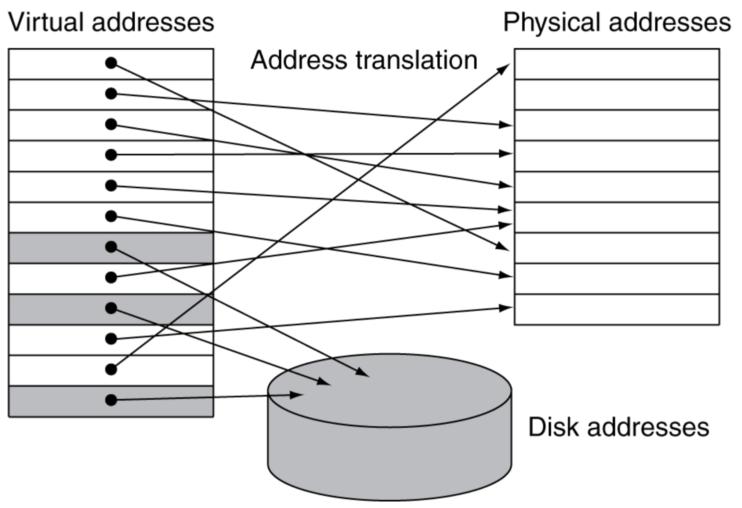
Virtual Memory

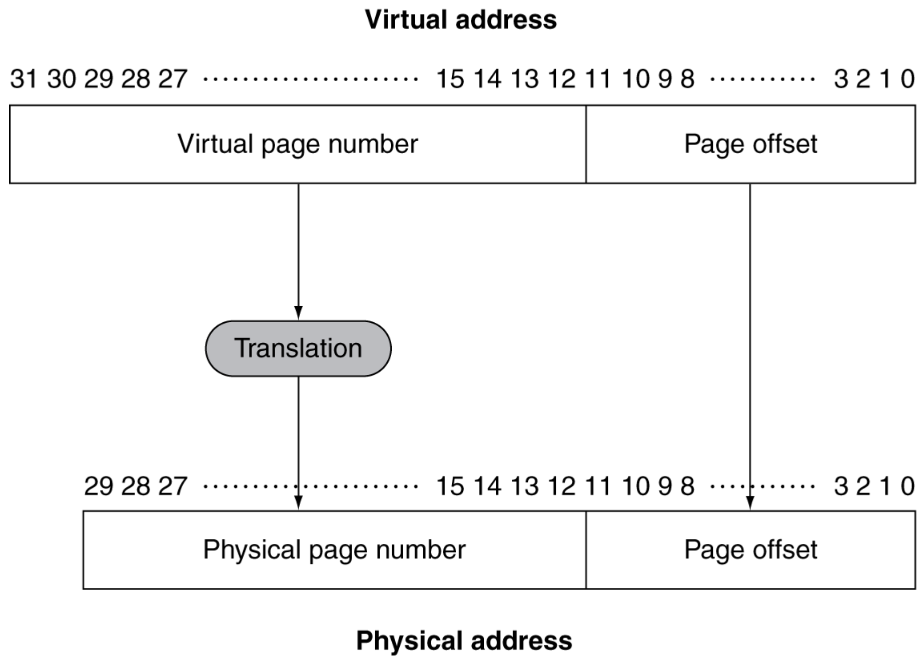
- Two kinds of addresses:
  - CPU (also programs) uses virtual addresses
  - Main memory uses physical addresses
- Hardware translates virtual addresses to physical addresses via an operating system(OS)-managed table, the page map or page table.
- The price of VM is address translation


Memory Management Unit
CPU
VA
MMU
PA
RAM

Address Translation

- Assume we have 1 GB main memory: how many bits required to represent physical address?
- Fixed-size pages (e.g., 4K): how many bits to describe page size?





Virtual Memory: Paging

- Divide physical memory in fixed-sized blocks, called pages
  - Typical page size: 4~16KB
  - Virtual address: Virtual page number + offset bits
  - Physical address: Physical page number + offset bits
- MMU maps virtual address to physical address
  - Use page table (or page map) to perform translation
  - Miss: or called a **page fault** if no translation, which means the page is not present in main memory

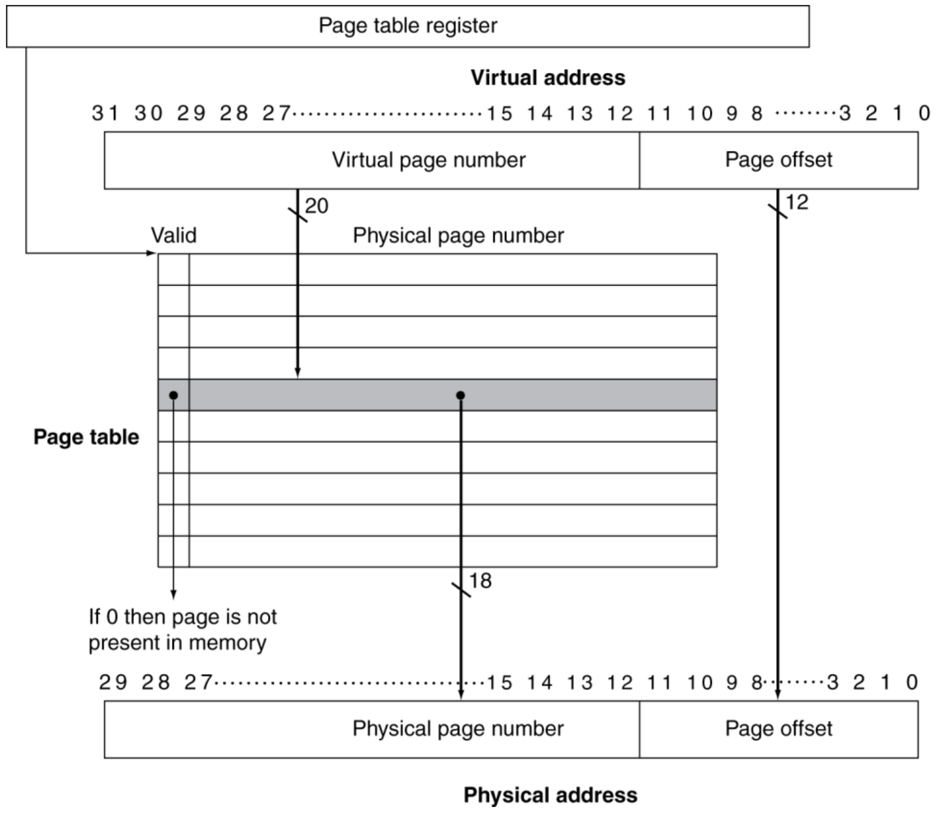
Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code

- Try to minimize page fault rate
  - Fully associative placement
  - Smart replacement algorithms

Page Tables

- Stores placement information
  - An array of page table entries(PTE), indexed by virtual page number
  - Page table register in CPU points to page table

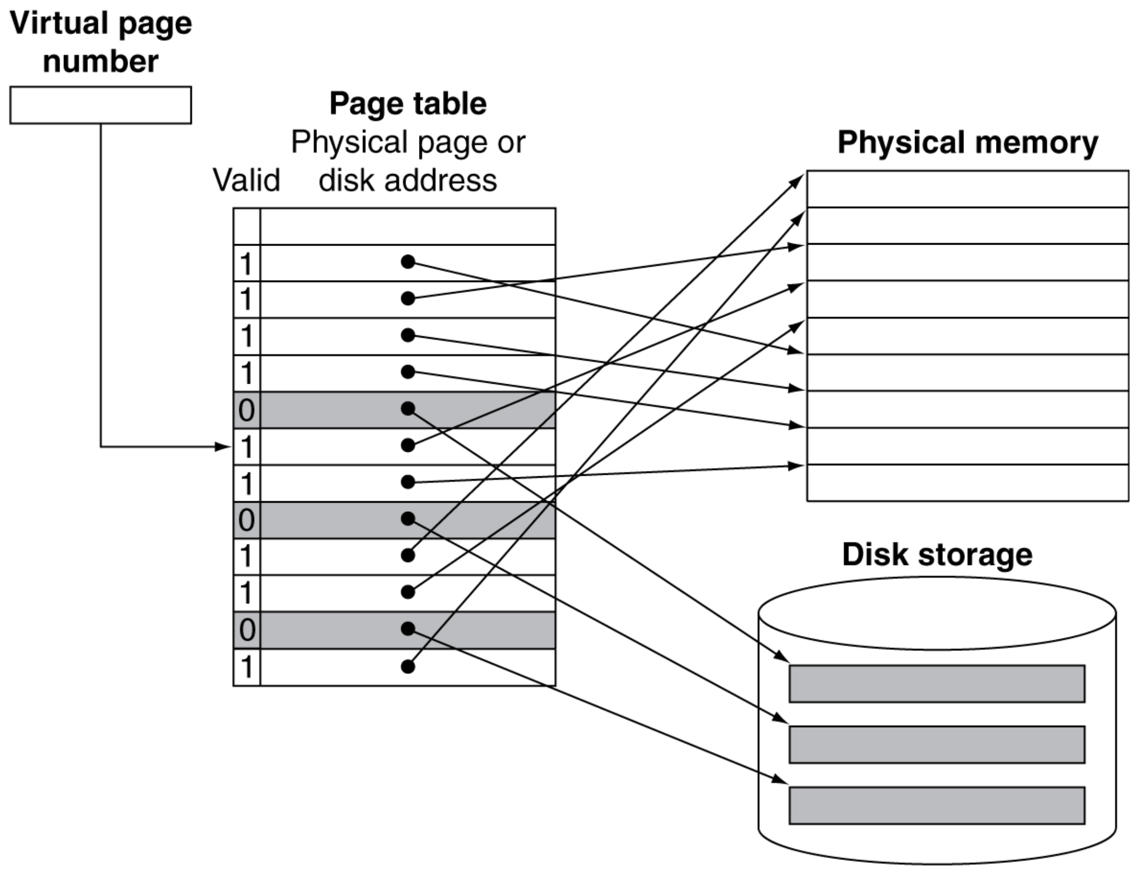


Page Tables

- If page is present in memory
  - PTE(Page Table Entry) stores the physical page number
  - Plus other status bits (valid, …)
- If page is not present in memory
  - look for the page on disk

Page Tables

- Virtual page number serves as the index
- What if page fault and there is no empty page in physical memory.

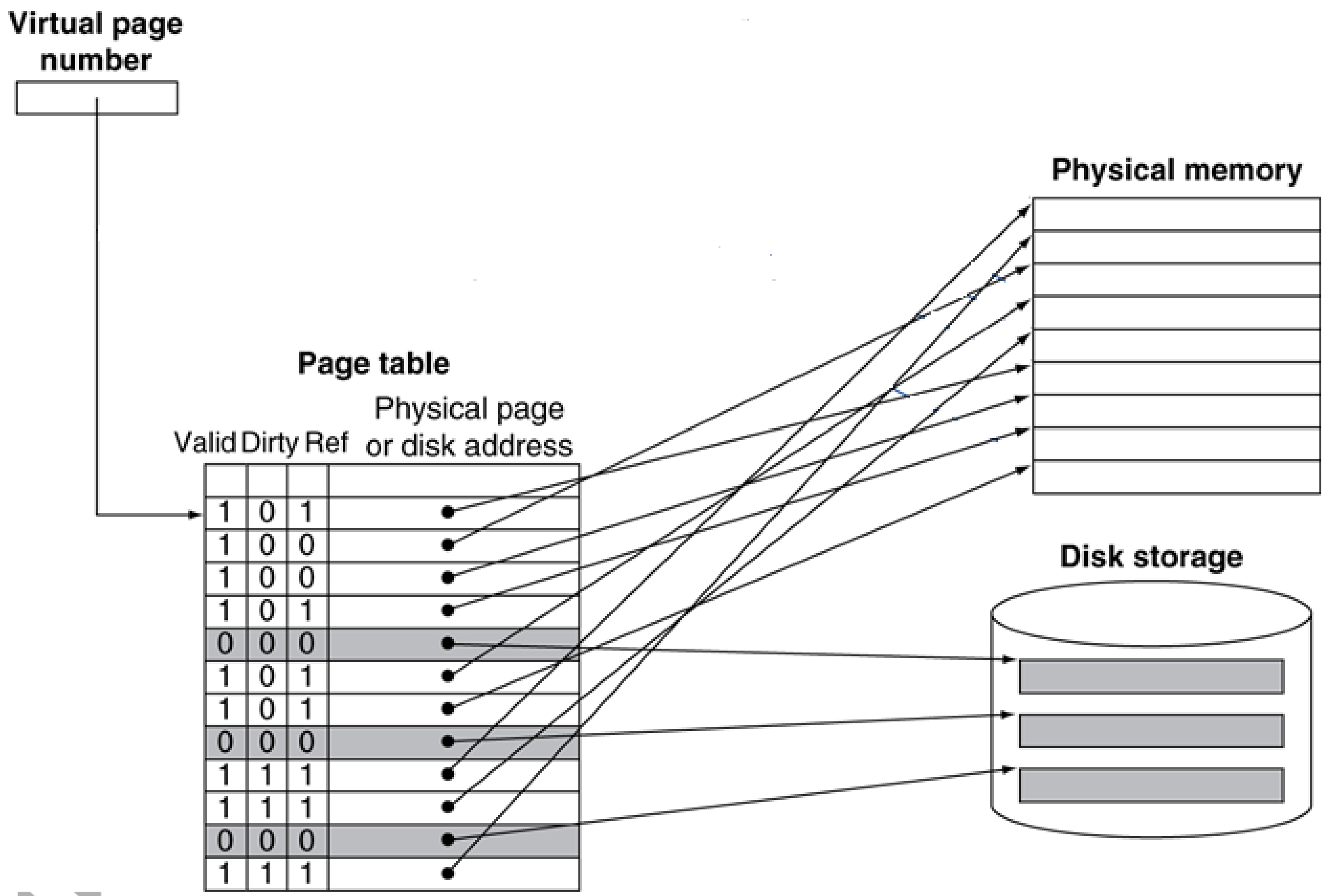


Replacement and Writes

- Replacement: To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Use write-back
  - Dirty bit in PTE set when page is written
- Two more bits for each entry in the page table

Page Tables

- Physical address m + p: m #bits represent page #
- How many bits in the page table:



Page Fault Handler

- Use virtual address to find PTE
- Locate page on disk
- Choose a page in memory to replace
  - If dirty, write to disk first
- Read page into memory and update page table
- Make process runnable again

Page Map Arithmetic

(v+p)	bits in virtual address
(m+p)	bits in physical address
2^v	# of virtual pages
2^m	# of physical pages
2^p	bytes per physical page
2^{m+p}	bytes in physical memory
$(m+3) *2^v$	bits in the page table:

- typical page size: 4~16KB
- typical (v+p): 32 bit, 64 bit
- typical (m+p): 30~40 bits (1GB~1TB)

Example:Page Map Arithmetic

- Suppose:
  - 32-bit virtual address
  - Page size: 4KB
    - 2^{12}
  - RAM max: 1GB
    - 2^{30}
- Then:
  - # Physical pages: 2^{18} 256K
  - # Virtual pages: 2^{20}
  - # PTE (page table entries): 2^{20}
  - # bits in page table: $(18+3)*2^{20}$
- Use SRAM for page table???
  - 21Mbit ➔ 3MB