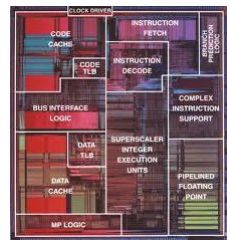
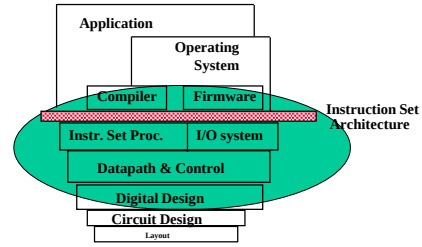
UTD

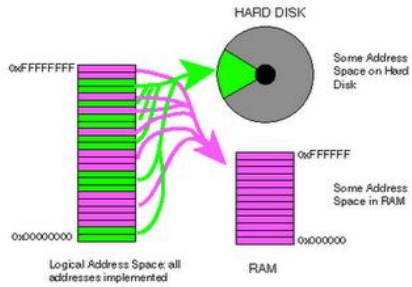
CS/SE 3340
Computer Architecture





Memory Hierarchy – Virtual Memory

Adapted from slides by Profs. D. Patterson and J. Hennessey



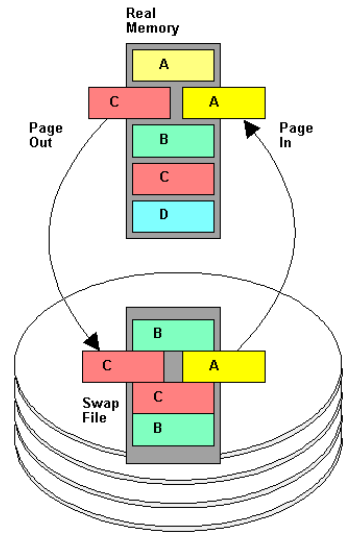
Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	5
2	2	2	2	2	2
3	5	5	4	4	4
4	4	3	3	3	3

- Counter implementation
  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
  - When a page needs to be changed, look at the counters to determine which are to change

From Computer Desktop Encyclopedia
© 2008 The Computer Language Co. Inc.



2

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*

3

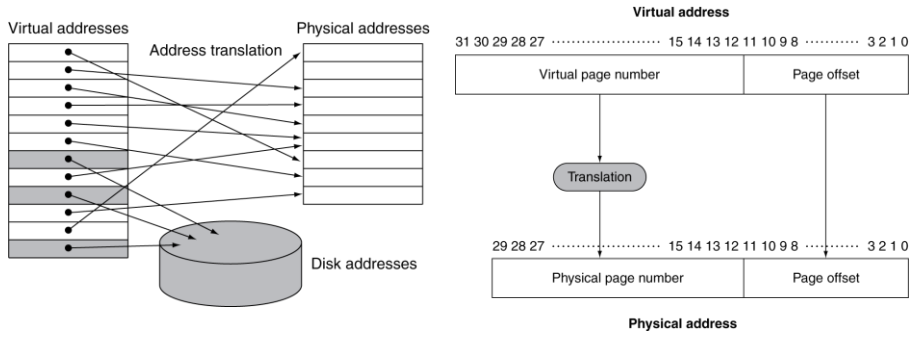
Sources of Misses

- *Compulsory misses (aka cold start misses)*
  - First access to a block
- *Conflict misses (aka collision misses)*
  - Due to competition for entries in a set
  - In a non-fully associative cache only, would not occur in a fully associative cache of the same total size
- *Capacity misses*
  - Due to finite cache (physical memory) size
  - A replaced block is later accessed again

4

Address Translation

- Fixed-size pages (e.g., 4K)



Page Fault Penalty

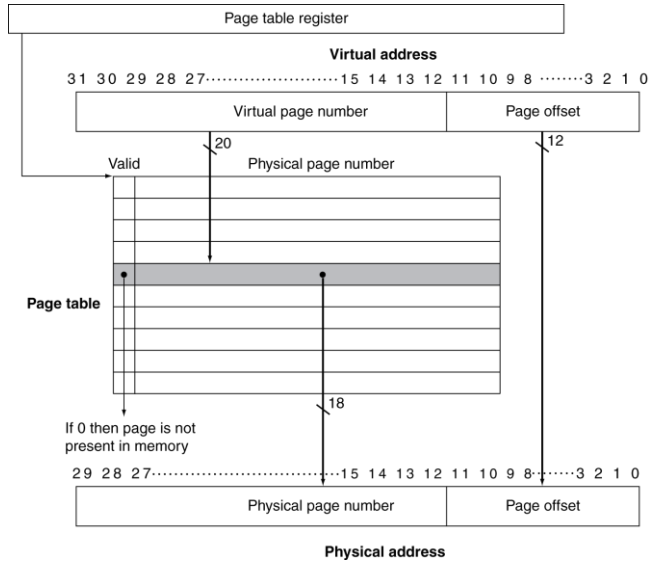
- On *page fault*, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code
- Try to *minimize page fault rate*
  - Fully associative placement
  - Smart replacement algorithms

Page Tables

- Stores placement information
  - Array of *page table entries (PTE),* indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

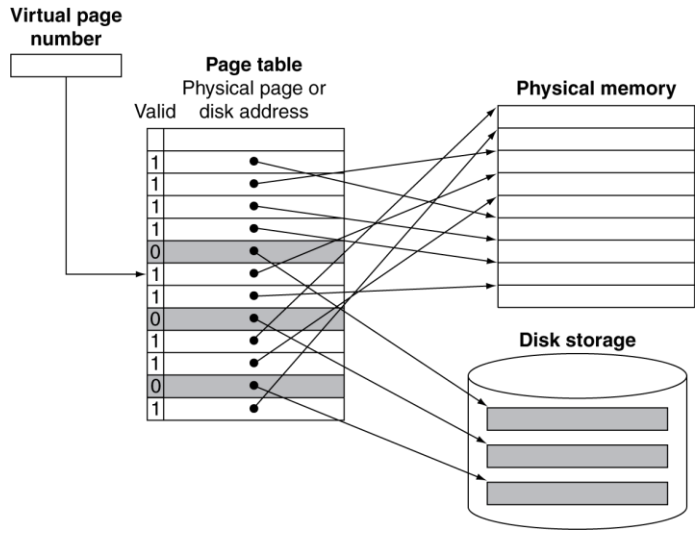
7

Translation Using a Page Table



8

Mapping Pages to Storage



Replacement and Writes

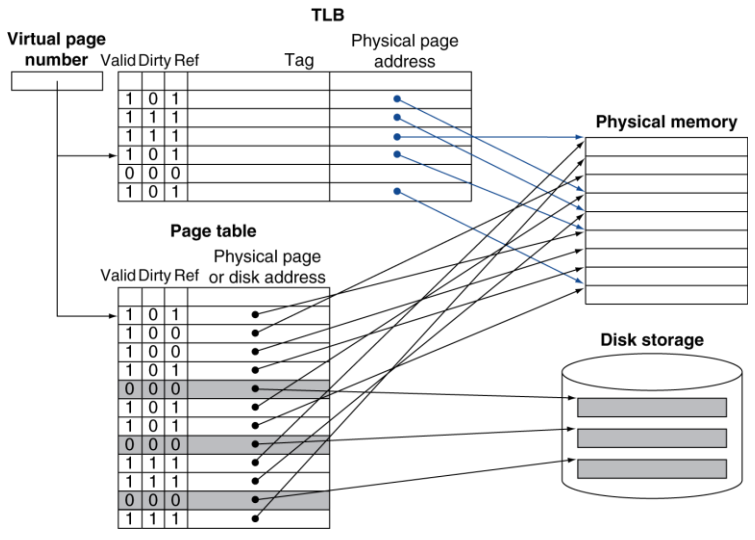
- To reduce page fault rate, prefer *least-recently used (LRU)* replacement
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write through is impractical
  - Use write-back
  - Dirty bit in PTE set when page is written

Fast Translation Using a TLB

- Address translation would appear to require extra memory references
  - One to access the PTE
  - Then the actual memory access
- But access to page tables has good *locality*
  - So use a fast cache of PTEs within the CPU
  - Called a Translation Look-aside Buffer (TLB)
  - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate
  - Misses could be handled by hardware or software

11

Fast Translation Using a TLB



12

TLB Misses

- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

13

TLB Miss Handler

- TLB miss indicates
  - Page present, but PTE not in TLB
  - Page not present
- Must recognize TLB miss before destination register overwritten
  - Raise exception
- Handler copies PTE from memory to TLB
  - Then restarts instruction
  - If page not present, page fault will occur

14

Page Fault Handler

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

15

Memory Protection

- Different tasks can share parts of their virtual address spaces
  - But need to protect against errant access
  - Requires OS assistance
- Hardware support for OS protection
  - Privileged supervisor mode (aka kernel mode)
  - Privileged instructions
  - Page tables and other state information only accessible in supervisor mode
  - System call exception (e.g., syscall in MIPS)

16

The Memory Hierarchy

- Common principles apply at all levels of the memory hierarchy
  - Based on notions of caching
- At each level in the hierarchy
  - Block placement
  - Finding a block
  - Replacement on a miss
  - Write policy

17

Block Placement

- Determined by associativity
  - *direct mapped*
    - 1-way associative (# of sets = # of cache blocks)
    - 1 block associated with 1 set
  - *n-way* set associative
    - n choices within a set
    - # of sets = # of cache blocks / n
  - *fully* associative
    - All cache blocks are associated with one set
- Higher associativity reduces miss rate
  - Increases complexity, cost, and access time

18

Finding a Block

Associativity	Location method	Tag comparisons
Direct mapped	Index	1
n-way set associative	Set index, then search entries within the set	n
Fully associative	Search all entries	#entries
	Full lookup table	0

- Hardware caches
  - Reduce comparisons to reduce cost
- Virtual memory
  - Full table lookup makes full associativity feasible
  - Benefit in reduced miss rate

19

Replacement

- Choice of entry to replace on a miss
  - Least recently used (LRU)
    - Complex and costly hardware for high associativity
  - Random
    - Close to LRU, easier to implement
- Virtual memory
  - LRU approximation with hardware support

20

Write Policy

- *Write-through*
  - Update both upper and lower levels
  - Simplifies replacement, but may require write buffer
- *Write-back*
  - Update upper level only
  - Update lower level when block is replaced
  - Need to keep more state
- Virtual memory
  - Only write-back is feasible, given disk write latency

21

Cache Design Trade-offs

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty. For very large block size, may increase miss rate due to pollution.

22

Summary

- Virtual memory increases address space of a computer system by using secondary storage (e.g. hard disk)
  - Main memory used as ‘cache’ for secondary storage!
  - Page-in and page-out
  - Page fault
- Page fault causes performance hit
  - Data needs to be read from slow secondary storage to main memory
  - Pages in main memory are replaced
- Page replacement strategies
  - LRU – Least Recently Used
  - Random