
IC220 Set #18: Caching Finale and Virtual Reality (Chapter 7)

1

Cache Performance

- Simplified model:

$$\begin{aligned} \text{execution time} &= (\text{execution cycles} + \text{stall cycles}) \times \text{cycle time} \\ &= \text{execTime} + \text{stallTime} \end{aligned}$$

$$\text{stall cycles} = \frac{\text{MemoryAccesses}}{\text{Program}} \cdot \text{MissRate} \cdot \text{MissPenalty}$$

$$\text{(or)} \quad = \frac{\text{Instructions}}{\text{Program}} \cdot \frac{\text{Misses}}{\text{Instruction}} \cdot \text{MissPenalty}$$

- Two typical ways of improving performance:
 - decreasing the miss rate
 - decreasing the miss penalty

What happens if we increase block size?

Add associativity?

3

ADMIN

- Reading – finish Chapter 7
 - Sections 7.4 (skip 531-536), 7.5, 7.7, 7.8

2

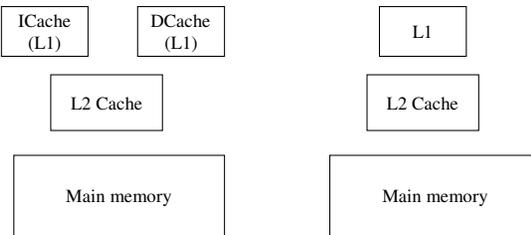
Performance Example

- Suppose processor has a CPI of 1.5 given a perfect cache. If there are 1.2 memory accesses per instruction, a miss penalty of 20 cycles, and a miss rate of 10%, what is the effective CPI with the real cache?

4

Split Caches

- Instructions and data have different properties
 - May benefit from different cache organizations (block size, assoc...)

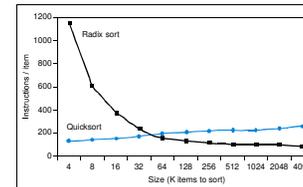


- Why else might we want to do this?

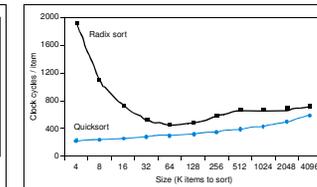
5

Cache Complexities

- Not always easy to understand implications of caches:



Theoretical behavior of Radix sort vs. Quicksort

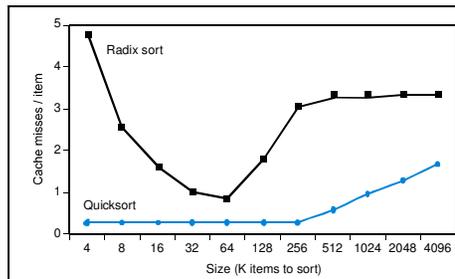


Observed behavior of Radix sort vs. Quicksort

6

Cache Complexities

- Here is why:



- Memory system performance is often critical factor
 - multilevel caches, pipelined processors, make it harder to predict outcomes
 - Compiler optimizations to increase locality sometimes hurt ILP
- Difficult to predict best algorithm: need experimental data

7

Program Design for Caches – Example 1

- Option #1


```
for (j = 0; j < 20; j++)
    for (i = 0; i < 200; i++)
        x[i][j] = x[i][j] + 1;
```
- Option #2


```
for (i = 0; i < 200; i++)
    for (j = 0; j < 20; j++)
        x[i][j] = x[i][j] + 1;
```

8

Program Design for Caches – Example 2

- Why might this code be problematic?

```
int A[1024][1024];
int B[1024][1024];
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
        A[i][j] += B[i][j];
```

- How to fix it?

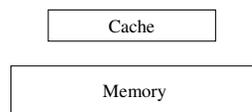
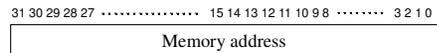
9

VIRTUAL MEMORY

10

Virtual memory summary (part 1)

Data access without
virtual memory:

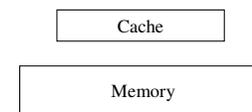
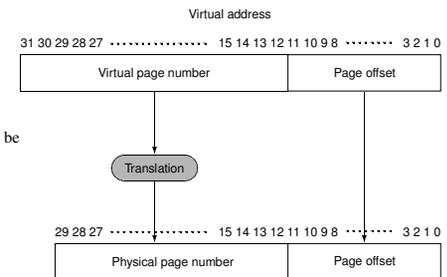


11

Virtual memory summary (part 2)

Data access with
virtual memory:

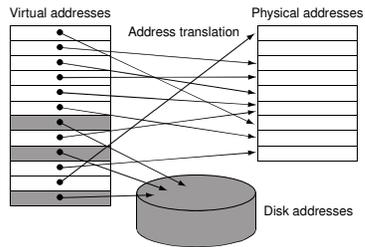
"all problems in Computer Science can be
solved by another level of indirection"
-- Butler Lampson



12

Virtual Memory

- Main memory can act as a cache for the secondary storage (disk)



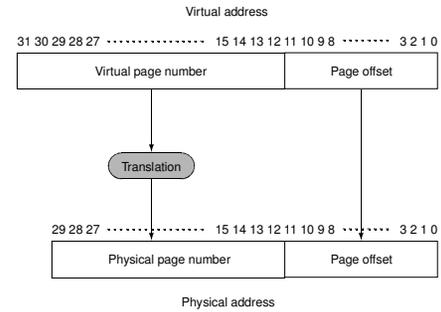
- Advantages:
 - Illusion of having more physical memory
 - Program relocation
 - Protection
- Note that main point is caching of disk in main memory but will affect all our memory references!

13

Address Translation

Terminology:

- Cache block →
- Cache miss →
- Cache tag →
- Byte offset →



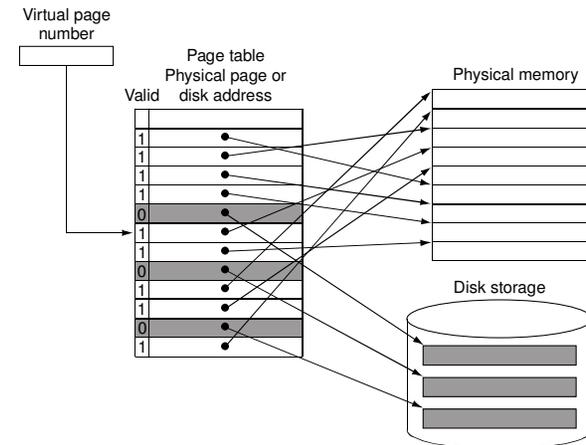
14

Pages: virtual memory blocks

- Page faults: the data is not in memory, retrieve it from disk
 - huge miss penalty (slow disk), thus
 - pages should be fairly
 - Replacement strategy:
 - can handle the faults in software instead of hardware
- Writeback or write-through?

15

Page Tables



16

Example – Address Translation Part 1

- Our virtual memory system has:
 - 32 bit virtual addresses
 - 28 bit physical addresses
 - 4096 byte page sizes
- How to split a virtual address?

Virtual page #	Page offset
----------------	-------------

- What will the physical address look like?

Physical page #	Page offset
-----------------	-------------

- How many entries in the page table?

17

Example – Address Translation Part 2

EX 7-31...

Translate the following addresses:

1. C0001560
2. C0006123
3. C0002450

Page Table

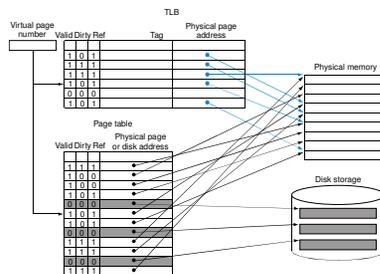
	Valid?	Physical Page or Disk Block #
C0000	1	A204
C0001	1	A200
C0002	0	FB00
C0003	1	8003
C0004	1	7290
C0005	0	5600
C0006	1	F5C0

...

18

Making Address Translation Fast

- A cache for address translations: translation lookaside buffer



Typical values: 16-512 entries,
miss-rate: .01% - 1%
miss-penalty: 10 - 100 cycles

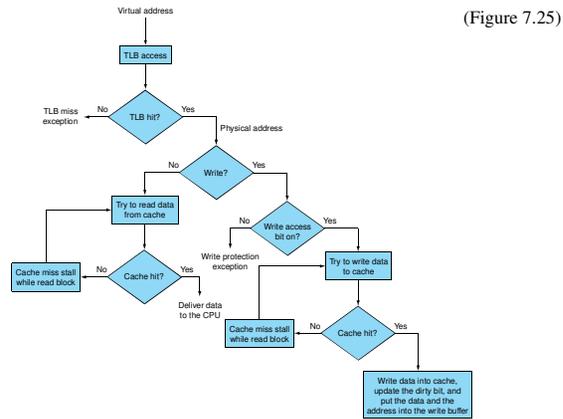
19

Protection and Address Spaces

- Every program has its own “address space”
 - Program A’s address 0xc000 0200 not same as program B’s
 - OS maps every virtual address to distinct physical addresses
- How do we make this work?
 - Page tables –
 - TLB –
- Can program A access data from program B? Yes, if...
 1. OS can map different virtual page #'s to same physical page #'s
 - So A’s 0xc000 0200 = B’s 0xb320 0200
 2. Program A has read or write access to the page
 3. OS uses supervisor/kernel protection to prevent user programs from modifying page table/TLB

20

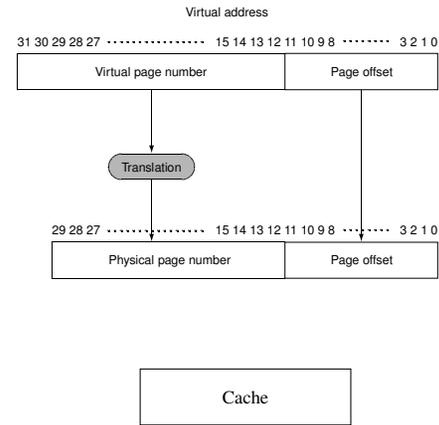
Integrating Virtual Memory, TLBs, and Caches



21

TLBs and Caches

What happens after translation?



22

Modern Systems

- Things are getting complicated!

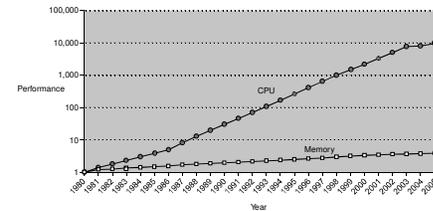
ICPU	AMD Opteron	Intel Pentium 4	Intel Pentium 4	Intel Pentium 4	Sun UltraSPARC IV
Instruction set architecture	x86_64, AMD64	x86_64	x86_64	x86_64	SPARC v9
Intended application	server	embedded	desktop	low-power embedded	server
Die size (mm ²) (2004)	193	122	217	—	366
Instructions issued/clock	3	2	3 RISC ops	1	4 × 2
Clock rate (2004)	2.0 GHz	2.0 GHz	3.2 GHz	0.4 GHz	1.2 GHz
Instruction cache	64 KB, 2-way set associative	16 KB, direct mapped	12000 RISC op trace cache (~96 KB)	32 KB, 32-way set associative	32 KB, 4-way set associative
Latency (clocks)	37	4	4	1	2
Data cache	64 KB, 2-way set associative	16 KB, 1-way set associative	8 KB, set associative	32 KB, 32-way set associative	64 KB, 4-way set associative
Latency (clocks)	3	3	2	1	2
TLB entries (I/D/L2 TLB)	40/40/512/ 512	16	128/128	32/32	128/512
Minimum page size	4 KB	4 KB	4 KB	1 KB	8 KB
On-chip L2 cache	1024 KB, 12-way set associative	1024 KB, 4-way set associative	512 KB, 8-way set associative	—	—
Off-chip L2 cache	—	—	—	—	16 MB, 2-way set associative
Block size (L1/L2, bytes)	64	64	64/128	32	32

FIGURE 7.16 Desktop, embedded, and server microprocessors in 2004. From a memory hierarchy perspective, the primary difference between categories is that L2 cache. There is no L2 cache for the low-power embedded, a large on-chip L2 for the embedded and desktop, and 16 MB off-chip for the server. The processor clock rates also vary: 0.4 GHz for low-power embedded, 1 GHz or higher for the rest. Note that UltraSPARC IV has two processors on the chip.

23

Some Issues

- Processor speeds continue to increase very fast
— much faster than either DRAM or disk access times



- Design challenge: dealing with this growing disparity
— Prefetching? 3rd level caches and more? Memory design?

24