
SI232 Set #18: Caching Finale and Virtual Reality (Chapter 7)

1

Down the home stretch...

	Monday	Wednesday	Friday
2-Apr	Review	Exam	Memory
9-Apr	Memory	Ethics Discussion. Reading Quiz.	Virtual Memory. I/O
16-Apr	I/O.	Pipelining. HW (Ch 7) Due.	Pipelining, hazards.
23-Apr	ILP and multiple issue. Course paper due.	Improving multiple issue	Last class. Advanced topics/review.

Final Exam – Monday May 1 (first exam day)

3

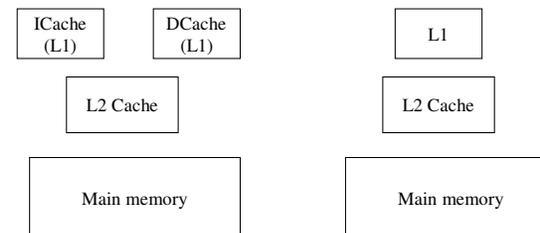
ADMIN

- **Ethics Discussion & Reading Quiz – Wed April 12**
 - Reading posted online
- **Reading – finish Chapter 7**
 - Sections 7.4 (skip 531-536), 7.5, 7.7, 7.8

2

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?**

4

Cache Performance

- Simplified model:

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

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

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

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

What happens if we increase block size?

Add associativity?

5

Performance Example #2 – Split Cache

- Suppose processor has a CPI of 1.0 given a perfect cache. If the instruction cache miss rate is 3% and the data cache miss rate is 10%, what is the effective CPI with the real cache?
Assume a miss penalty of 10 cycles and that 40% of instructions access data.

7

Performance Example #1 – Unified Cache

- 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?

6

Exercise #1

- Suppose processor has a CPI of 2.0 given a perfect cache. If there are 1.5 memory accesses per instruction, a miss penalty of 40 cycles, and a miss rate of 5%, what is the effective CPI with the real cache?

8

Exercise #2

- You are given a processor with a 64KB, direct-mapped instruction cache and a 64 KB, 4-way associative data cache. For a certain program, the instruction cache miss rate is 4% and the data cache miss rate is 5%. The miss penalty is 10 cycles for the I-cache and 20 cycles for the D-cache. If the CPI is 1.5 with a perfect cache, and 30% of instructions access data, what is the effective CPI?

9

Exercise #3

- Suppose a processor has a base CPI of 1.0 (no cache misses) but currently an effective CPI of 2.0 once misses are considered. There are 1.5 memory accesses per instruction. If the processor has a unified cache with a miss rate of 2%, how low must the miss penalty be in order to improve the effective CPI to 1.3?

10

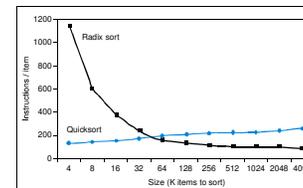
Exercise #4 – Stretch

- A certain processor has a CPI of 1.0 with a perfect cache and a CPI of 1.2 when memory stalls (due to misses) are included. We wish to speed up the performance of this processor by 2x, which we will do by increasing the clock rate. This, however, will not improve the memory system, so misses will take just as long in absolute terms. How much faster must the clock rate be in to meet our goal?

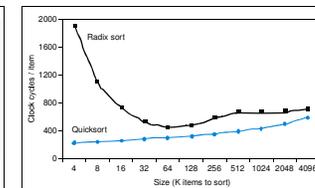
11

Cache Complexities

- Not always easy to understand implications of caches:



Theoretical behavior of Radix sort vs. Quicksort

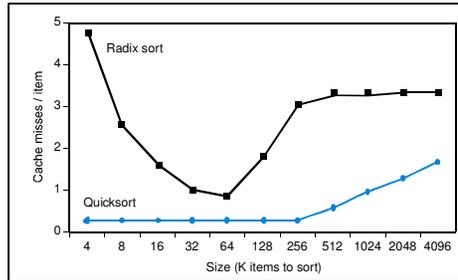


Observed behavior of Radix sort vs. Quicksort

12

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

13

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;
```

14

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?

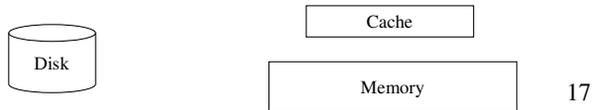
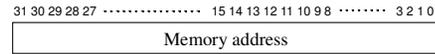
VIRTUAL MEMORY

15

16

Virtual memory summary (part 1)

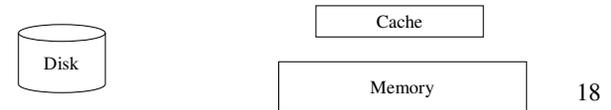
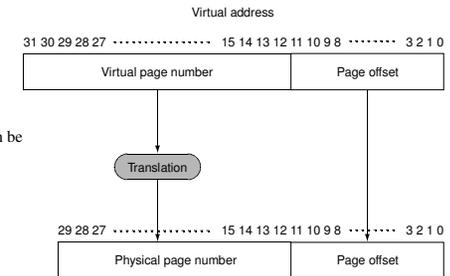
Data access without virtual memory:



Virtual memory summary (part 2)

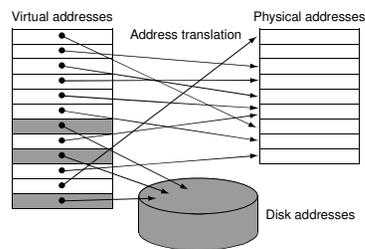
Data access with virtual memory:

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



Virtual Memory

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



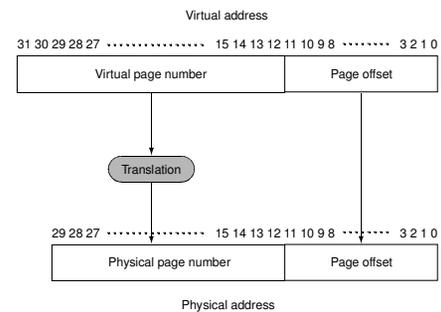
- Advantages:
 - Illusion of having more physical memory
 - Program relocation
 - Protection
- Note that main point is caching of disk in main memory but will affect all our memory references!

19

Address Translation

Terminology:

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



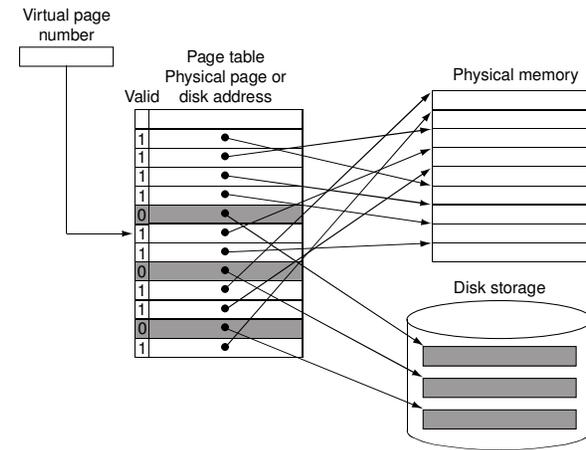
20

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?

21

Page Tables



22

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?

23

Example – Address Translation Part 2

Translate the following addresses:

1. C0001560
2. C0006123
3. C0002450

	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

...

24

Exercise #1

- Given system with
 - 20 bit virtual addresses
 - 16 bit physical addresses
 - 256 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?

25

Exercise #2 (new problem – not related to #1)

Translate the following addresses:

- B004890
- B002123
- B006001

Page Table

	Valid?	Physical Page or Disk Block #
B000	1	B004
B001	1	A120
B002	0	AB00
B003	0	8003
B004	1	7590
B005	1	5800
B006	1	F4C0

...

26

Exercise #3

Given the fragment of a page table on the right, answer the following questions assuming a page size of 1024 bytes

- What is the virtual address size (# bits)

- What is the physical address size (# bits)

- Number of entries in page table?

Page Table

	Valid?	Physical Page #
B000	1	B0
B001	1	A0
B002	0	AB
B003	0	80
B004	1	90
B005	1	58
B006	1	F4

...

27

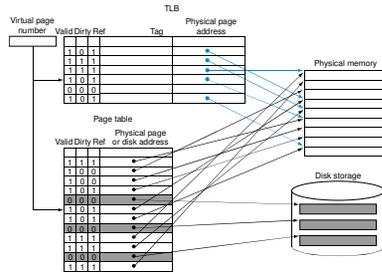
Exercise #4

- Is it possible to have the physical address be wider (more bits) than the virtual address?
If so, when would this ever make sense?

28

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

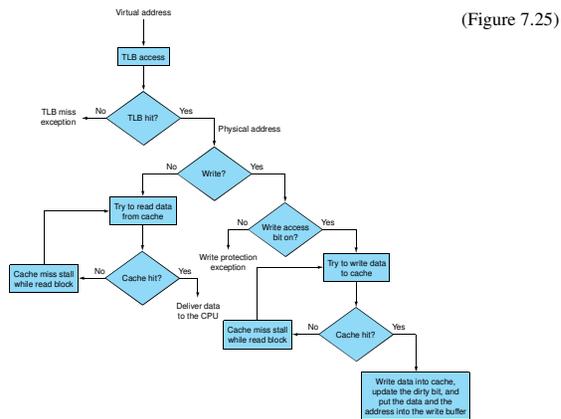
29

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

30

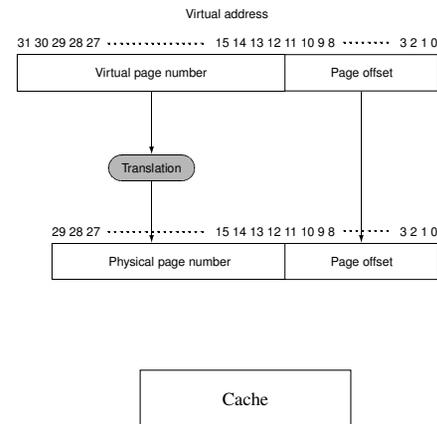
Integrating Virtual Memory, TLBs, and Caches



31

TLBs and Caches

What happens after translation?



32

Modern Systems

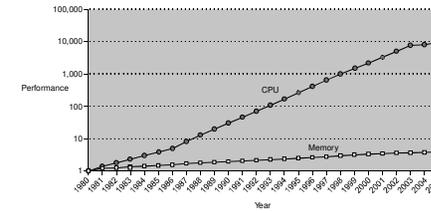
- Things are getting complicated!

NPU	AMD Opteron	Infiniety FastMATH	Intel Pentium 4	Intel PXA250	Sun UltraSPARC IV
Instruction set architecture	IA-32, AMD64	MIPS32	IA-32	ARM	SPARC v9
Intended application	server	embedded	desktop	low-power embedded	server
Die size (mm ²) (2004)	193	1.22	217	—	356
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, 4-way 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, 16-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 7.36 Desktop, embedded, and server microprocessors in 2004. From a memory hierarchy perspective, the primary difference between categories is the 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.

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?