

Lecture 15 Memory Hierarchy (3)

Instructor: Jun Yang

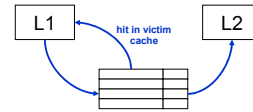
11/25/2003

Lec. 15

1

Reducing Cache Miss Penalty (5)

- Victim Cache – “recycling”
 - Holds victim blocks discarded from the L1 cache due to replacement.
 - Small (otherwise an L2) and fully associative.
 - Checked on a L1 miss. If found, block is swapped back to L1 (the block previously took its place is put into victim cache).
 - Works better for small L1 caches since it saves victim blocks from conflict misses.
 - Effective: a 4-entry vc can reduce $\frac{1}{4}$ of the misses in a 4KB L1 cache [Jouppi, 1990].



11/25/2003

Lec. 15

2

Reducing Miss Rate (1–3)

1. Larger block size
 - Takes advantage of locality
 - But, longer miss penalty and maybe more conflict misses (w/ same cache size)
 - Block size increase should not reach the point where miss rate increases.
2. Larger cache size
 - Longer hit time – suitable for lower level caches.
3. Higher associativity
 - 2:1 cache rule of thumb: a direct-mapped cache of size N has about the same miss rate as a 2-way set-associative cache of size N/2 (for cache size < 128KB)
 - Longer hit time

Improving an aspect of AMAT comes at the expense of another!

11/25/2003

Lec. 15

3

Reducing Miss Rate (4,5)

4. Pseudoassociative cache – a direct-mapped cache having same hit rate as a 2-way set-associative cache
 - If the first access is a miss, try an alternative block (by modifying an address portion)
 - A normal hit time and a pseudohit time – in addition to the miss penalty

11/25/2003

Lec. 15

4

Parallelize mem. Access with Execution

1. Nonblocking cache (lockup-free)
 - Continue to service cache accesses during a miss – works for ooo processors.
 - Significantly increase the complexity of the cache controller.
2. Hardware prefetching
 - Get the data or instruction before it is accessed
 - Does not slow down other cache activities
 - ❖ Continue to serve other instructions or data while waiting for the prefetched data – normally nonblocking.
 - Usually do not replace useful data (use additional buffers).
 - Stream buffer for instructions
 - ❖ On l-cache miss, check stream buffer.
 - ❖ Stream buffer hit → move the block into l-cache, refill (prefetch) stream buffer with **next** block.
 - ❖ Stream buffer miss → fetch target block into l-cache and **next** block into stream buffer

11/25/2003

Lec. 15

5

Parallelize mem. Access with Execution

3. Compiler-controlled prefetching
 - Compiler inserted *prefetch* instructions; two types:
 - ❖ *register prefetch*: data loaded to registers
 - ❖ *cache prefetch*: data to CM.
 - Either type can be
 - ❖ *faulting* or
 - ❖ *non-faulting*, i.e. allowed to cause a page fault or not.
 - Prefetching is effective with loop unrolling or software pipelining.
 - But,
 - ❖ instruction overhead (such overhead can not exceed the benefits)
 - ❖ Maybe the data or instruction is already in the cache
 - ❖ Is the prefetch early enough for the data to arrive by the time it is needed

11/25/2003

Lec. 15

6

Reducing Hit Time

- Use small and simple L1 cache
 - Small hardware is faster
 - Direct map cache is faster
- Pipeline writes
 - In writes: must check tag BEFORE write is done.
 - Separate tag check and data write, delay data write with respect to tag check: back to back writes (Alpha 21064).
- Trace cache, another type of I-cache (Pentium 4)
 - Stores dynamic instruction sequence (trace) instead of static sequence.
 - Include multiple taken branches and make them into straight line code
 - Reduce I-cache misses due to fetch branch target since the target now is just the next instruction
 - Downside:
 - Instruction may repeatedly occur in trace cache – wasting space

11/25/2003

Lec. 15

7

Reducing Hit Time

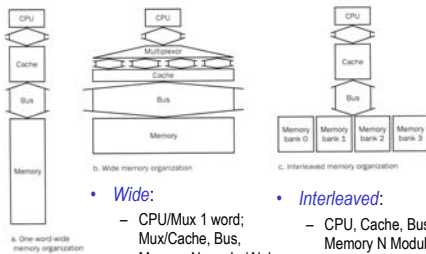
- Avoid address translation
 - Addresses sent to cache need to be translated from virtual addresses to physical addresses – done by **translation lookaside buffer** (TLB)
 - Translation occur before going to the L1 cache – cost time
 - Virtually addressed cache:
 - Index the cache using virtual addresses, avoid TLB accesses, save time
 - Context switch causes flushing the entire cache
 - Alias: different virtual addresses may map into same physical address – need protection mechanism (pp. 445!)
- Way prediction – approaching hit time of a direct-mapped cache for set associative caches
 - Do not compare all the tags.
 - Predict one and access the way just as a direct-mapped cache.
 - Prediction is done in the previous cache access (extra prediction bits are maintained).
 - On a misprediction, all the rest ways are compared as a normal set-associative cache – takes longer time.
 - Alpha 21264 instruction cache

11/25/2003

Lec. 15

8

Main Memory Organization



Simple:

- CPU, Cache, Bus, Memory same width (32 bits)

Wide:

- CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)

Interleaved:

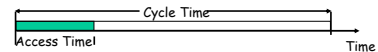
- CPU, Cache, Bus 1 word; Memory N Modules (4 Modules); example is *word interleaved*

11/25/2003

Lec. 15

9

Main Memory Performance



- DRAM (Read/Write) Cycle Time >> DRAM (Read/Write) Access Time
- DRAM (Read/Write) Cycle Time :
 - How frequent can you initiate an access? $1/\text{Cycle Time}$
- DRAM (Read/Write) Access Time:
 - How quickly will you get what you want once you initiate an access?

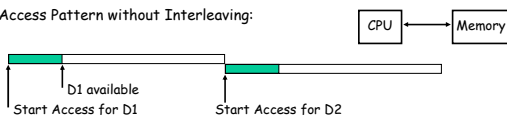
11/25/2003

Lec. 15

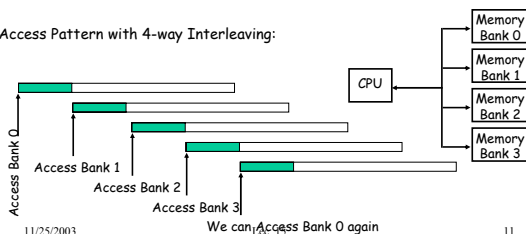
10

Increasing Bandwidth – Interleaving

Access Pattern without Interleaving:



Access Pattern with 4-way Interleaving:



11/25/2003

Lec. 15

11

Main Memory Performance

- Timing model
 - 1 to send address,
 - 4 for access time, 10 cycle time, 1 to return data
 - Cache Block is 4 words
- Simple M.P. = $4 \times (1+10+1) = 48$
- Wide M.P. = $1 + 10 + 1 = 12$
- Interleaved M.P. = $1+4+1+3=9$

address	address	address	address
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
Bank 0	Bank 1	Bank 2	Bank 3

11/25/2003

Lec. 15

12

And that's about it!

- You learned a lot – congratulations!
- But there are more ...
 - Multiprocessor, bus structure, disks, interconnection networks and clusters
 - Some are offered in 203B
 - Some will be covered in 260
 - ✦ Thread-level parallelism (TLP instead of ILP)
 - ✦ Embedded system architecture
 - ✦ Network processors
 - ✦ Low power designs
 - ✦ And more...
- The book is very informative!