CS4100: Computer System Design

Memory Hierarchy Design

Madhu Mutyam
PACE Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Aug 31, 2015

Basic Cache Optimizations

- Average Memory Access Time (AMAT)

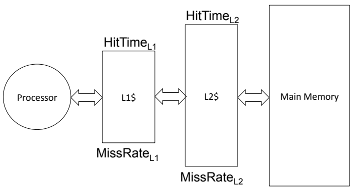
$$\text{AMAT} = \text{Hit\_Time} + \text{Miss\_Rate} \times \text{Miss\_Penalty}$$

- Reducing the miss rate
  - larger block size, larger cache size, and higher associativity
- Reducing the miss penalty
  - multilevel caches, giving reads priority over writes
- Reducing the hit time
  - avoiding address translation when indexing the cache

Effects on Cache Miss Rate

Cache Parameter	Cold Misses	Capacity Misses	Conflict Misses	Overall Misses
Reduced capacity	No effect	Increase	May increase	May increase
Increased capacity	No effect	Decrease	May decrease	May decrease
Reduced block size	Increase	May decrease	May decrease	Varies
Increased block size	Decrease	May increase	May increase	Varies
Reduced associativity	No effect	No effect	May increase	May increase
Increased associativity	No effect	No effect	May decrease	May decrease

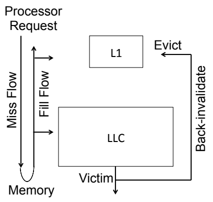
Reducing the Miss Penalty: Multi-Level Caches



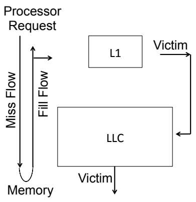
$$\begin{aligned}\text{Miss\_Penalty}_{L1} &= \text{Hit\_Time}_{L2} + \text{Miss\_Rate}_{L2} \times \text{Miss\_Penalty}_{L2} \\ \text{AMAT}_{2Level} &= \text{Hit\_Time}_{L1} + \text{Miss\_Rate}_{L1} \times (\text{Hit\_Time}_{L2} \\ &\quad + \text{Miss\_Rate}_{L2} \times \text{Miss\_Penalty}_{L2})\end{aligned}$$

- Local miss rate is w.r.t. the number of memory accesses to the cache
  - Miss_Rate_{L1} and Miss_Rate_{L2}
- Global miss rate is w.r.t. the number of memory accesses generated by the processor
  - Miss_Rate_{L1} and $(\text{Miss\_Rate}_{L1} \times \text{Miss\_Rate}_{L2})$

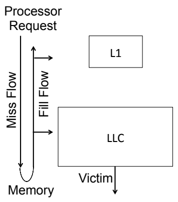
Different Types of Cache Hierarchies



Inclusive Cache

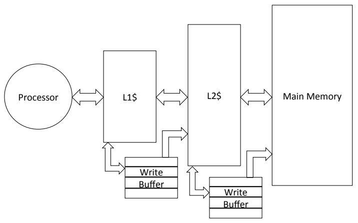


Exclusive Cache



Non-Inclusive Cache

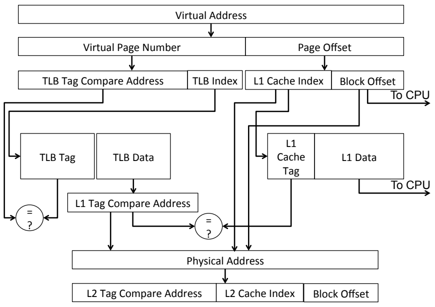
Reducing the Miss Penalty: Prioritize Reads Over Writes



- *Write buffers* improve performance in both *write-through* and *write-back* caches
- Write buffers can create *Read-After-Write* (RAW) hazards through memory
  - Check the contents of the write buffer on a read miss
  - If there are no conflicts, and if the memory system is available, send the read before write

Reducing the Hit Time: Avoid Address Translation during the Cache Indexing

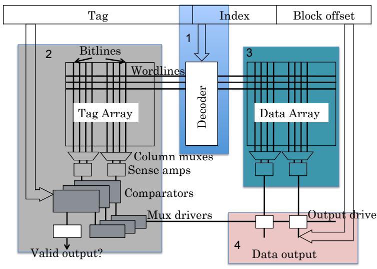
- Physically indexed and physically tagged caches
- Virtually indexed and physically tagged caches

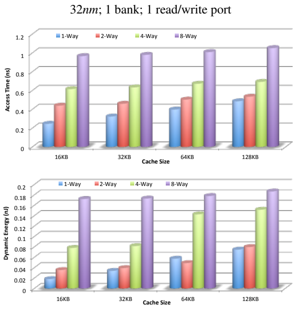


Advanced Optimizations

- Reducing the hit time
  - Small and simple first-level caches, and way-prediction
- Increasing the cache bandwidth
  - Pipelined caches, non-blocking caches, and multi-banked caches
- Reducing the miss penalty
  - Critical word first, early restart, and merging write buffers
- Reducing the miss rate
  - Compiler optimizations
- Reducing the miss penalty or miss rate via parallelism
  - Hardware and compiler prefetching

Small and Simple First-Level Caches

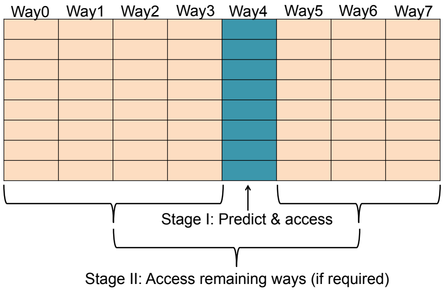




Cache access = set decoding + tag comparison + data read + data out

- CACTI tool from HP (http://www.hpl.hp.com/research/cacti/)

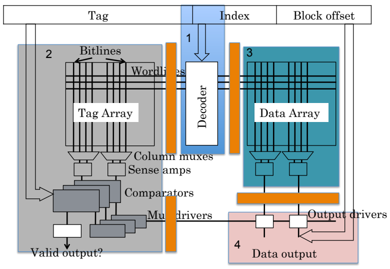
Way Prediction



Way0
Way1
Way2
Way3
Way4
Way5
Way6
Way7
Stage I: Predict & access
Stage II: Access remaining ways (if required)

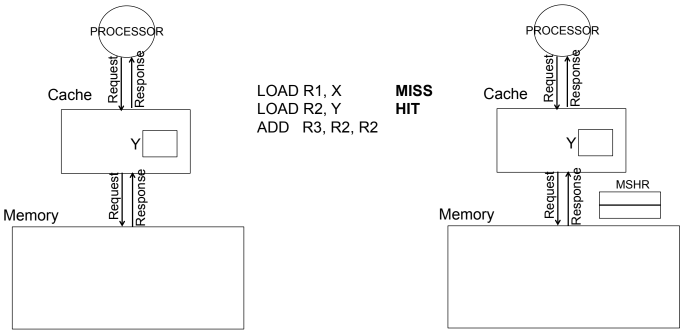
- Cache hit time can be reduced
  - Misprediction increases the hit time
- Way prediction can also reduce the energy significantly
- Instruction caches can have better accuracy than data caches

Pipelined Caches



- Pipeline cache access to improve bandwidth
  - Pentium: 1 cycle; Pentium Pro – Pentium III: 2 cycles; Pentium IV – Core i7: 4 cycles
- Makes it easier to increase associativity
- Increases branch misprediction penalty

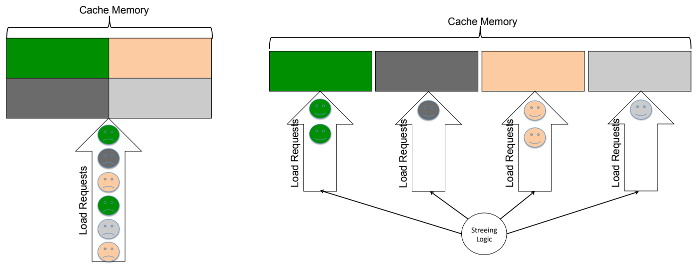
Non-Blocking Caches



```
LOAD R1, X       MISS
LOAD R2, Y       HIT
ADD   R3, R2, R2
```

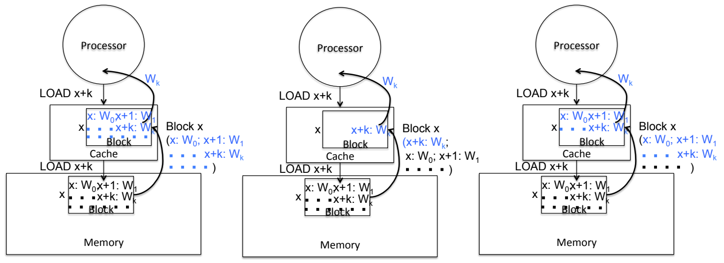
- Allow hits before previous misses
  - *Hit under miss* or *Hit under multiple misses*

Multi-Banked Caches



- Organize the cache as independent banks to support simultaneous access
  - Intel Core i7 supports 4 banks for L1 and 8 banks for L2
- Multiple banks are also a way to reduce power consumption

Critical Word First and Early Restart



Servicing a Load Request

Critical Word First

Early Restart

- Benefits depend on the cache block size and the likelihood of another access to the portion of the block that has not yet been fetched

Merging Write Buffers

Write Address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

Write Address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

- When storing to a block that is already pending in the write buffer, update the write buffer
- Do not apply to I/O addresses

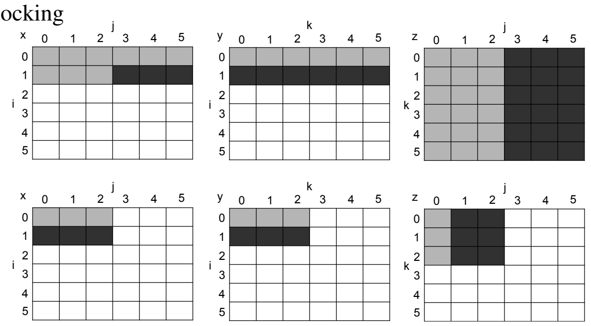
Compiler Optimizations

- Loop interchange

```
for(j=0; j<100; j++)
 for(i=0; i<5000; i++)
  x[i][j] = 2*x[i][j]
```

```
for(i=0; i<5000; i++)
 for(j=0; j<100; j++)
  x[i][j] = 2*x[i][j]
```

- Blocking



Prefetching

- Prefetch data before the processor requests them
- Hardware prefetching
  - *Stream-based* and *stride-based*
  - Ex: Intel Core i7 supports simple stream-based hardware prefetching into both L1 and L2
  - Aggressive hardware prefetching may sometimes degrade the performance
- Software prefetching
  - Insert prefetch instructions before data is needed
  - *Register prefetching* and *cache prefetching*

Thank You