CS4100: Computer System Design

Memory Hierarchy Design

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

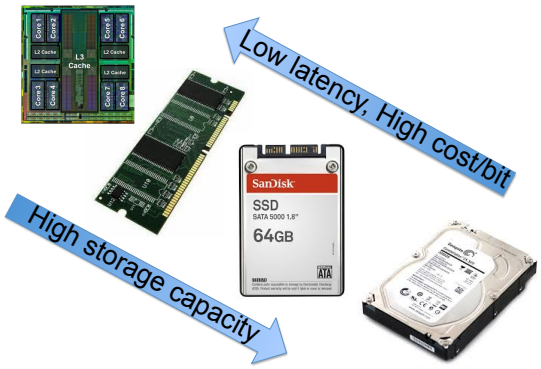
Aug 20-24, 2015

Principle of Locality

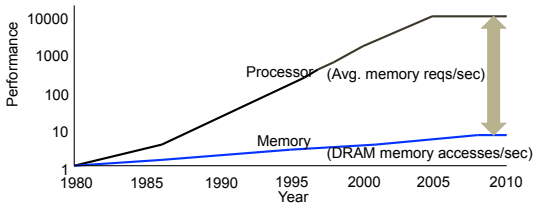
```
for(i=0; i<l; i++)
   for(j=0; j<m; j++)
      for(k=0; k<n; k++)
         A[i][j]+ = B[i][k] × C[k][j];
```

- Temporal locality – Recently accessed data items are likely to be accessed in the near future
- Spatial locality – Nearby data items of an accessed data item are likely to be accessed
- Organize memory system into a hierarchy with faster (but smaller) memory closer to processor

Memory Hierarchy



The Memory Wall



- Aggregate peak bandwidth requirement grows with # of cores

Quantifying Cache Performance

- When a perfect cache is considered:

$$\text{CPU time} = \text{CPU clock cycles} \times \text{Clock cycle time}$$

- If processor is stalled for a memory access

$$\text{CPU time} = (\text{CPU clock cycles} + \text{Memory stall cycles}) \times \text{Clock cycle time}$$

$$\text{Memory stall cycles (MemStall)} = \text{Number of misses} \times \text{Miss penalty}$$

$$\begin{aligned}\text{MemStall} &= \text{IC} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \frac{\text{Misses}}{\text{Memory accesses}} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}\end{aligned}$$

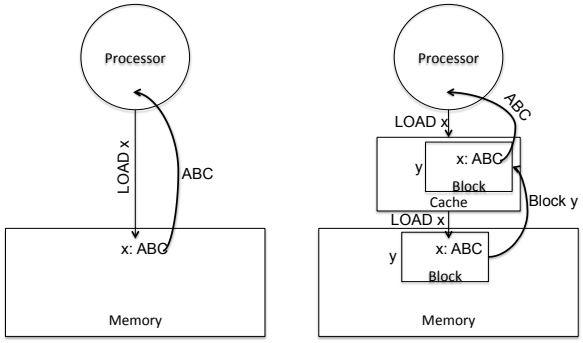
Example #1

- Assume that the CPI of a computer is 1 when all memory accesses hit in the cache. If 30% of the instructions are loads and stores, miss penalty is 100 cycles and miss rate is 5%, how much faster the computer be if all instructions were cache hits?

If all memory accesses are hits

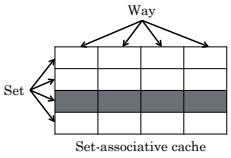
$$\begin{aligned}\text{CPU time} &= ((\text{CPU clock cycles} + 0) \times \text{Clock cycle time} \\ &= (\text{IC} \times \text{CPI} + 0) \times \text{Clock cycle time} \\ &= \text{IC} \times 1.0 \times \text{Clock cycle time} \\ \text{Memory stall cycles} &= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty} \\ &= \text{IC} \times (1 + 0.3) \times 0.05 \times 100 = \text{IC} \times 6.5 \\ \text{CPU time}_{MemStall} &= (\text{IC} \times 1.0 + \text{IC} \times 6.5) \times \text{Clock cycle time} \\ &= 7.5 \times \text{IC} \times \text{Clock cycle time} \\ \text{Speedup} &= 7.5\end{aligned}$$

Cache Memory

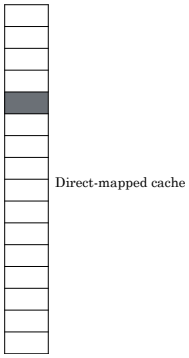


- Cache memory is designed using SRAM-based technology
- Cache block is typically 32B or 64B

Cache Organization

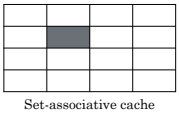


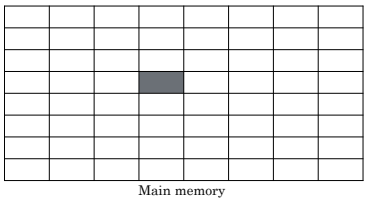
Mapping: (Block address) MOD (# of sets in cache)



Mapping: (Block address) MOD (# of blocks in cache)

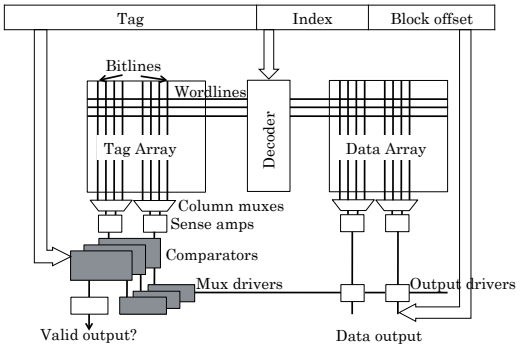
Finding a Block in a Cache





Block address		
Tag	Index	Block offset

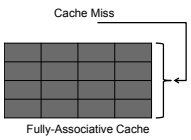
Cache Access

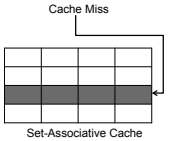


- *Parallel mode of access* – access both tag and data arrays in parallel
- *Serial mode of access* – access the tag array first followed by the data array

Cache Miss Classification

- *Compulsory* misses – the very first access to a block, independent of the cache size
  - Increase the block size



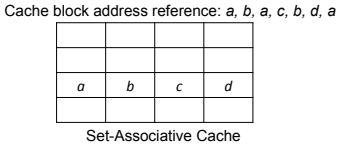


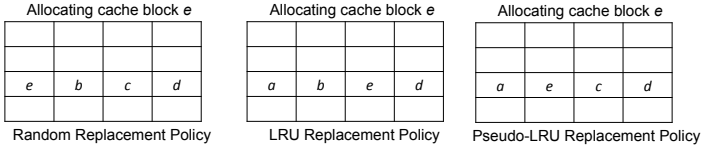
- *Capacity* misses
  - Increase the cache size
- *Conflict* misses
  - Increase the associativity

- Direct mapped cache – search is simple, but high conflict miss rate
- Fully-associative cache – low conflict miss rate, but search time is more
- Set-associative cache – reduces conflict miss rate

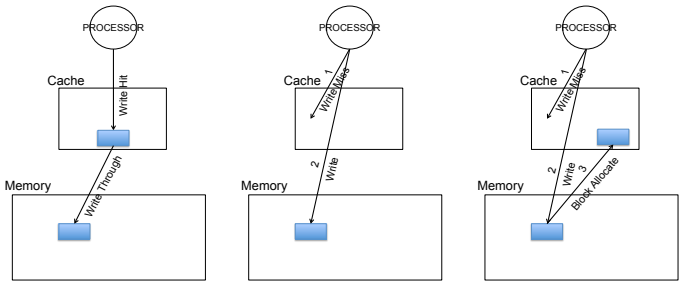
Block Replacement on a Cache Miss

- Direct-mapped caches – Only one block position
- Set-associative caches – Random, LRU, Pseudo-LRU





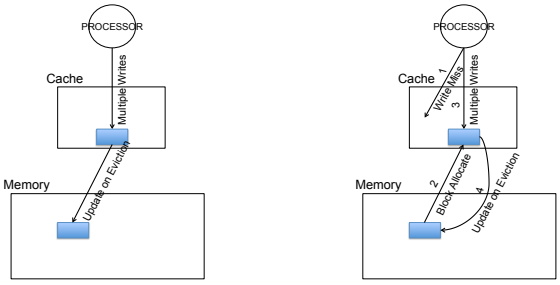
Cache Block Updates: Write-Through Caches



On a Write Hit | On a Write Miss – Write No Allocate | On a Write Miss – Write Allocate

- Requires more bandwidth at lowest levels of caches
- Keeps the cache consistent with lower levels of the memory

Cache Block Updates: Write-Back Caches



On a Write Hit | On a Write Miss

- Requires less write bandwidth
- Attractive for multiprocessor systems
- Power efficient

Thank You