CS147 - Lecture 22

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Cache Performance
- Improving Cache Performance

Reference Books / Source:

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*

Cache Performance ...

2

System Performance Measurement

- **CPU Time :** $T_{CPU} = (C_{CPU\ execute} + C_{Memory\ stall}) \times T_C$
- **Memory Stall Cycle :** $C_{Memory\ stall} = C_{Read\ stall} + C_{Write\ stall}$
- **Read stall cycle :** $C_{Read\ stall} = \frac{I_{read}}{I_{total}} \times I_{total} \times MR_{Read} \times P_{Read\ Miss}$
- **Write stall cycle :**

$$C_{Write\ stall} = \left(\frac{I_{write}}{I_{total}} \times I_{total} \times MR_{Write} \times P_{Write\ Miss}\right) + C_{Buffer\ Stall}$$

3

• In the above formulae the parameters are as following.
  - T – time
  - C – Cycle
  - I – Number of instruction
  - MR – Miss rate
  - P – Penalty in terms of number of cycles

• Total CPU time or execution time is the total number of cycles multiplied by time per cycle. Total cycle includes cycles required to execute the instructions and cycles incur due to memory stall. Memory stall cycle contains two parts – read stall and write stall. Number of read stall can be computed using fraction of read operation in entire program, read miss rate and the penalty incurred from miss. Similarly write stall can be computed using fraction of write instructions, write miss rate, and penalty for write miss. Additionally for write-through cache has number of cycle needed for buffer stall.

System Performance Measurement

- Usually with a write buffer of depth 4 or more, write buffer stall comes to a very small number.
  - If it is still high, need to
    - Increase the depth of the buffer
    - Move to write back strategy
  - In this case we can safely ignore the write buffer stall.

$$C_{Write\,stall} = \left(\frac{I_{write}}{I_{total}} \times I_{total} \times MR_{Write} \times P_{Write\,Miss}\right)$$

4

- With a reasonable write buffer depth (4 or more) write buffer stall usually becomes insignificant. Even if using a write buffer strategy we observe a stall due to write buffer, we need to increase the buffer depth or move to write back strategy. In most of the practical cases we can drop the buffer stall cycle form the equation.

System Performance Measurement

- Usually the Read and Write Penalty are same.
  - For write through cache both read and write miss will need to replace the cache line.
  - For write back cache both read and write miss will need to write back dirty cache line and replace the cache line.

$$C_{Memory\ stall} = \left(\frac{I_{mem\ access}}{I_{total}} \times I_{total} \times MR_{mem\ access} \times P_{mem\ access}\right)$$

5

- If we review the read and write handling flow chart in cache operation, we can observe that the read and write penalty are same (both in the case of write-through or write-back). Therefore, we can unify the read and write stall cycle computation and create cycle equation for memory access stall. The equation involves fraction of memory access instruction (read and write), miss rate (average of read and write) and penalty upon miss.

System Performance Measurement

- Assume instruction cache miss rate is 2% and data cache miss rate is 4%. If a processor has a CPI of 2 without any memory stalls and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed assuming there are 36% instructions is a memory instructions.

$$Total\ Number\ of\ \ Instruction = I$$

$$C_{Instruction\ Miss} = I \times 0.02 \times 100 = 2.00 \times I$$

$$C_{Data\ miss} = I \times 0.36 \times 0.04 \times 100 = 1.44 \times I$$

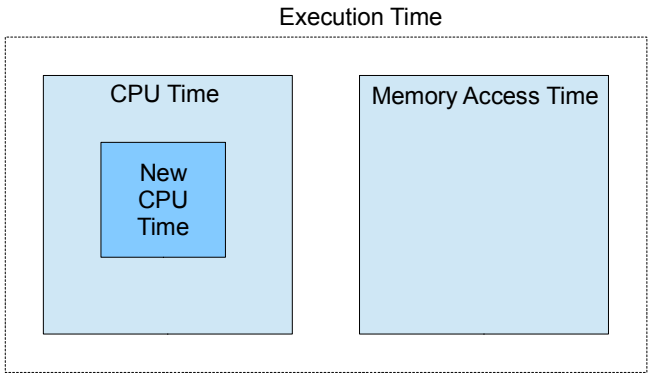
$$\Rightarrow C_{Memory\ stall} = C_{Instruction\ Miss} + C_{Data\ miss} = (2.00 + 1.44) \times I = 3.44\ I$$

$$Therefore, C_{Avg\ Stall}\ per\ instruction\ is\ 3.44; Hence\ average\ CPI\ is\ (2 + 3.44) = 5.44$$

$$\Rightarrow \frac{T_{CPU\ time\ with\ stall}}{T_{CPU\ time\ with\ perfect\ cache}} = \frac{I \times CPI_{stall} \times T_{clock}}{I \times CPI_{perfect} \times T_{clock}} = \frac{CPI_{stall}}{CPI_{perfect}} = \frac{5.44}{2} = 2.72$$

System Performance Measurement

- If the computing system is made faster without improving memory system speed the slow down becomes more.



7

- What is the effect of speeding up the processor without speeding up the memory system?
  - Recall the Amdahl's law – following that rule we can say that memory access will take relatively more time than what CPU is taking. We'll not be getting performance improvement benefit in full if we do not improve memory system's performance.

System Performance Measurement

- In the previous example if the CPI is improved to be 1, what would be the performance difference between cache with stall and without stall?

$$CPI_{perfect} = 1$$

$$The\ new\ CPI_{stall} = 1 + 3.44 - 4.44$$

$$\frac{T_{with\ stall}}{T_{without\ stall}} = \frac{4.44}{1.0} = 4.4$$

$$The\ previous\ difference\ was\ 2.72$$

System Performance Measurement

- In the previous example if the clock frequency for CPU is increased double what would be the performance comparison between system with cache miss and perfect cache?

Since the memory speed remains same , the miss penalty will be doubled. i.e. 200

Total miss per instruction $= (0.2 \times 200) + (0.36 \times 0.04 \times 200) = 6.88$

Faster computer with cache miss will have effective CPI as $2 + 6.88 = 8.88$

$$\frac{P_{fast\,clock}}{P_{slow\,clock}} = \frac{E_{slow\,clock}}{E_{fast\,clock}} = \frac{I \times CPI_{slow\,clock} \times T_c}{I \times CPI_{fast\,clock} \times \frac{T_c}{2}}$$

$$= \frac{5.44 \times 2}{8.8} = 1.23$$

Cache Performance Improvement ...

Performance Improvement Approach

- Two ways to increase cache access performance
  - Introducing more flexible placement of blocks
  - Reducing penalty by introducing multi-level cache
    - Since caches are on the same silicon, accessing it is faster

11

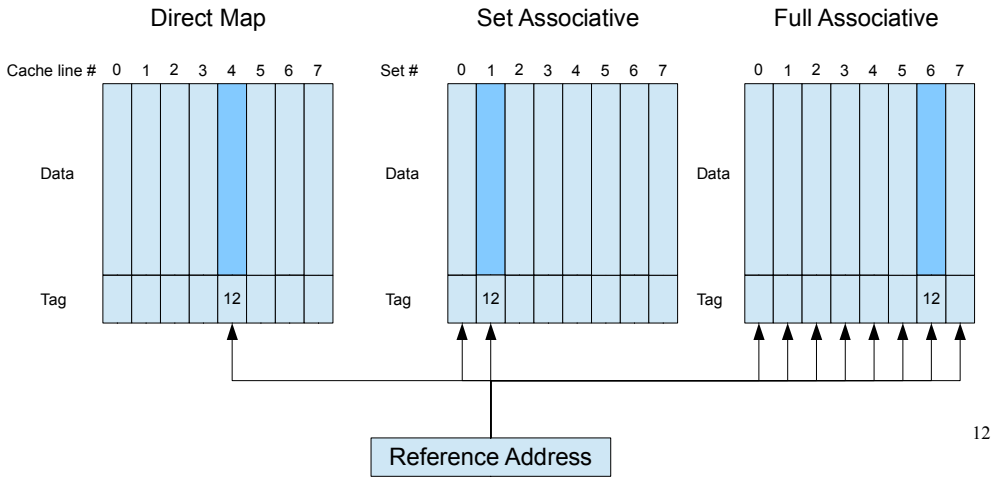
- If we carefully review the cache operation so far, we can exploit the performance bottleneck of the cache system. Since an address can only be mapped into one specific cache line and multiple address can be mapped into same cache line, there is a situation when rest of the cache line is empty but cache line replacement is happening on the same line. One way to eliminate that the compiler optimization re-layout the memory to target different variables in the program into different cache line.

- The hardware approach is to put more flexible block placement policy. If we do not restrict ourselves to be mapped onto single cache line only for a specific memory address, we can possibly utilize empty cache lines upon a conflict (cache line replacement situation).

- The other improvement is to use multiple cache levels. Usually the main memory technology is slower and the cache is faster. Therefore if we put more levels in between cache near to the processor in the hierarchy (level one cache – or L1 cache) and the main memory we can take advantage of storing larger blocks on a faster part of the memory system. Systems now a days have L2, L3 cache on the same silicon as the processor.

Flexible Block Placement

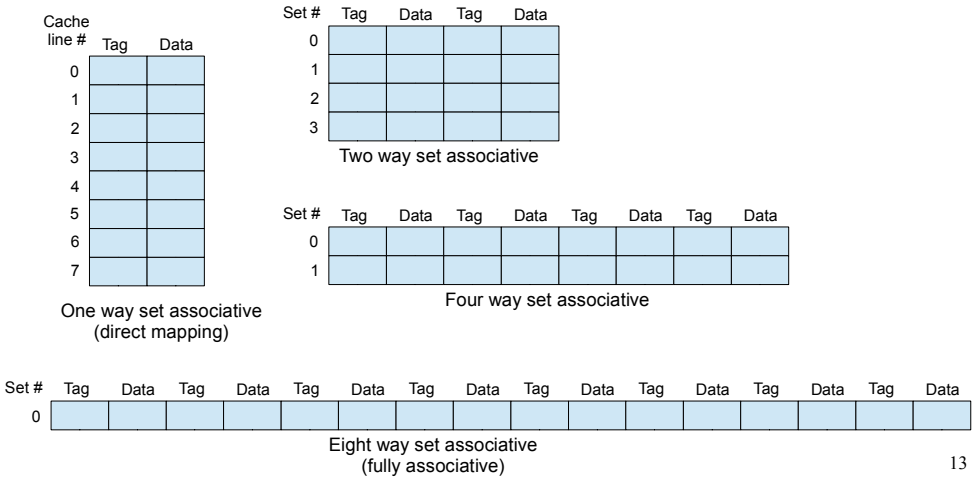
- Increase the search space



• Flexible block placement reduces the cache miss my providing multiple alternative to place a block into a cache line (it become many to may mapping relation). For a direct mapping an address can only be mapped into one cache line. For associative cache it can be mapped into multiple (2,4,8, …) cache line or all the cache line (fully associative). For an associative cache, we do not use the term cache line, instead we call it a set number.

Associative Cache

- Re-organize the direct map cache into sets



13

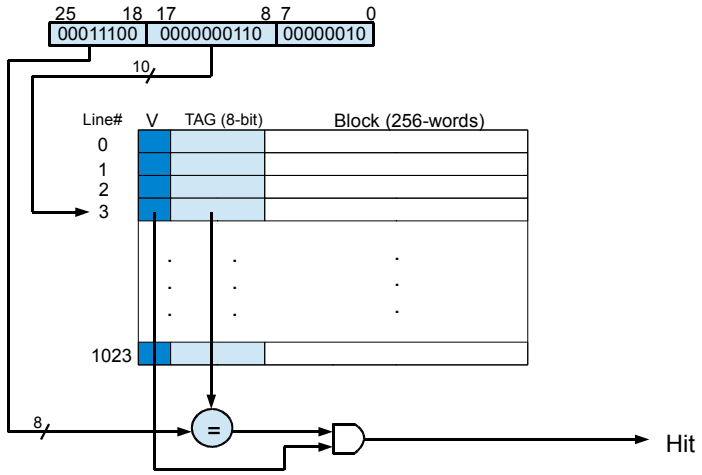
- We can achieve associativity by simple reorganization of cache lines into set. For a direct mapping or one way associative cache, the structure remains same as we have reviewed. Each index number of cache points to single tag and data.

- For a two way associative cache, each index points to a set of two tags and data. Similarly a 4 way associative cache contains a set of 4 tags and data per index. A fully associative cache contains only one set and a block can be placed into any place in the cache.

- Please note that we keep the number of cache line stored same for any such associative cache, only the associativity increased (in other words number of sets decreased).

Associative Cache Circuit

- One way associative or direct mapping Cache

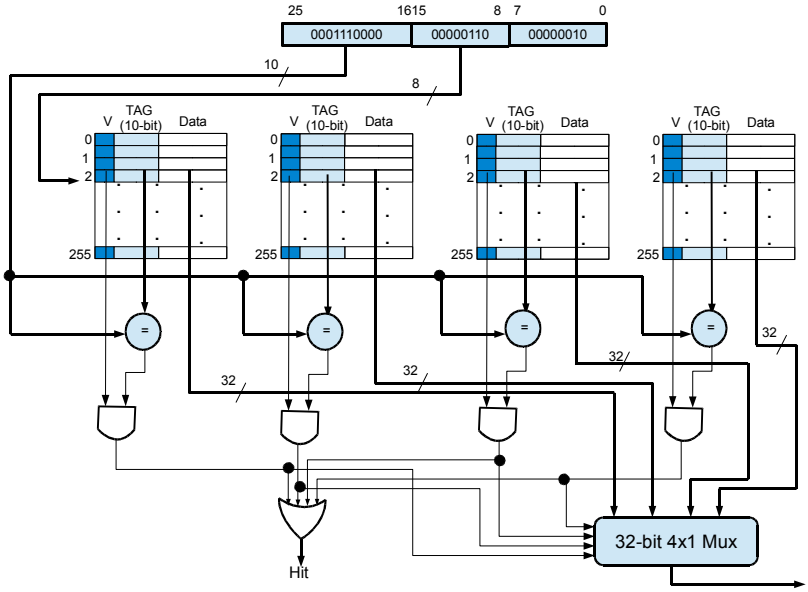


14

- This is what has been reviewed earlier – a direct mapping (or one way associative cache).

Associative Cache Circuit

- Four way associative Cache



15

- We can reorganize the same 1K cache dividing it into 255 sets having 4 way association. In this case set number if located using 8 bit and tag stored is 10 bit. All the tags are compared to determine if the referenced address is present in the cache. If present, corresponding data is send out from cache using a 32-bit 4x1 mux (1-hot mux in this case) where selection function is logical and between valid bit and the comparison result out of each element in the set.

Line Replacement Strategy

- If a cache line needs to be replaced which one it would be?
  - Least Recently Used (LRU)
  - Random

16

- In a LRU strategy, least recently used line is replaced in a set. Additional control bit is needed to keep track of which one is least recently used. Usually it is almost impossible to take LRU approach for more than 4 way association due to hardware limitation. Even for 4 way association, approximation of LRU is taken.

- Random replacement strategy is much simpler to handle – the randomness is hardware generated. One of the line from the set is replaced randomly if replacement is needed. Statistical analysis shows that random replacement incurs only 1.1 higher miss rate than LRU.

CS147 - Lecture 22

Kaushik Patra
(kaushik.patra@sjsu.edu)

17

- Cache Performance
- Improving Cache Performance

Reference Books / Source:

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*