

CSEN 702: Microprocessors
Winter 2023
Practice assignment 3-Solution

Exercise 1

Assume we have a computer where the cycles per instruction (CPI) is 1.0 when all memory accesses hit in the cache. The only data accesses are loads and stores, and these total 50% of the instructions. If the miss penalty is 25 clock cycles and the miss rate is 2%, how much faster would the computer be if all instructions were cache hits?

Solution:

Case 1: All accesses are hits, memory stall cycles = 0.

$$\begin{aligned}\text{CPU execution time}_{\text{without cache misses}} &= (\text{CPU clock cycles} + \text{Memory stall cycles}) * \text{Clock cycle time} \\ &= (\text{IC} * \text{CPI} + 0) * \text{Clock cycle time} \\ &= 1.0 * \text{IC} * \text{CCT} \quad (\text{CPI} = 1.0)\end{aligned}$$

Case 2: Not all accesses are hits, memory stall cycles > 0 and we have to compute it.

$$\begin{aligned}\text{Memory stall cycles} &= \text{IC} * (\text{Memory accesses/instruction}) * \text{Miss rate} * \text{Miss penalty} \\ &= \text{IC} * (1.0 + 0.5) * 0.02 * 25 \\ &= \text{IC} * 0.75\end{aligned}$$

Note: Memory accesses/instruction = (1 + 0.5), because each instruction of any type needs to access the instruction cache (remember IF stage). Loads and stores require an extra access to the data cache. Since only 50% of the instructions are loads and stores, this means that the average memory accesses per instruction = 1 + 0.5 = 1.5

$$\begin{aligned}\text{CPU execution time}_{\text{with cache misses}} &= (\text{CPU clock cycles} + \text{Memory stall cycles}) * \text{Clock cycle time} \\ &= (\text{IC} * 1.0 + \text{IC} * 0.75) * \text{Clock cycle time} \\ &= 1.75 * \text{IC} * \text{CCT}\end{aligned}$$

The performance ratio is the inverse of the execution times:

$$\frac{\text{CPU execution time}_{\text{cache}}}{\text{CPU execution time}} = \frac{1.75 \times \text{IC} \times \text{Clock cycle}}{1.0 \times \text{IC} \times \text{Clock cycle}} \\ = 1.75$$

The computer with *no* cache misses is 1.75 times faster.

Exercise 2

Assume that the cache miss penalty is 200 clock cycles, and all instructions normally take 1.0 clock cycles (ignoring memory stalls).

Assume that the average miss rate is 2%, there is an average of 1.5 memory references per instruction, and the average number of cache misses per 1000 instructions is 30. What is the impact on performance when behavior of the cache is included? Also what's the impact when no cache is used at all.

Hint: Calculate the impact using both misses per instruction and miss rate.

Solution:

Case 1: Cache is included

$$\begin{aligned} \text{CPU time} &= \text{IC} * (\text{CPI} + \text{Memory stall cycles/instruction}) * \text{Clock cycle time} \\ &= \text{IC} * (\text{CPI} + \text{Misses/instruction} * \text{Miss penalty}) * \text{Clock cycle time} \end{aligned}$$

Using the misses per instruction number, we get:

$$\begin{aligned} \text{CPU time} &= \text{IC} * [1 + (30/1000) * 200] * \text{Clock cycle time} \\ &= \text{IC} * 7 * \text{Clock cycle time.} \end{aligned}$$

Using the miss rate number, the formula can be re-written as:

$$\begin{aligned} \text{CPU time} &= \text{IC} * (\text{CPI} + (\text{Memory accesses/instruction}) * \text{Miss rate} * \text{Miss penalty}) * \text{CCT} \\ &= \text{IC} * [1 + 0.02 * (1.5) * 200] * \text{Clock cycle time} \\ &= \text{IC} * 7 * \text{Clock cycle time} \end{aligned}$$

Of course, we got the same answer in both, which is the CPU time increases 7 times due to stalls coming from cache misses, compared with a perfect cache that doesn't miss.

Case 2: Cache is not included

Meaning that every memory instruction needs to stall for 200 cycles, the CPU time would be:

CPU time = IC * (1 + 1.5 * 200) * Clock cycle time = IC * 301 * Clock cycle time.

Compare 301 to only 7 when using caches to conclude the importance of caches in performance.

Exercise 3

What is the impact of two different cache organizations on the performance of a processor?

Assume that the CPI with a perfect cache is 1.6, the clock cycle time is 0.35 ns, there are 1.4 Memory references per instruction, the size of both caches is 128 KB, and both have a block size of 64 bytes.

One cache is *direct mapped* and the other is *two-way set associative*.

Since the speed of the processor can be tied directly to the speed of a cache hit, assume the processor clock cycle time must be stretched 1.35 times to accommodate the added selection hardware of the set associative cache.

The cache miss penalty is 65 ns for either cache organization.

Assume the *hit time* is 1 clock cycle, the *miss rate* of a direct-mapped 128 KB cache is 2.1%, and the *miss rate* for a two-way set associative cache of the same size is 1.9%.

1) Calculate the average memory access time. Discuss the result.

2) Calculate the processor performance. Discuss the result.

Solution:

1) The formula to find the average memory access time is:

$$\text{Avg. memory access time} = \text{Hit time} + \text{Miss rate} * \text{Miss penalty}$$

Hits are considered to take 1 clock cycle, and 1 cycle takes 0.35 ns. Therefore, the hit time = 1 * 0.35 = 0.35.

Direct mapped:

$$\text{Avg. memory access time} = 0.35 + 0.021 * 65 = 1.72 \text{ ns}$$

2-Way Set associative:

$$\text{Avg. memory access time} = 0.35 * 1.35 + 0.019 * 65 = 1.71 \text{ ns}$$

(Remember that the clock cycle for hitting the set associative is longer by 1.35 times due to added hardware)

Conclusion: The set associative is slightly better than direct mapped in terms of access times.

2) The formula of CPU time is just like in exercise 1.

$$\text{CPU time} = \text{IC} * (\text{CPI} + \text{Miss rate} * (\text{Memory accesses/instruction}) * \text{Miss penalty}) * \text{CCT}$$

However, the miss penalty in the above formula is expressed in cycles, while in the given it's expressed in seconds which means it is the combined (miss penalty * clock cycle time) so we must split our equation to accommodate that. Hence, we get:

$$\text{CPU time} = \text{IC} * [(\text{CPI}) * \text{Clock cycle time} + \text{Miss rate} * (\text{Mem accesses/instr.}) * 65]$$

Direct mapped:

$$\text{CPU time} = \text{IC} * [(1.6) * 0.35 + 0.021 * (1.4) * 65] = 2.47 * \text{IC}$$

2-way set associative:

$$\text{CPU time} = \text{IC} * [(1.6) * 0.35 * 1.35 + 0.019 * (1.4) * 65] = 2.49 * \text{IC} \quad \text{which is longer than Direct mapped!}$$

Conclusion: In contrast to the results of average memory access time comparison, the direct mapped cache leads to slightly better average performance because the clock cycle is stretched for all instructions for the two-way set associative case, even if there are fewer misses. Since CPU time is our bottom-line evaluation and since direct mapped is simpler to build, the preferred cache is direct mapped in this exercise.

Exercise 4

Compare the two below schemes by comparing A) the miss rates, and B) average memory access time.

- Scheme 1: 16 KB instruction cache with a 16 KB data cache.
- Scheme 2: 32 KB unified cache.
- Both caches are write-through, use the table below for misses per 1000 instructions.
- 36% of the instructions are data transfer instructions.
- Assume a hit takes 1 clock cycle and the miss penalty is 200 clock cycles.
- A load or store hit takes 1 extra clock cycle on a unified cache if there is only one cache port to satisfy two simultaneous requests. *Table 1. Misses per 1000 instructions.*

Size (KB)	Instruction cache	Data cache	Unified cache
8	8.16	44.0	63.0
16	3.82	40.9	51.0
32	1.36	38.4	43.3

Solution

A) Comparing the *miss rates*:

Recall that:

$$(\text{Misses/instruction}) = \text{Miss rate} * (\text{Memory accesses/instruction})$$

Therefore:

$$\text{Miss rate} = (\text{Misses/instruction}) / (\text{Memory accesses/instruction})$$

And because the numbers in the table represent *misses/1000 instructions*, we have to divide over 1000 to get *misses/instruction*.

Case 1: Split cache (16 KB instruction and 16 KB data)

$$\text{Miss rate}_{16 \text{ KB instruction}} = (3.82/1000) / 1.00 = 0.004$$

All instructions access the instruction cache once, hence the division by 1. (memory accesses per instruction = 1).

$$\text{Miss rate}_{16 \text{ KB data}} = (40.9/1000) / 0.36 = 0.114$$

Only 36% of instructions access the data cache since they are data transfer instructions. Hence the 0.36. (memory accesses per instruction is 0.36).

- However, It's better to find a unified value for miss rate for split cache scheme. To compute the overall miss rate, we need to know the percentage of accesses to memory as if it was a one component and not split.
- How many instructions and how many data? Let's say we have 100 instructions issues:
- 100 instructions references memory access
- 36 additional access for loads and stores -> Total 136 accesses. 100 out 136 of them instructions (=74%), 36 out 136 (=26%) of them data
- N.B. This also applies to unified cache.

$$\text{Overall Miss Rate for split caches} = (74\% * 0.004 + 26\% * 0.114) = 0.0326$$

Case 2: Unified cache (32 KB)

$$\text{Miss rate}_{32 \text{ KB unified}} = (43.4/1000) / (1.0 + 0.36) = 0.0318$$

Each instruction requires 1 access to get the instruction and then 36% of those require an extra access to the memory => $1 + 0.36$

Verdict: Miss rate of unified cache (0.0318) is < the split caches (0.0326) but let's dig further!

B) Comparing the *average memory access times*:

The formula to find the average memory access time is:

$$\text{Avg. memory access time} = \text{Hit time} + \text{Miss rate} * \text{Miss penalty}$$

$$\begin{aligned} &= \\ &= \% \text{ instructions} \times (\text{Hit time} + \text{Instruction miss rate} \times \text{Miss penalty}) \\ &\quad + \% \text{ data} \times (\text{Hit time} + \text{Data miss rate} \times \text{Miss penalty}) \end{aligned}$$

Annotations for the formula above:

- 74% points to "% instructions"
- 26% points to "% data"
- 1 points to "Hit time" (in both terms)
- 1 or 2 depending on split/unified points to "Miss rate" (in both terms)
- 200 points to "Miss penalty" (in both terms)

Case 1: Split cache (16 KB instruction and 16 KB data)

$$\begin{aligned} \text{Avg. memory access time}_{\text{split}} &= 74\% * (1 + 0.004 * 200) + 26\% * (1 + 0.114 * 200) \\ &= (74\% * 1.8) + (26\% * 23.8) = 1.332 + 6.188 = 7.52 \end{aligned}$$

Case 2: Unified cache (32 KB)

$$\begin{aligned} \text{Avg. memory access time}_{\text{split}} &= 74\% * (1 + 0.0318 * 200) + 26\% * (1 + 1 + 0.0318 * 200) \\ &= (74\% * 7.36) + (26\% * 8.36) = 5.446 + 2.174 = 7.62 \end{aligned}$$

Verdict: Access time of unified is longer, which makes the split better! Different from what we got only relying on miss rates.