HW7.3. Cache Properties

How do the following caches compare in hit time and hit rates to a 64 KiB, 8-way set associative cache with 1 KiB blocks?

Q1: A 64 KiB, direct-mapped cache with 1 KiB blocks
How does cache associativity affect the hit time/hit rate?

Q1.1: Hit Time:

- (a) Increased Hit Time
- (b) Minimal effect/Could increase or decrease the Hit Time
- [✓ (c) Decreased Hit Time] ✓

✓ 100%

Q1.2: Hit Rate:

- (a) Increased Hit Rate
- (b) Minimal effect/Could increase or decrease the Hit Rate
- [✓ (c) Decreased Hit Rate] ✓

✓ 100%

Q2: A 128 KiB, 16-way set associative cache with 1 KiB blocks
How does cache capacity affect the hit time/hit rate?

Q2.1: Hit Time:

- [✓ (a) Slightly Increased Hit Time] ✓
- (b) Minimal effect/Could increase or decrease the Hit Time
- (c) Slightly Decreased Hit Time

✓ 100%

Q2.2: Hit Rate:

- [✓ (a) Increased Hit Rate] ✓
- (b) Minimal effect/Could increase or decrease the Hit Rate
- (c) Decreased Hit Rate

✓ 100%

Q3: A 64 KiB, 8192-way set associative cache with 1 B blocks
What happens if we have a very large cache associativity but very small block size?

Q3.1: Hit Time:

- [✓ (a) Increased Hit Time] ✓
- (b) Minimal effect/Could increase or decrease the Hit Time
- (c) Decreased Hit Time

✓ 100%

Q3.2: Hit Rate:

- (a) Most likely increased Hit Rate
- (b) Minimal effect/Could increase or decrease the Hit Rate
- [✓ (c) Most likely decreased Hit Rate] ✓

✓ 100%

Recall the formula for the average memory access time:

$$AMAT = HitTime + MissRate * MissPenalty$$

The processor always checks the cache first. If there is a hit, it will only need to access the cache (thus the inclusion of hit time in the formula). If there is a miss, then the cache has to access the next level of memory first (the main memory if there's only one cache, or the next level of cache L2, L3, and so on) to fetch the block (the miss penalty is the penalty in accessing the next level of memory during misses).

Total points:	70/100
Score:	70%

Question

Value:	20
History:	20
Awarded points:	20/20

Report an error in this question

Attached files

No attached files

Attach a file

Attach text

Q4: For this question, assume that we have 1 MiB of main memory, and that accessing main memory takes 100 clock cycles.

Q4.1: We add a 1 KiB cache which has a hit time of 5 clock cycles; our miss penalty is still the 100 clock cycles needed to access main memory. To test this cache, we then run a program that accesses **random memory addresses**. To the nearest clock cycle, what does the AMAT converge to as the program runs indefinitely?
Hint: If the memory accesses are purely random, are you exploiting the spatial/temporal locality offered by caches? What happens to your miss rate?

105 ? ✓ 100%

Q4.2: We test the above cache on another program, which exhibits an 80% hit rate. What is our AMAT when running this program on this cache?
Hint: How does hit rate relate to miss rate which is part of the AMAT calculation?

25 ? ✓ 100%

Q4.3: In order to "break even" (having a cache is more efficient than the naive solution of directly accessing the main memory), what would you need your hit rate to be?
Hint: This is the point where AMAT = main memory access time (miss penalty).

0.05 ? ✓ 100%

Q4.4: We add an L2 cache (the earlier cache is the L1 cache), which has a hit time of 20 cycles. We run a program that exhibits an 80% L1 hit rate, and a 95% L2 hit rate (note that the L2 hit rate only counts accesses that miss on the L1 cache). Our miss penalty is still the 100 clock cycles needed to access main memory. What is the AMAT for this cache setup on this program?
Hint: L1 cache is closer to the processor. If it misses, it accesses the L2 cache. If L2 cache misses, it accesses the main memory. How would AMAT calculation change in this configuration?

10 ? ✓ 100%

Try a new variant

Correct answer

Q1: A 64 KiB, direct-mapped cache with 1 KiB blocks
How does cache associativity affect the hit time/hit rate?

Q1.1: Hit Time:
(c) Decreased Hit Time

Q1.2: Hit Rate:
(c) Decreased Hit Rate

Q1: A decreased associativity generally means we need to check fewer spots in our cache (so faster hits), but also increases conflict misses.

Q2: A 128 KiB, 16-way set associative cache with 1 KiB blocks
How does cache capacity affect the hit time/hit rate?

Q2.1: Hit Time:
(a) Slightly Increased Hit Time

Q2.2: Hit Rate:
(a) Increased Hit Rate

Q2: Increasing the cache size generally increases the hit rate, but also means we need to check more data, and thus the cache tends to be slower.

Q3: A 64 KiB, 8192-way set associative cache with 1 B blocks

What happens if we have a very large cache associativity but very small block size?

Q3.1: Hit Time:
(a) Increased Hit Time

Q3.2: Hit Rate:
(c) Most likely decreased Hit Rate

Q3: This cache will increase the hit time, since we now have a thousand times more blocks (there would likely be more metadata in this cache than actual data!). Hit rate would also likely suffer, since we essentially give up on spatial locality, though there are access patterns which would cause an increased hit rate.

Generally, the rule for caches is that any variable we can change causes a trade-off between speed and hit rate, though extreme caches tend to suffer in both metrics. As with many things, a middle-ground approach tends to be most efficient on the average use case. Since the cache is a hardware component more often than not, we can't modify the cache properties after we design the CPU, so these design considerations are especially important.

Recall the formula for the average memory access time:

$$AMAT = HitTime + MissRate * MissPenalty$$

The processor always checks the cache first. If there is a hit, it will only need to access the cache (thus the inclusion of hit time in the formula). If there is a miss, then the cache has to access the next level of memory first (the main memory if there's only one cache, or the next level of cache L2, L3, and so on) to fetch the block (the miss penalty is the penalty in accessing the next level of memory during misses).

Q4: For this question, assume that we have 1 MiB of main memory, and that accessing main memory takes 100 clock cycles.

Q4.1: We add a 1 KiB cache which has a hit time of 5 clock cycles; our miss penalty is still the 100 clock cycles needed to access main memory. To test this cache, we then run a program that accesses **random memory addresses**. To the nearest clock cycle, what does the AMAT converge to as the program runs indefinitely?
Hint: If the memory accesses are purely random, are you exploiting the spatial/temporal locality offered by caches? What happens to your miss rate?

`105`

Q4.1: With a random access pattern, the chance that we get a hit is around 1/1000 (since we have 1/1000 of main memory in our cache). As such, we end up with an AMAT of about 105 clock cycles. This is generally true for any cache; regardless of the cache properties, they will perform worse than the naive solution on a random sequence of memory accesses. The only reason why a cache is better is that "useful" memory access patterns tend to be very much nonrandom, so we can trade off 5% slower runtime on 99.999% of cases for 10-20x faster runtime on the 0.001% of cases which most useful memory access patterns actually fall into.

Q4.2: We test the above cache on another program, which exhibits an 80% hit rate. What is our AMAT when running this program on this cache?
Hint: How does hit rate relate to miss rate which is part of the AMAT calculation?

`25`

Q4.2: Let's imagine that we have a representative sample of 5 memory accesses; 4 of them are hits, and 1 would be a miss. Each hit takes 5 cycles to complete, while the miss would take 105 cycles (5 cycle miss penalty + 100 cycles to get to main memory). It thus takes 125 cycles for every 5 memory accesses, or 25 cycles per access.

You can also just directly apply the AMAT formula: Hit time + (Miss Rate)*(Miss Penalty). Miss Rate = 1 - Hit Rate. So, we have: 5 + (1-0.8)*(100) = 25.

Q4.3: In order to "break even" (having a cache is more efficient than the naive solution of directly accessing the main memory), what would you need your hit rate to be?
Hint: This is the point where AMAT = main memory access time (miss penalty).

`0.05`

Q4.3: We need our AMAT to be 100, so we have $5 + (1 - x)(100) = 100$, so $0.95 = 1 - x$, so $x = 0.05$.

Q4.4: We add an L2 cache (the earlier cache is the L1 cache), which has a hit time of 20 cycles. We run a program that exhibits an 80% L1 hit rate, and a 95% L2 hit rate (note that the L2 hit rate only counts accesses that miss on the L1 cache). Our miss penalty is still the 100 clock cycles needed to access main memory. What is the AMAT for this cache setup on this program?
Hint: L1 cache is closer to the processor. If it misses, it accesses the L2 cache. If L2 cache misses, it accesses the main memory. How would AMAT calculation change in this configuration?

```
10
```

Q4.4: Let's imagine that we have 100 memory accesses. Of the 100 accesses, 80 hit on L1 (and finish in 5 cycles), 19 hit on L2 (and finish in 5+20 = 25 cycles), and one misses both (and finishes in 5+20+100=125 cycles). Our total runtime for these 100 accesses is thus $(80 * 5 + 19 * 25 + 125) = (400 + 475 + 125) = 1000$ cycles, for an AMAT of 10 cycles.

You can also just directly apply the AMAT formula on 2-level caches: L1 hit time + (L1 miss rate)*(L2 hit time + (L2 miss rate)*(Miss penalty)). L1 miss rate is 1-0.8 = 0.2. L2 miss rate is 1-0.95 = 0.05. Thus, we have: 5 + 0.2(20+0.05(100)) = 10.

Submitted answer 4 correct: 100%
Submitted at 2022-10-29 06:17:59 (PDT)

hide

Q1: A 64 KiB, direct-mapped cache with 1 KiB blocks
How does cache associativity affect the hit time/hit rate?

Q1.1: Hit Time:
(c) Decreased Hit Time ✓ 100%

Q1.2: Hit Rate:
(c) Decreased Hit Rate ✓ 100%

Q2: A 128 KiB, 16-way set associative cache with 1 KiB blocks
How does cache capacity affect the hit time/hit rate?

Q2.1: Hit Time:
(a) Slightly Increased Hit Time ✓ 100%

Q2.2: Hit Rate:
(a) Increased Hit Rate ✓ 100%

Q3: A 64 KiB, 8192-way set associative cache with 1 B blocks
What happens if we have a very large cache associativity but very small block size?

Q3.1: Hit Time:
(a) Increased Hit Time ✓ 100%

Q3.2: Hit Rate:
(c) Most likely decreased Hit Rate ✓ 100%

Recall the formula for the average memory access time:

$$AMAT = HitTime + MissRate * MissPenalty$$

The processor always checks the cache first. If there is a hit, it will only need to access the cache (thus the inclusion of hit time in the formula). If there is a miss, then the cache has to access the next level of memory first (the main memory if there's only one cache, or the next level of cache L2, L3, and so on) to fetch the block (the miss penalty is the penalty in accessing the next level of memory during misses).

Q4: For this question, assume that we have 1 MiB of main memory, and that accessing main memory takes 100 clock cycles.

Q4.1: We add a 1 KiB cache which has a hit time of 5 clock cycles; our miss penalty is still the 100 clock cycles needed to access main memory. To test this cache, we then run a program that accesses **random memory addresses**. To the nearest clock cycle, what does the AMAT converge to as the program runs indefinitely?
Hint: If the memory accesses are purely random, are you exploiting the spatial/temporal locality offered by caches? What happens to your miss rate?

`105` ✓ 100%

Q4.2: We test the above cache on another program, which exhibits an 80% hit rate. What is our AMAT when running this program on this cache?
Hint: How does hit rate relate to miss rate which is part of the AMAT calculation?

`25` ✓ 100%

Q4.3: In order to "break even" (having a cache is more efficient than the naive solution of directly accessing the main memory), what would you need your hit rate to be?
Hint: This is the point where AMAT = main memory access time (miss penalty).

`0.05` ? ✓ 100%

Q4.4: We add an L2 cache (the earlier cache is the L1 cache), which has a hit time of 20 cycles. We run a program that exhibits an 80% L1 hit rate, and a 95% L2 hit rate (note that the L2 hit rate only counts accesses that miss on the L1 cache). Our miss penalty is still the 100 clock cycles needed to access main memory. What is the AMAT for this cache setup on this program?
Hint: L1 cache is closer to the processor. If it misses, it accesses the L2 cache. If L2 cache misses, it accesses the main memory. How would AMAT calculation change in this configuration?

`10` ✓ 100%

Submitted answer 3 **partially correct: 90%**
Submitted at 2022-10-29 06:16:15 (PDT)
show

Submitted answer 2 **partially correct: 70%**
Submitted at 2022-10-29 06:14:08 (PDT)
show

Show/hide older submissions