## HW7.4. Cache Programming

Q1: The following C program is run (with no optimization) on a processor with a direct-mapped data cache (**the cache starts out empty**) with a size of 1 KiB and a block size of 32 bytes:

```
int i, j, array[256*256];

/* ... */

for (i = 1 ; i < 256 ; i++) {
    for (j = 0 ; j < 256 ; j++) {
        array[256*j] += array[256*j+i];
    }
}
```

Assume `sizeof(int) == 4` and `array == 0x0000 4000`.
*The sizeof(int) is highly relevant for the following questions. How many ints can fit in a cache block?*

Q1.1: For the first iteration of the outer loop (`i = 1`), what is the hit rate of this code?
*Hint: How would you expand the expression array[x] += array[y]? How many memory accesses are in there? Which of those would miss?*

| Hit Rate= | 0.67 | ✓ 100% |
|---|---|---|

Q1.2: After the first `n` iterations of the outer loop, the hit rate changes. What is `n`?
*Hint: Try writing out the array indices for every iteration of the loop. Recall the sizeof(int). At what point will array[256*j+i] exceed a cache block?*

| n= | 7 | ✓ 100% |
|---|---|---|

Q1.3: After the first `n` iterations of the outer loop, the hit rate changes. What is the new hit rate of each iteration of the outer loop?
*Hint: From 1.2: if array[256*j+i] exceeds a cache block, how many misses would we have?*

| Hit Rate= | 0.33 | ✓ 100% |
|---|---|---|

Q1.4: What is the overall hit rate of this code?
*Hint: How many total memory accesses do we have for one iteration of the outer loop? How would you incorporate the pattern you observed from the previous questions?*

| Hit Rate= | 0.34 | ✓ 100% |
|---|---|---|

Q2: We decide to rewrite our code to be more cache-efficient, while maintaining the same behavior. Fill in the blanks of this code (write your answers in the same format as the original code):

```
int i, j, array[256*256];

/* ... */

for ([CODE A]; i < 256 ; i++) {
    for ([CODE B]; j < 256 ; j++) {
        array[[CODE C]] += array[[CODE D]];
    }
}
```

*Hint: Coming from the previous questions, we now want to have more consecutive memory accesses on the inner loop so that we get more cache hits.*

| Q2.1: | CODE A: | i = 0 | ✓ 100% |
|---|---|---|---|
| Q2.2: | CODE B: | j = 1 | ✓ 100% |
| Q2.3: | CODE C: | 256 * i | ✓ 100% |

Homework 7

Assessment overview

Total points: 100/100

Score: 100%

Question

Value: 30

History: 30

Awarded points: 30/30

Report an error in this question

Previous question

Next question

Attached files

*No attached files*

Attach a file

Attach text

Q2.4: CODE D: 256 * i + j ✓100%

Q2.5: What is the new hit rate of this code?
*Hint: How many total memory accesses do we have for one iteration of the outer loop? How many misses do we get now after code reordering?*

Hit Rate= 0.96 ✓100%

Try a new variant

## Correct answer

Assume `sizeof(int) == 4` and `array == 0x0000 4000`.
*The sizeof(int) is highly relevant for the following questions. How many ints can fit in a cache block?*

Q1.1: For the first iteration of the outer loop (`i = 1`), what is the hit rate of this code?
*Hint: How would you expand the expression array[x] += array[y]? How many memory accesses are in there? Which of those would miss?*

Hit Rate= `0.67`

**Q1.1: In our first iteration of the inner loop, we get a miss on the first step, and two hits after, since all three accesses are in the same block, noting that the "+=" requires that we access the first memory address twice (once for read, once for write). Each iteration of the inner loop uses the same set of our cache, so no block stays in our cache for longer than 1 iteration of the inner loop.**

Q1.2: After the first `n` iterations of the outer loop, the hit rate changes. What is `n`?
*Hint: Try writing out the array indices for every iteration of the loop. Recall the sizeof(int). At what point will array[256*j+i] exceed a cache block?*

n= 7

Q1.3: After the first `n` iterations of the outer loop, the hit rate changes. What is the new hit rate of each iteration of the outer loop?
*Hint: From 1.2: if array[256*j+i] exceeds a cache block, how many misses would we have?*

Hit Rate= `0.33`

**Q1.2-1.3: We similarly stay in the same block for 7 iterations of the outer loop. On the 8th loop and on, we move to a second block for our second memory access, so we end up with two misses and one hit. Note that since our cache size is exactly equal to 256 ints, we never evict the first block on an iteration, so we always get one hit from each set of three accesses.**

Q1.4: What is the overall hit rate of this code?
*Hint: How many total memory accesses do we have for one iteration of the outer loop? How would you incorporate the pattern you observed from the previous questions?*

Hit Rate= `0.34`

**Q1.4: Our overall hit rate is thus** $\frac{2*7+(255-7)}{3*255} = \frac{262}{765}$

Q2.1: CODE A: `i=0`

Q2.2: CODE B: `j=1`

Q2.3: CODE C: `256*i`

Q2.4: CODE D: `256*i+j`

**Q2.1-2.4: We can improve our cache efficiency by swapping our loops; now the inner loop has more consecutive memory accesses.**

Q2.5: What is the new hit rate of this code?

*Hint: How many total memory accesses do we have for one iteration of the outer loop? How many misses do we get now after code reordering?*

Hit Rate= `0.96`

**Q2.5: Each iteration of the inner loop accesses the same set of 256 consecutive indexes, which fits perfectly in our cache. As such, we get $256/8 = 32$ misses per iteration of the outer loop, for a total hit rate of $\frac{765-32}{765} = \frac{733}{765}$**

**Note the significant improvement we get on miss rate. On an average system, L1 Cache hits take about 5-10 cycles, while disk accesses take about 100-300 cycles to resolve. Assuming that disk accesses take 100 cycles to resolve and L1 caches take 5 cycles to resolve, we speed up our memory access time by a factor of about 7.7, all by reordering our code!**

Submitted answer **correct: 100%**

Submitted at 2022-10-29 06:32:44 (PDT)

hide

Assume `sizeof(int) == 4` and `array == 0x0000 4000`.
*The sizeof(int) is highly relevant for the following questions. How many ints can fit in a cache block?*

Q1.1: For the first iteration of the outer loop (`i = 1`), what is the hit rate of this code?
*Hint: How would you expand the expression array[x] += array[y]? How many memory accesses are in there? Which of those would miss?*

Hit Rate= `0.67` ? ✓100%

Q1.2: After the first `n` iterations of the outer loop, the hit rate changes. What is `n`?
*Hint: Try writing out the array indices for every iteration of the loop. Recall the sizeof(int). At what point will array[256*j+i] exceed a cache block?*

n= 7 ✓100%

Q1.3: After the first `n` iterations of the outer loop, the hit rate changes. What is the new hit rate of each iteration of the outer loop?
*Hint: From 1.2: if array[256*j+i] exceeds a cache block, how many misses would we have?*

Hit Rate= `0.33` ? ✓100%

Q1.4: What is the overall hit rate of this code?
*Hint: How many total memory accesses do we have for one iteration of the outer loop? How would you incorporate the pattern you observed from the previous questions?*

Hit Rate= `0.34` ? ✓100%

Q2.1: CODE A: `i = 0` ✓100%

Q2.2: CODE B: `j = 1` ✓100%

Q2.3: CODE C: `256 * i` ✓100%

Q2.4: CODE D: `256 * i + j` ✓100%

Q2.5: What is the new hit rate of this code?
*Hint: How many total memory accesses do we have for one iteration of the outer loop? How many misses do we get now after code reordering?*

Hit Rate= `0.96` ? ✓100%