219223 Computer System Laboratory

Lab 7: Cache

You are to provide answers to all the questions in a file named lab7_ans.pdf

1. Cache simulator

Now that we are more familiar with how caches work, let's get some practice with the cache simulator! First, go to:

https://www.cpe.ku.ac.th/~paruj/219222/cachesim/

(This is a local copy of the original simulator located at https://courses.cs.washington.edu/courses/cse351/cachesim/)

Immediately you'll notice 4 boxes:

- *System Parameters: This lets you play around with the structure/format of the cache
- *Manual Memory Access: This is where you actually make reads and writes to memory
- History: This is an interactive log of previous commands. You can even input commands through this! - Simulation Message: Describes the most recent execution.

*These include 'explain' toggles that walk you through any execution step-by-step.

Before generating, set:

"Address Width" → 6, "Cache Size" → 16, "Block Size" → 4, "Associativity" → 2 (Leave other settings as is)

A) Determine the following:

i) Highest Address in Memory: 0b 0011 1111 **ii)** Number of Sets in Cache: 2

Hit 'Generate' and double check your answer.

B) We want to make a read at address 0x2A. Determine the following:

i) The Set containing the block that was read is number 0. ii) The tag bit in this block is 0b 101.

iii) The full 4 bytes in this block are (in order) 0xe9, 0x36, 0xae, 0x32.

Hit 'Read' and double check your answer.

C) We want to write at address 0x1B the value '0xB1'. Determine the following:

i) The Set containing the block that was read is number 0. ii) The tag bit in this block is 0b 011.

Hit 'Write' and double check your answer.

iii) Notice that the value stored in the cache is now different from the value stored in memory. What, in the cache, indicates this disparity? Given that this was a write miss, what would have happened if our write miss policy were "No Write-Allocate" instead?

Dirty bit. Write directly to memory and not cache the block starting at 0x18

D) We want to make a read at address 0x01. Determine the following:

i) The Set containing the block that was read is number 0. ii) The tag bit in this block is 0b 000

iii) Will this read cause a conflict in the cache? (Yes) No

iv) If yes, which block will be evicted? (Read made in B) Write made in C

Hit 'Read' and double check your answer.

E) We want to write at address 0x1C the value '0xE9'. Determine the following:

i) The Set containing the block that was read is number 1. ii) The tag bit in this block is 0b 011

iii) Will this write cause a conflict in the cache? Yes (No)

iv) If yes, which block will be evicted? Read made in B Write made in C Read made in D

Hit 'Write' and double check your answer.

As a note, your history should look like this:

```
R(0x2a) = M
W(0x1b, 0xb1) = M
R(0x01) = M
W(0x1c, 0xe9) = M
```

G) Append the following text to the current History:

```
W(0x03, 0xff)
R(0x27)
R(0x10)
W(0x1d, 0x00)
```

Hit Load. You'll notice that appended to each of these memory accesses is " = ?"

Determine if '?' will resolve to Hit (H) or Miss (M) for each execution.

i) `W(0x03, 0xff) =` H ii) `R(0x27) =` M iii) `R(0x10) =` M iv) `W(0x1d, 0x00) =` H

Use the down arrow to check your answer for each

H) The cache, after the 8 executions detailed above, should look like this:

	V	D	T	Cache Data	
Set 0	1	1	0	20 f6 ef ff	2
	1	0	2	b8 bd 1a ca	1
Set 1	1	1	3	e9 00 f6 e5	1
	1	0	4	1a 6f 7e 63	2

The numbers on the right indicate the most recent use of the cache (where 1 was more recent).

i) A LRU replacement policy will evict which block on the next cache conflict? (Block 1) Block 2

ii) What is one benefit of using LRU over Random?

Favors temporal locality, as local variables usually are reused frequently

iii) What is one benefit of using Random over LRU?

Faster and Cheaper

I) If we were to flush the cache right now (don't actually) how many bytes in memory would change? 3 Bytes

How many bytes would change if our "Write Hit" policy were "Write Through" instead of "Write Back"? 0 Bytes

Can you explain why these numbers are the same/different? (if not, try changing the write hit policy and re-running using the same history above).

2.Analyze the following snippets of C code

```
# define N 64

typedef int array_t[N][N];

int sum1(array_t a) {
  int i, j;
  int sum = 0;
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      sum += a[i][j];
  return sum;
}
```

```
int sum2(array_t a) {
  int i, j;
  int sum = 0;
  for (j = 0; j < N; j++)
    for (i = 0; i < N; i++)
      sum += a[i][j];
  return sum;
}

int sum3(array_t a) {
  int i, j;
  int sum = 0;
  for (i = 0; i < N; i+=2)
    for (j = 0; j < N; j+=2)
      sum += (a[j][i] + a[j][i+1] + a[j+1][i] + a[j+1][i+1]);
  return sum;
}
```

Given the above snippets of code, let each of them run on a CPU with 4 KiB direct-mapped cache whose block size is 16 bytes. Calculate the approximate miss rates when sum1, sum2, and sum3 run with N equal to 64 and 60. Focus exclusively on the accesses to the a array and let the starting address of the a array be 0x0800 0000. **Show your work** and fill in the following table.

	Miss Rate		
	sum1	sum2	sum3
N = 64			
N = 60			

3. Analyze matrix multiplication code

Download the zip file matmult.zip, unzip it, and go to the matmult directory. Then,

```
make clean
make
```

Note: if you cannot run the two commands above, you do not have the make utility on your system. In that case, execute the following two commands to compile the relevant C code for this problem:

```
gcc -O3 -Wall -o mm mm.c clock.c fcycmm.c
gcc -O3 -Wall -o bmm bmm.c clock.c fcycbmm.c
```

You will get the two executables, mm and bmm, which are executables for matrix multiplication without and with blocking, respectively.

Run mm:

./mm

Answer the following questions:
- Which loop ordering has the most number of cycles per loop iteration? jki
- Which loop ordering has the highest performance? ikj
- Compare the loop ordering with the highest and lowest performance, how do they differ? The different is they has swap start and end point of the loop
- Why is it that when n gets larger, the value for cycles per loop iteration increases? They have more access to do in array because array is bigger.

Run bmm (take a bit of time for older “slow” computers):

./bmm

Because they start from the biggest it not normal to do in this way

Answer the following questions:

• Why is it that when blocking is employed and the size n increases, the number of cycles per loop iteration for the ijk ordering does not vary as much as when blocking is not employed?

•As we have learned that the number of misses when blocking is used is about $1/(4B)*n^3$
So, when B increases, the number of misses should be decreasing. Why is it that the result does not necessarily reflect this? Because to loop has seperate to each block so it make it faster

4. Code transformation with blocking

From the transpose.c file, compile and run this program:

```
gcc -O3 -o tp transpose.c
./tp
```

The diffrecent of the code is it change to block so it make it faster

Modify the code in transpose.c to measure the time it takes to transpose each of matrices of size 100x100, 200x200, 500x500, 1000x1000, 2000x2000, and 5000x5000. Then, transform the code in transpose.c so that blocking is used to enhance the performance of matrix transpose. Put the new transformed code into transposeB.c file. Run transposeB.c and compare the result with transpose.c regarding the time it takes to transpose each of the matrices above. Discuss the result.

Submission:

- ***Create StudentID_Firstname_lab7 folder, where StudentID is your KU ID and Firstname is your given name***
- ***Put the files to submit lab7_ans.pdf and transposeB.c into this folder***
- ***Zip the folder and submit the zip file to the course's Google Classroom before the due date***