CSM 61C Spring 2020

Caches

Exam Question Compilation

This document is a PDF version of old exam questions by topic, ordered from least difficulty to greatest difficulty.

Questions:

- Spring 2018 Midterm 2 Q5
- Summer 2015 MT2 Q5
- Summer 2018 Final Q6
- Fall 2015 Final MT2-4
- Fall 2017 Midterm 2 Q5
- Fall 2018 Midterm Q3
- Summer 2018 Midterm 2 Q5
- Summer 2018 Final Q1
- Fall 2019 Final Q8
- Summer 2018 Final Q8

Problem 5 $$$ **(17 points)**

You are given following RISC-V Code:

```
// a0:  Integer array location
// a1:  End bound of the array
// a2:  Offset to new location in words
// Assume these registers hold the following values
// at the start of the program:
// a0 -> 0x1000, a1 -> 2048, a2 -> 2048

     add t0, x0, x0
     slli t3, a2, 2
loop:
     beq t0, a1, done
     slli t1, t0, 2
     add t1, t1, a0
     lw t2, 0(t1)
     add t1, t1, t3
     sw t2, 0(t1)
     addi t0, t0, 4
     jal x0, loop
done:
     ...
```

Assume there is enough memory allocated for the array such that there are no memory out of bounds issues. Also assume the code is run on a machine with a 32-bit address space. Questions will only involve the code starting from `loop` and only refer to one data cache. This cache uses a LRU replacement policy and is write allocate unless otherwise stated.

(a) Consider an 8 words/block, 512B **direct-mapped** data cache. The cache starts empty and we run the program above to completion.

(i) Calculate the number of tag, index, and offset bits for this cache.

Tag:_____ Index: _____ Offset:_____

(ii) What is the hit rate of this direct-mapped cache?

_____

(iii) What types of of cache misses occur? Mark all that apply.

- O Capacity
- O Conflict
- O Compulsory

(iv) Assume the cache is emptied and we re-run the program above to completion, but this time with a cache **block size of 4 words**. What is the hit rate of this new cache?

_____

(b) Now consider an 8 words/block, 512B **Two-Way Set Associative** data cache. The cache starts empty and we run the program above to completion.

(i) What is the hit rate of this Two-Way Set Associative cache?

_____

(ii) What types of of cache misses occur? Mark all that apply.

O Capacity O Conflict

O Compulsory

(iii) Assume the cache is emptied and we re-run the program above to completion, but this time with a cache **block size of 4 words**. What is the hit rate of this new cache?

_____

(c) Now consider a 8 words/block, 512B **Four-Way Set Associative** data cache. The cache starts empty and we run the program above to completion.

(i) What is the hit rate of this Four-Way Set Associative cache?

_____

(ii) Assume the Four-Way Set Associative cache is emptied and we re-run the program above to completion, but this time with a **random replacement policy**. How would the hit rate most likely change compared to part c.i?

O Increase O Stay the Same

O Decrease

(d) Consider the 8 words/block, 512B **direct-mapped** data cache again. The cache starts empty and we run the program above to completion, **except this time with a2 initialized to 2056**. What is the hit rate of this direct-mapped cache?

_____

SID: _____

Q5) Caches (17 pts)

Assume we are working in a 32-bit physical address space. We have two possible data caches: cache X is a direct-mapped cache, while cache Y is 2-way associative with LRU replacement policy. Both are 4 KiB caches with 512 B blocks and use write-back and write-allocate policies.

a) Calculate the number of bits used for Tag, Index and Offset:

Cache	Tag bits	Index bits	Offset bits
X			
Y			

Use the code below to answer the following parts. Assume that ints are 4 B and doubles are 8 B.

```
int DOUBLE_ARRAY_SIZE = 2 * 1024;
double double_arr[DOUBLE_ARRAY_SIZE];

for (int i = 0; i < DOUBLE_ARRAY_SIZE; i++) /* loop 1 */
      double_arr[i] = i;
for (int i = 0; i < DOUBLE_ARRAY_SIZE;  i += 8) /* loop 2 */
     double_arr[i] *= double_arr[0];
```

b) What is the hit rate for each cache if we run only loop 1? (hint: they're both the same). What types of misses do we get?

c) What is the hit rate of each cache when you execute loop 2? Assume that you have executed loop 1. Assume the worst case ordering of accesses within a single iteration of the loop if multiple orders are possible. You may leave your answer as an expression involving products and sums of fractions.

X: _____ Y: _____

d) Compute the AMAT for the following system with 3 levels of caches. (You should not need any information from the previous parts of this problem.) Give your answer as a decimal value.

L1$	L2$	L3$	Main Memory
Global miss rate: 50% Hit time: 1ns	Local miss rate: 20% Hit time: 5ns	Local miss rate: 1% Hit time: 15ns	Hit time: 500ns

SID: _____

Question 6: Cache These Hands (16 pts)

A machine with a 19 bit address space has a single 256 B cache. The cache is 4-way set associative with 8 total entries.

1. Determine the number of bits in Tag, Index, and Offset fields for an address on this machine.

Tag: _____ Index: _____ Offset: _____

The following piece of code is executed on the aforementioned machine. This code computes an outer product of a **N x 1** vector A and a **1 x N** vector B, placing the result in a **N x N** matrix C. Use this code to answer the follow questions about the hit rate the code was produced.

For all questions assume the following:

- sizeof (double) == 8
- A = 0x10000
- B = 0x20000
- C = 0x30000
- The cache begins cold before each question.
- Code is executed from left to right.

```
#define N 16

void outer_product (double *A, double *B, double *C) {
      for (int i = 0; i < N; i++) {
                  for (int j` = 0; j < N; j++) {
                              C [j + i * N] = A[ i ] * B[ j ];
                  }
      }
}
```

SID: _____

2. What is the hit rate for executing this code if it uses LRU replacement and is a write back cache with write allocate on a miss? Fill in all blanks for credit.

HR for accesses to A:_____

HR for accesses to B:_____

HR for accesses to C:_____

OVERALL HR: _____

3. What is the hit rate for executing this code if it uses LRU replacement and is a write back cache with no write allocate on a miss? Fill in all blanks for credit.

HR for accesses to A:_____

HR for accesses to B:_____

HR for accesses to C:_____

OVERALL HR: _____

SID: _____

MT2-4: If you do well, it's *clobbering* time! (12 points)

The information for one student in regards to clobbering a single midterm is captured in the data of the following *tightly-packed* struct:

```
typedef struct student {
   int studentID;
   float oldZScore;
   float newZScore;
   int clobber;          /* a value equal to 1 if a student clobbers,
                            0 if otherwise */
} student;
```

SID: _____

We run the following code on a 32-bit machine with a 4 KiB write-back cache. `importStudent()` returns a `struct student` that is in the course roster and that has not been returned by `importStudent()` previously. For simplicity, assume `importStudent()` does not affect the cache.

```
int ARR_SIZE = 512; //Class size rounded down for simplicity
student *61CStudents = (student *) malloc (sizeof(student) * ARR_SIZE);

/* Assume malloc returns a cache block aligned address */
for (int i = 0; i < ARR_SIZE; i++) {                        <=== part I
   61CStudents[i] = importStudent()
}

for (int i = 0; i < ARR_SIZE; i++) {                        <=== part II
   if (61CStudents[i].oldZscore > 61CStudents[i].newZscore){
     61CStudents[i].clobber = 0;
   } else {
     61CStudents[i].clobber = 1;
   }
}
```

a. How many bytes is needed to store the information for a single student?

b. Assume that the block size is 32 B. What is the tag:index:offset breakdown of the cache?
Tag: ____ Index: ____ Offset: ____

c. At the label **`part I`**, assume that `61CStudents` is filled with the correct data. What type of misses will occur among **all** memory accesses during the process? Why?

d. Suppose we run the code again and the cache block size is now 8 B long and the cache is direct-mapped. For the for-loop in **part II**, what is the miss rate in the best case scenario (we want the highest hit rate possible)? What type of misses occur?

e. For the `for`-loop **in part II**, assume that the cache block size is now 128B.
   i. If the cache is direct-mapped, what is the hit rate?

   ii. If the cache is fully associative, what is the hit rate? Does associativity help? Why or why not?

Q5: Do the Monster Cache

a) For the following cache questions, please **bubble** in your answer on the answer sheet:

I. True or False? A fully associative cache has no conflict misses.

II. True or False? A write-back cache must write to memory *immediately* when a block is modified.

For all portions of this question assume that an integer is one word in size and that ALL operations occur from left to right. Consider a 16-way set-associative cache with two-word blocks, 16 sets and a 128 TiB physical byte-addressed address space.

b) When breaking down a physical address into the Tag, Index, and Offset fields, how many bits long is each field? (i.e. what is the T:I:O for the cache?) Write your answer on the blanks provided on your answer sheet.

Now consider the following code segment:

```
void sequence(int* A, int* B) {
    int i;
    //PART C
    for (i = 0; i < 16; i++) {
      B[i] = 2;
      A[i] = 4;
    }
    //PARTS D & E
    for (i = 16; i < 272; i++) {
      B[i] = B[i - 8] + A[i - 8];
      A[i] = B[i - 16] + A[i - 16];
    }
}
```

Let A's address for the following code segment be 0x10000 and B's address be 0x20000 (leading zeroes are omitted from the addresses for conciseness). Assume that an integer is one word in size, that ALL OPERATIONS are evaluated from left to right, and that all of the cache's valid bits are set to zero before running the sequence function. Remember to write all of your answers to the questions below on your answer sheet.

c) What is the hit rate for running the loop below **PART C** using the cache from (b)?

Ⓐ 1/2 Ⓑ 0 Ⓒ 7/8 Ⓓ 3/4 Ⓔ 15/16

d) What is the cache hit rate for cache accesses that occur below **PARTS D & E** when running sequence after **PART C** completes.

e) Assuming the cache and block sizes are held constant, what is the minimum cache associativity that results in a maximum hit rate for the segment of sequence?

Ⓐ 1-way Ⓑ 2-way Ⓒ 4-way
Ⓓ 8-way Ⓔ 16-way Ⓕ 32-way

f) Assume that sequence ran above resulted in a total of 50 accesses (this may or may not be true) and that it was run on a computer with an L1 and L2 cache. Say that the L1 cache has an access time of 10ns, the L2 cache has an access time of 20ns, main memory has an access time of 50ns, the L1 cache has an 80% hit rate, and that the total AMAT for running sequence is 16 ns. What is the local hit rate for the L2 cache? You do not need to know either of the caches' parameters for this question. **Please write your answer as a decimal, up to 2 decimal places, on your answer sheet.**

Q3) *Cache money, dollar bills, y'all.* (18 points; a-c 2pts d-g 3pts)

We have a 32-bit machine, and a 4 KiB direct mapped cache with 256B blocks. We run the following code from scratch, with the cache initially cold, to add up the values of an uninitialized array to see what was there.

```
uint8_t addup() {
    uint8_t A[1024], sum = 0; // 8-bit unsigned
    touch(A);
    for (int i = 0; i < 1024; i++) { sum += A[i]; }
    return sum - 1;
}
```

```
void touch(uint8_t *A) {
    // Touch random location
    // in A between first and
    // last elements, inclusive
    A[random(0, 1023)] = 0;
}   // e.g.,random(0,2)⇒ 0,1,or 2
```

a) Assume **sum** has the smallest possible value after the loop. *What would **addup** return?* _____

b) Let **A = 0x100061C0**. *What cache index is **A[0]**?* _____

c) Let **A = 0x100061C0**. If the cache has a hit at **i=0** in the loop, what is the *maximum value **random** could have returned*? _____

For d and e, assume we don't know where **A** is, and we run the code from scratch again.

d) What's the *fewest number of cache misses* caused by the loop? _____

e) What's the *most number of cache misses* caused by the loop? _____

f) If we change to a fully associative LRU cache, how would c, d, e's values change? (select ONE per col)

c: ◯up ◯down ◯same	d: ◯up ◯down ◯same	e: ◯up ◯down ◯same

g) When evaluating your code's performance, you find an AMAT of 4 cycles. Your L1 cache hits in 2 cycles and it takes 100 cycles to go to main memory. What is the *L1 hit rate*? _____%

SID: _____

Question 5: C.R.E.A.M. (15 pts)

1) A machine has an 8 way set associative cache with 512 B of data. The size of each block is 16 B and there are 8 MiB of main memory. How large are the tag, index, and offset fields for an address on this machine using this cache?

Tag:_____ Index:_____ Offset:_____

2) Now we have a different machine with two caches, an L1 and an L2 cache. Both caches are direct mapped caches. The L1 cache can hold 256 B of data and the L2 cache can hold 4 KiB of data. Assume the following code is run on this machine:

```
#define ARR_SIZE 2048

uint16_t sum (uint16_t *arr) {
    total = 0;
    for (int i = 0; i < ARR_SIZE; i++) {
       total += arr[i];
    }
    return total;
}
```

This produces a hit rate (HR) of ⅞ for the L1 cache and ¾ for the L2 cache. Given that `arr` is a block aligned address and `sizeof (uint16_t) == 2`:

A. What is the blocksize of the L1 cache **in bytes** that produces its hit rate?

L1_BLOCKSIZE: _____

SID: _____

B. Use the variable **Y** to represent the answer to part A. What is the blocksize of the L2 cache **in bytes** that produces its hit rate? Express your answer as a function of **Y** and NOT as a single number.

L2_BLOCKSIZE: _____

Recall that the L1 HR is ⅞ and the L2 HR is ¾. If the L1 Cache has a hit time of 2 cycles, the L2 cache has a hit time of 8 cycles, and main memory has a hit time of 96 cycles:

3) How many total cycles are spent accessing memory on this piece of code? Express your answer in the form **C * 2^{i}**, where **C** is an integer not divisible by 2.

TOTAL_CYCLES: _____

4) If we change the L1 cache from being direct mapped to being fully associative with LRU, how does its HR change on the same code?

Ⓐ Increases Ⓑ Decreases Ⓒ No Change Ⓓ Impossible to tell

Q8) This is for all the money! (15 pts = 3 + 7 + 5)

Assume we have a single-level, 1 KiB direct-mapped L1 cache with 16-byte blocks. We have 4 GiB of memory. An integer is 4 bytes. The array is block-aligned.

```
#define LEN 2048

int ARRAY[LEN];
int main() {
    for (int i = 0; i < LEN - 256; i+=256) {
       ARRAY[i] = ARRAY[i] + ARRAY[i+1] + ARRAY[i+256];
       ARRAY[i] += 10;
    }
}
```

a) Calculate the number of tag, index, and offset **bits** in the L1 cache.

T:___	I:___	O:___

SHOW YOUR WORK

b) What is the hit rate for the code above? Assume C processes expressions left-to-right.

SHOW YOUR WORK

c) You decide to add an L2 cache to your system! You shrink your L1 cache, so it now takes 3 cycles to access L1. Since you have a larger L2 cache, it takes 50 cycles to access L2. The L1 cache has a hit rate of 25% while the L2 cache has a hit rate of 90%. It takes 500 cycles to access physical memory. What is the average memory access time in cycles?

SHOW YOUR WORK

SID: _____

Question 8: Mr. MOESI (6 pts)

For this question you will be asked to determine which cache coherence scheme(s) can fulfill tasks efficiently based upon assumptions of what task our machine must perform. For each question there will be two parts:

- **Expected Behavior**: the behavior that your scheme must perform efficiently.
- **Necessary Behavior**: the behavior that doesn't need to be performed efficiently but must be supported.

You should select **all** schemes that can handle the expected behavior with maximum efficiency.

For all scenarios, we have the following assumptions:

- The machine has multiple cores
- **Writing** to memory is very very slow (a performance bottleneck)

If this is unclear, consider an example. If the expected behavior involves writing and **MSI** and **MOESI** can do fewer writes to memory than **MESI**, then you would select **MSI** and **MOESI** and not **MESI**.

1. **Expected Behavior:** Processing completely read only data.
   Necessary Behavior: Nothing additional.

   Ⓐ **MSI** Ⓑ **MESI** Ⓒ **MOESI**

2. **Expected Behavior:** A single program with threads for different purposes. A single thread will write in the information about a user. Then after this write, the other three threads will in parallel perform different computations based upon the user's data (each written to different, unique location).
   Necessary Behavior: Writing to a final shared location may be necessary to accumulate results once all threads have completed.

   Ⓐ **MSI** Ⓑ **MESI** Ⓒ **MOESI**

3. **Expected Behavior:** Processing a unique program on each core (with its own memory space) and quick reading from memory shared by programs.
   Necessary Behavior: Writing to data that is shared across programs.

   Ⓐ **MSI** Ⓑ **MESI** Ⓒ **MOESI**