CS61C Fall 2016 Guerilla Session #5:
Caches

Q1 Warm up with Tiny Cache

These two warm up questions are a bit long (and may seem tedious), but will hopefully give you a better visualization of caches and a more straightforward way to tackle cache problems.

I have a special cache called Tiny Cache. It is 128 byte direct mapped cache with cache blocks of size 16 bytes, and a physical memory of 4 KiB (1 KiB = 2^10 bytes). I also have some arrays:

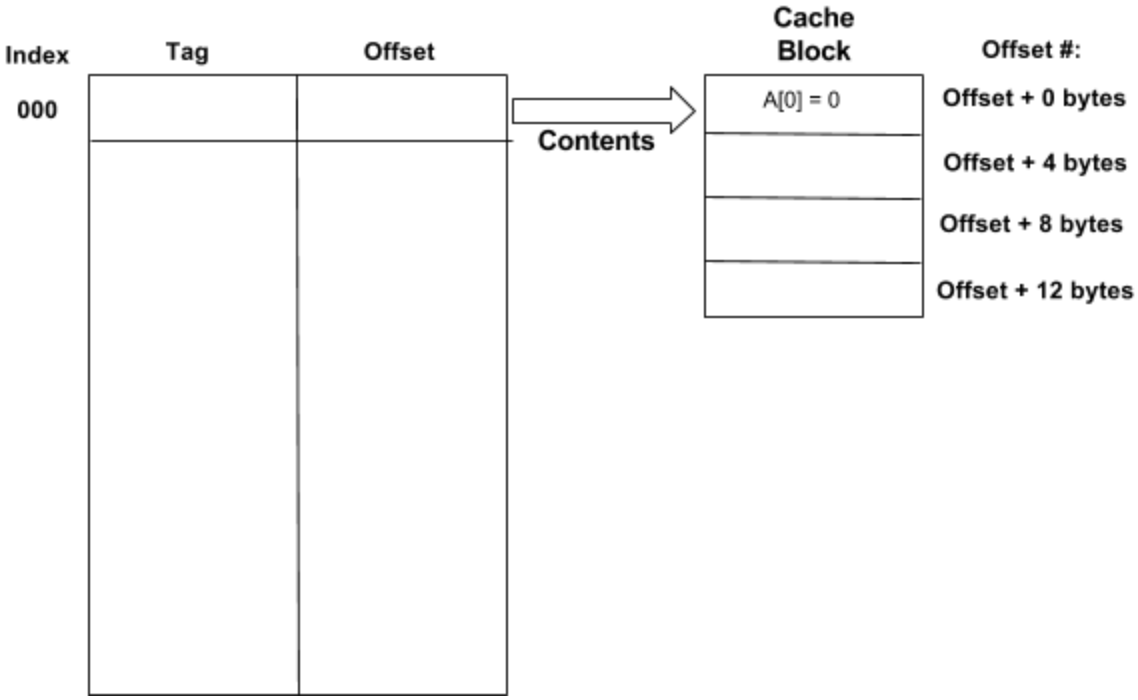
```
int A[4] = {0, 1, 2, 3};
int B[8] = {-1, -2, -3, -4, -5, -6, -7, -8};
int C[4] = {1, 2, 4, 8};
```

A is located at address 0x100, B is located at 0x310, and C is located at 0x200.

1) *Calculate the sizes of everything*

a. What is the number of tag bits, index bits, and offset bits? How many cache blocks are there in Tiny Cache?

b. Extend this drawing of Tiny Cache (you do not need to redraw the individual cache blocks, just add the number of rows/cache blocks that should be in each row and then fill in the index column for each row).



c. How many bytes is an int? How many ints can you store in one cache block?

d. Can the entire contents of A fit inside a single cache block of Tiny Cache? How about the entire contents of B?

2) *Figure out the access pattern of the code chunk*
NOTE: All of these code lines are executed sequentially in the same program (i.e. the cache is not cleared after each line).

a. Let's say I execute: `int foo = A[0];` Fill in the drawing from Q1 by labeling which row will now be filled in Tiny Cache, and then filling in the tag column. Then draw an individual cache block and fill out its contents resulting from this code line. It will help to write out the address of A in binary.

b. Next, I execute the for loop below. For each i, do I find A[i] in the cache? What is the hit rate and the miss rate?

```
for (int i = 0; i < 4; i++)
    int x = A[i] + i;
```

c. Say we changed the body of that for loop to: `A[i] += 4;` How many times is the value A[i] accessed? (i.e. A[i] is read from or written to). Remember that `A[i] += 4` is shorthand for `A[i] = A[i] + 4;`

d. Now, I execute the for loop below. It may be helpful to fill in the drawing from Q1.

```
for (int i = 0; i < 8; i++)
    B[i] += 4
```

   i. How many times is the array B accessed? (i.e. how many times is B[i] read from or written to). This is total number of accesses.

   ii. Is B[i] read from or written to first? (i.e. does the left or right side of the for loop body happen first). This is the first access of B[i] for an iteration of the for loop.

   iii. When i = 0, does the first access of B[i] hit or miss? (i.e. is B[0] in Tiny Cache) How about the second access of B[i]?

   iv. Repeat (ii) for i = 1, 2, 3.

v. When i = 4, does the first access of B[i] hit or miss? How about the second access of B[i]? If Tiny Cache instead had block sizes of 32 bytes would your answers change?

vi. Repeat (iv) for i = 5, 6, 7.

vii. Now you can calculate the total hit rate and miss rate for this for loop. Based on (iii) through (vi), how many accesses hit and how many accesses missed? Divide by the total number of accesses, and what is the total hit rate and miss rate?

e. Next, I execute the line: `int baz = C[0];`

i. Which entry/row of Tiny Cache will the contents of C be entered in? Is that entry/row already occupied? If so, does the occupant have the same or a different tag from C? Draw the contents of the corresponding individual cache block after executing this code line.

ii. I then execute the for loop from part (c) again. Will the hit rate and miss rate be different?

iii. Let's say Tiny Cache is now a 2-way associative cache of 256 B, with the same block size. Would the behavior of Tiny Cache in (i) have been different? How about the hit and miss rates in (ii)? (Hint: how many cache blocks are there per entry/row now that Tiny Cache is 2-way associative?)

SID: _____

Q4: Cache Rules Everything Around Me (15 points) (Fa15 MT2)

You are given a MIPS machine with a single level of **2KiB direct-mapped** cache with **512B cache blocks**. It has **1MiB of physical address space**.

The function foo is ran on the system with a **cold** cache and as the only process:

```
#define ARRAY_LEN 4096
#define STEP_SIZE 64

// A starts at 0x10000
// B starts at 0x20000
foo( int* A, int* B ) {
     int total = 0;
     for ( int i = 0; i < ARRAY_LEN; i += STEP_SIZE ) {
          total += A[ i ];
          total -= B[ i ];
     }
}
```

1. Calculate the number of Tag, Index, and Offset bits for this cache.

2. Calculate the hit percentage for this cache after running `foo`.

3. The cache is now cleared and the code is run again. This time, **`A`** and **`B`** are pointing to the same array, which starts at **`0x10000`**. Calculate the new hit percentage.

4. Assume **`A`** and **`B`** starts once again at **`0x10000`** and **`0x20000`**. What is the new hit percentage if we ran `foo` on a fully associative cache, with all other parameters staying the same?

MT2-4: If you do well, it's clobbering time! (12 points) (Fa15 Final)

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?

SID: _____

Q4: Cache Operations (6 points) (Sp15 MT2)

a) Consider a 32-bit physical memory space and a 32 KiB 2-way associative cache with LRU replacement. You are told the cache uses 5 bits for the offset field. Write in the number of bits in the tag and index fields in the figure below.

b) Assume the same cache as in part a).

```
int ARRAY_SIZE = 64 * 1024;
int arr[ARRAY_SIZE]; // *arr is aligned to a cache block

/* loop 1 */ for (int i = 0; i < ARRAY_SIZE; i += 8) arr[i] = i;
/* loop 2 */ for (int i = ARRAY_SIZE - 8; i >= 0; i -= 8) arr[i+1] = arr[i];
```

1. What is the hit rate of loop 1? What types of misses (of the 3 Cs), if any, occur as a result of loop 1?

2. What is the hit rate of loop 2? What types of misses (of the 3 Cs), if any, occur as a result of loop 2?

Q5: AMAT (4 points)

Suppose you have the following system that consists of an:
L1$ with a local hit rate of 80% and a hit time of 2 cycles
L2$ with a global miss rate of 8% and a hit time of 15 cycles
DRAM accesses take 50 cycles

i. What is the AMAT of the L1 cache? _____

ii. Suppose we want to improve our AMAT, making sure that it is no greater than 6 cycles, by improving our L2$'s hit rate. What is the minimum possible local hit rate for L2$ that allows us to meet our AMAT requirement?