Computer Architecture and Engineering

CS152 Quiz #2

March 3rd, 2009

Professor Krste Asanovic

Name:_____

This is a closed book, closed notes exam.

80 Minutes

10 Pages

Notes:

- **Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.**
- **Please carefully state any assumptions you make.**
- **Please write your name on every page in the quiz.**
- **You must not discuss a quiz's contents with students who have not yet taken the quiz. If you have inadvertently been exposed to the quiz prior to taking it, you must tell the instructor or TA.**
- **You will get no credit for selecting multiple choice answers without giving explanations if the instructions ask you to explain your choice.**

Writing name on each sheet	**_____**	**1 Point**
Question 1	**_____**	**30 Points**
Question 2	**_____**	**26 Points**
Question 3	**_____**	**23 Points**
TOTAL	**_____**	**80 Points**

NAME:_____

Problem Q.2.1: Way-Predicting Cache Evaluation [30 Points]

To improve the hit rate for our data cache, we made it 2-way set associative (it was formerly direct mapped). Sadly as a consequence the hit time has gone up, and we are going to use way-prediction to improve it. Each cache set will have a way prediction indicating which way is likely to be accessed.

When doing a cache access, the prediction is used to route the data. If it is incorrect, there will be a delay as the correct way is used. If the desired data is not resident in the cache, it is like a normal cache miss. After a cache miss, the prediction is not used since the correct block is already known. Figure Q2.1-A summarizes this process.

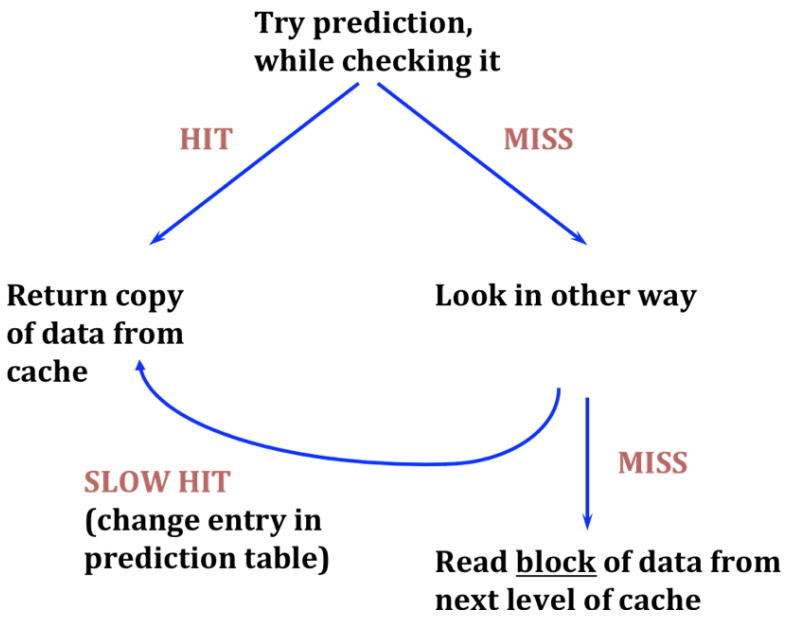


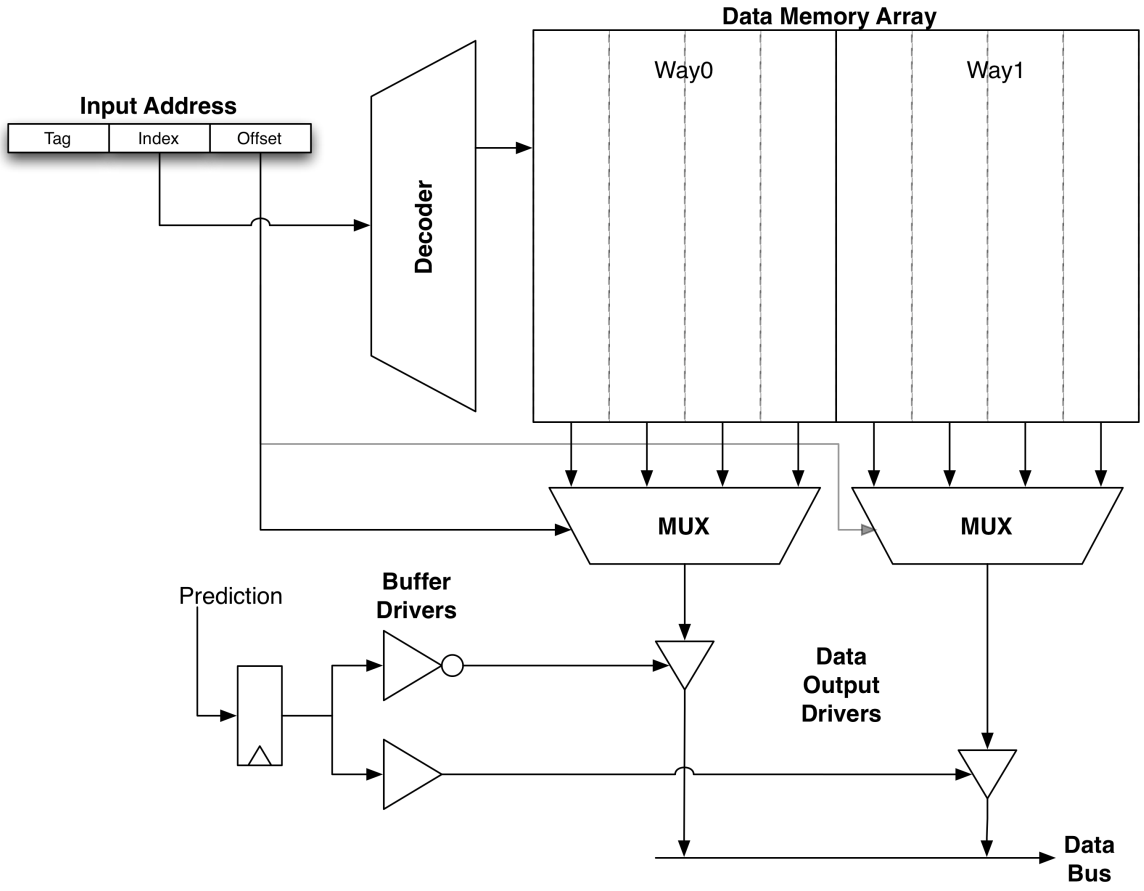
Figure Q2.1-A: Way-prediction FSM

Since there are two ways, only one bit will be used per prediction, and its value will directly correspond to the way. How the predictions are generated or maintained are beyond the scope of this problem. You can assume that at the beginning of a cycle, the selected prediction is available, and determining the prediction is not on the critical path.

NAME:_____

Problem Q2.1.A **Baseline Cache Design**
10 Points

The diagram below shows the data portion of our cache.



Our cache has **16 byte** lines, is **2-way** set associative, and has a total capacity of **4kB**.

Please complete Table Q2.1-1 on the next page with delays across each element of the cache. Using the data you compute in Table Q2.1-1, calculate the critical path delay through this cache (from when the Input Address is set to when the correct data is on the Data Bus).

NAME:_____

Component	Delay equation (ns)	Total (ns)
Decoder	0.2×(# of index bits) + 1	2.4
Memory Array	0.2×$\log_2$(# of rows) + 0.2×$\log_2$(# of bits in a row) + 1	4
N-to-1 MUX	0.5×$\log_2$ N + 1	2
Buffer driver	2	2
Data output driver	0.5×(associativity) + 1	2
Critical Path Delay		10.4

Table Q2.1-1

You may assume that the prediction register is correctly loaded at the start of the cycle, and the clk-to-q delay is 100ps. The inverting and non-inverting buffer drivers both have the same delay. You only need to worry about the case of a fast hit (cache hit with correct prediction).

Show your work:

Blocks = (Capacity) / (Block size) = 4KB / 16B = $2^{12-4} = 2^8 = 256$ blocks

Sets = (# Blocks) / (Associativity) = 256/2 = 128 sets = 2^7 => **7 index bits**

of rows = # Sets = 256 => **$\log_2$(# of rows) = 7**

# of bits in a row = (# of blocks per row)(# of bytes per block)(# of bits per byte)

= (2)(16)(8) = $2^{1+4+3} = 2^8$ => **$\log_2$(# of bits in a row) = 8**

Note that this is only the data part of the cache (tags not part of this problem)

N = 4 (shown in diagram, but 16B holds 4 words)

Critical Path = Decoder + Memory Array + N-to-1 MUX + Data output driver

Note that the prediction and buffer drivers were not on the critical path

Scoring

- Decoder & Memory Array: 2 pts each
- N-to-1 MUX, Buffer driver, Data output driver: 1 pt each
- Critical Path Delay: 3 pts (2pts if correct path but wrong total)

NAME:_____

Problem Q2.1.B **Way-Prediction Behavior**
10 Points

Now we will study the impact of way prediction on cache hit rate. For this problem, the cache is a **128 byte**, **2-way** set associative cache with **16 bytes** per cache line. The cache is byte addressable and uses a least recently used (LRU) replacement policy.

Please complete Table Q2.1-2 on the next page showing a trace of memory accesses. In the table, each entry contains the {tag,index} contents of that line, or "-", if no data is present. You should only fill in elements in the table when a value changes. For simplicity, the addresses are only 8 bits.

The first 3 lines of the table have been filled in for you. The initial values marked with a '*' are the least recently used ways in that set. For your convenience, the address breakdown for access to the main cache is depicted below.

7	6	5	4	3	2	1	0
TAG		INDEX		WORD SELECT		BYTE SELECT	

Problem Q2.1.C **Average Memory Access Time**
10 Points

Assume the cache has a **90%** hit rate, and that the way-predictor is right **75%** of the time there is a cache hit. A cache hit with a correct way-prediction will take **2 cycles**, a cache hit with an incorrect prediction will take **4 cycles**, and a cache miss (way-prediction irrelevant) will take **60 cycles**. Compute the average memory access time of the cache with way-prediction.

Without way-prediction (the original 2-way set associative cache) has the same hit rate and miss time, but a **3 cycle** hit time. By how many cycles does way-prediction improve the average memory access time?

Its important to notice that 60 cycles is the miss time, not the miss penalty (miss time = miss penalty + hit time).

AMAT (with WP) = (Hit %)((% WP correct)(WP correct time) + (% WP incorrect time)(WP incorrect time)) + (1 – Hit %)(Miss time) = (0.9)((0.75)2 + (0.25)4) + (0.1)60 = (0.9)(1.5 + 1) + 6 = (0.9)(2.5) + 6 = 8.25 (worth 6 pts)

AMAT (without WP) = (Hit %)(Hit time)+ (1 – Hit %)(Miss time) = (0.9)(3) + (0.1)(60) = 2.7 + 6 = 8.7 (worth 4 points with improvement)

Improvement: 8.7 – 8.25 = 0.45 (you can have fractional cycles since this is AMAT)

NAME:_____

Read Address	Set0		Set1		Set2		Set3		Cache Hit?	Way-Prediction	
	Way0	Way1	Way0	Way1	Way0	Way1	Way0	Way1		Prediction	Correct?
	0	-*	9	-*	A*	6	7	B*			
04									Y	0	Y
68									Y	1	Y
2C					2				N	1	-
B4									Y	0	N
54				5					N	0	-
3C							3		N	0	-
94									Y	1	N
28									Y	0	Y
64									Y	0	N
80		8							N	0	-
64									Y	1	Y
B4									Y	1	Y

Table Q2.1-2

1pt per line (1 pt is free if not blank)

NAME:_____

Problem Q2.2: Cache Code Co-Design 26 Points

Problem Q2.2.A **Cache from SW**
8 Points

Examine the code given below to compute the average of an array:

```
total = 0;
for(j=0; j < k; j++) {
   sub_total = 0;
   /* Nested loops to avoid overflow */
   for(i=0; i < N; i++) {
      sub_total += A[j*N + i];
   }
   total += sub_total/N;
}
average = total/k;
```

When designing a cache to run this application, given a constant cache capacity and associativity, will you want a larger or smaller block size? Why?

Examining the code its good to see that the program accesses consecutive addresses and never reuses any of them. Because of this you will want to leverage its spatial locality by using a larger blocker size to reduce compulsory misses

Problem Q2.2.B **Adversarial Cache Patterns**
8 Points

Given two fully associative caches that differ only in replacement policy (FIFO or LRU), can you come up with a memory access stream where the cache with FIFO replacement will have less misses than the cache with LRU replacement? To make it easier to describe, assume the caches have 4 entries and the line size is only 1 unit of the address space.

Any pattern where FIFO does better than LRU at some point. An simple sequence:

	0	1	2	3	0	4	1
FIFO	Miss	Miss	Miss	Miss	Hit	Miss	Hit
LRU	Miss	Miss	Miss	Miss	Hit	Miss	**Miss**

NAME:_____

Problem Q2.2.C **Cache Tradeoffs**
10 Points

Examine the code given below (note it is slightly different than Q2.2.A):

```
total = 0;
for(i=0; i < N; i++) {
   sub_total = 0;
   /* Nested loops to avoid overflow */
   for(j=0; j < k; j++) {
      sub_total += A[j*N + i];
   }
   total += sub_total/k;
}
average = total/N;
```

Generally, how will the size of the array and the cache capacity impact the choice of line size for good performance? Why?

This problem is asking how to pick a line size given a cache capacity and array size. Changing the size of the array or the cache were not options.

Basically you want to be able to fit at least one iteration of the outer loop in the cache with a block size ≥2 words. To do this, you want at least two "columns" of A to be able to fit in the cache at a time, so the you will get some hits from spatial locality. Without this, the program will suffer 100% misses. If the block size is two and en entire column can fit, one iteration of the outer loop result in all misses, but the next iteration will be all hits.

As the size of the array gets smaller relative to the cache capacity, the "columns" are "shorter," so you will want a larger line size to reduce compulsory misses.

As the size of the array gets larger relative to the cache capacity, the "columns" will get "taller." You will need to make the line size smaller to get more lines in the cache to reduce conflict misses so it can still hold an entire iteration of the outer loop.

Problem Q2.3: Three C's of Cache Misses (Short Answer) [23 points]

Mark whether the following modifications will cause each of the categories to **increase, decrease**, or whether the modification will have **no effect**. You can assume the baseline cache is set associative. **Explain your reasoning** to receive credit. This table continues on the next page.

	Compulsory Misses	Conflict Misses	Capacity Misses
Double the associativity (capacity and line size constant) (halves # of sets)	No effect If the data wasn't ever in the cache, increasing associativity with the constraints won't change that.	Decrease Typically higher associativity reduces conflict misses because there are more places to put the same element.	No effect Capacity was given as a constant.
Halving the line size (associativity and # sets constant) (halves capacity)	Increase Shorter lines mean less "prefetching" for shorter lines. It reduces the cache's ability to exploit spatial locality.	No effect Same # of sets and associativity.	Increase Capacity has been cut in half.
Doubling the number of sets (capacity and line size constant) (halves associativity)	No effect If the data wasn't ever in the cache, increasing the number of sets with the constraints won't change that.	Increase Less associativity.	No effect. Capacity is still constant

	Compulsory Misses	Conflict Misses	Capacity Misses
Adding a victim cache	No effect Victim cache only holds lines previously held by CPU.	Decrease The victim cache holds the victim of a conflict, so it can be used again later.	Decrease Slightly larger cache capacity. *No effect, since victim cache doesn't count towards capacity total (only -0.5 off)*
Adding prefetching	Decrease Hopefully prefetched data is there when needed.	No effect – doesn't affect placement - or - Possibly increase – prefetch data could possibly pollute cache	No effect – doesn't affect change - or - Possibly increase – prefetch data could possibly pollute cache

Each box is worth 1.5pts, 0.5 pts were given for free.

See slide 12 of lecture 7 for reference on miss types.

END OF QUIZ