CPSC 313: Computer Hardware and Operating Systems
Assignment #4, due Thursday, Nov 1st, 2012 at 23:59

This assignment is all about caching. It consists of two theoretical questions, along with a question that will ask you to complete the implementation of a cache in C. The assignment is to be done alone. The late penalty is 33% per day. No late assignments accepted after 3 days.

[16] 1. Suppose we have a system with the following properties:

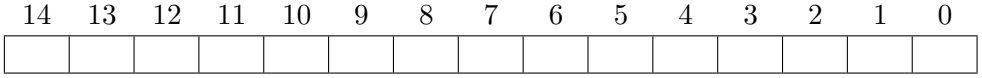
- The memory is byte addressable,
- Memory accesses are to `1-byte words` (not to 4-byte words).
- Addresses are 15-bits.
- The cache is four-way set associative ($E = 4$) with a 8-byte block size ($B = 8$) and eight sets ($S = 8$).

The contents of the cache are as follows, with all addresses, tags and values given in hexadecimal.

Set	Valid	Tag	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0	0	032	43	F6	A8	88	5A	30	8D	31
	1	131	98	A2	E0	37	07	34	4A	40
	1	193	82	22	99	F3	1D	00	82	EF
	1	1A9	8E	C4	E6	C8	94	52	82	1E
1	0	193	82	22	99	F3	1D	00	82	EF
	0	04E	90	C6	CC	0A	C2	9B	7C	97
	1	063	8D	01	37	7B	E5	46	6C	F3
	0	191	79	21	6D	5D	98	97	9F	B1
2	1	1C5	0D	D3	F8	4D	5B	5B	54	70
	1	0FF	D7	2D	BD	01	AD	FB	7B	8E
	1	1BD	13	10	BA	69	8D	FB	5A	C2
	0	004	5F	12	C7	F9	92	4A	19	94
3	1	01A	FE	D6	A2	67	E9	6B	A7	C9
	1	058	EF	C1	66	36	92	0D	87	15
	1	17B	39	16	CF	70	80	1F	2E	28
	1	0D7	E0	D9	57	48	F7	28	EB	65
4	1	087	18	BC	D5	88	21	54	AE	E7
	1	1B5	4A	41	DC	25	A5	9B	59	C3
	0	00D	53	92	AF	26	01	3C	5D	1B
	1	002	32	86	08	5F	0C	A4	17	91
5	1	18B	8D	B3	8E	F8	E7	9D	CB	06
	1	103	A1	80	E6	C9	E0	E8	BB	01
	1	1E8	A3	ED	71	57	7C	1B	D3	14
	1	1B2	77	8A	F2	FD	A5	56	05	C6
6	1	10E	65	52	5F	3A	A5	5A	B9	45
	0	074	89	86	26	3E	81	44	05	5C
	0	0A3	96	A2	AA	B1	0B	6B	4C	C5
	1	1C3	41	14	1E	8C	EA	15	48	6A
7	0	1F7	C7	2E	99	3B	3E	E1	41	16
	1	036	FB	C2	A2	BA	9C	55	D7	41
	1	083	1F	6C	E5	C3	E1	69	B8	79
	1	031	EA	FD	6B	A3	36	C2	4C	F5

[2] (a) The following diagram shows the format of an address (one bit per box). Indicate the fields that would be used to determine the following:

- CO: the cache block offset.
- CI: the cache set index.
- CT: the cache tag.



[14] (b) For each of the following memory accesses, indicate the cache set that will be searched, if it will be a cache hit or miss **when carried out in sequence** as listed, and the value of a read if it can be inferred from the information in the cache. Justify your answers.

Operation	Address	Cache Set	Tag	Hit?	Value (or unknown)
Read	`0x0362`				
Read	`0x7BFE`				
Read	`0x3FC9`				
Read	`0x0361`				

[16] 2. One way to determine what a cache's miss rate will be in practice is to run through a number of programs using a simulated cache, and take note of the number of cache hits and misses as a function of the total number of memory accesses. In this question, you will complete a C implementation of a simulated cache. Five files are provided in the a4.zip file:

- `cache.h`: a header file with some declarations.
- `cache.c`: the incomplete implementation of the cache.
- `cache-test.c`: a test program to help you debug your implementation.
- `cache-ref.o`: The object code from a reference implementation of the cache.c file. To use this file copy (don't rename it) it to cache.c and type make. The resulting cache-test program will be built and will run with the reference cache. Be warned that if you make changes to cache.c or cache.h when you type make it will overwrite cache.o, hence the reason for copying as opposed to renaming.
- `Makefile`: typing `make` at the command prompt will recompile any of the .c files that have changed.

File `cache.c` contains the comment `/* TO BE COMPLETED BY THE STUDENT */` whenever you need to supply code. The first two functions where this occur can be completed using a single line of code. The next three will need between 10 and 20 lines of code each (the exact number may of course vary depending on whether or not you place curly braces on separate lines, how you write the code, etc).

Complete the implementation of the cache in `cache.c`, and then run program `cache-test` to verify that it works correctly. Here is the expected output:

```
Sum = 4326400
Miss rate =   0.2500
Sum = 4326400
Miss rate =   1.0000
Sum = 4326400
Miss rate =   0.5000
```

[18] 3. This problem tests your ability to predict the cache behaviour of C code. For each of the following caches, array sizes and functions, first determine the miss rate using your cache implementation from the previous question (or the reference solution contained in the file `cache.o`), and then explain how you would have derived the miss rate given only the program code and the characteristics of the cache (that is, using pencil and paper only). You may need to make modifications to cache-test.c to get the information needed. Assume that we execute the code under the following conditions:

- `sizeof(int)` = 4
- The cache contains 128, sixteen-byte blocks.

- The arrays are stored in row-major order.
- The only memory accesses are to the entries of the array `a`.

In order to simplify your explanations, you can write them assuming that $a[0][0]$ ends up in cache set 0, even if this isn't actually the case when you run the program. Note that specific cache set numbers may in fact vary depending on things like the compiler or machine used. However the miss rate will remain the same.

Note: if you set the cache policies to `CACHE_TRACEPOLICY`, then the reference solution will output additional information about cache hits, misses, and which set was used in each case. You may find such information helpful. The scenarios you are to predict the cache behaviour for are:

[3] a. An array with 64 rows and 64 columns, a direct-mapped cache, and the function `sumA` from the program `cache-test` provided with question 2.

[3] b. An array with 64 rows and 64 columns, a direct-mapped cache, and the function `sumB` from the program `cache-test` provided with question 2.

[3] c. An array with 64 rows and 64 columns, a direct-mapped cache, and the function `sumC` from the program `cache-test` provided with question 2.

[3] d. An array with 68 rows and 68 columns, a direct-mapped cache, and the function `sumB` from the program `cache-test` provided with question 2.

[3] e. An array with 48 rows and 48 columns, a direct-mapped cache, and the function `sumB` from the program `cache-test` provided with question 2.

[3] f. An array with 48 rows and 48 columns, a two-way set-associative cache, and the function `sumB` from the program `cache-test` provided with question 2.

Deliverables

You should use the `handin` program to submit your assignment. The assignment name is `a4`, and the files to submit are:

1. A text file containing the following information:
   - Your name and student number.
   - Your answers questions 1 and 3
   - How long it took you to complete the assignment (not including any time you may have spent revising before starting to work on it).
2. The completed `cache.c` file.