Final Review – Last Lecture!

EECS 370 – Introduction to Computer Organization
Fall 2013

Profs. Valeria Bertacco, Robert Dick, & Satish Narayanasamy

EECS Department
University of Michigan in Ann Arbor, USA

Final exam

- 10:30-12:30, 16 Dec.
- Have an empty seat between you and the next person on each side
- Calculator without network connection O.K.
- Calculator with network connection, forbidden.
- Can bring two letter-sized pages of notes, front and back.
- Students with special needs or conflicts should have already received an email giving their exam room. Email i370@eecs.umich.edu if not.

Room assignments

Room	Last Initial	Planned # students
STAMPS (450 seats)	K-W	186
DOW1010 (69 seats)	H	25
DOW1013 (164 seats)	A-B, D-G, I-J	68
DOW1014 (70 seats)	C & X	28
DOW1017 (76 seats)	Y-Z	31

Seeking help

- Office hours continue through 13 December.
- 1:30-2:00 professor office hours canceled on Thursday,
  - but 4:00 office hours still held.
- If students still have questions at 5:00, I will stick around later.
- Prof. Narayanasamy will hold extra office hours 12:00-2:00 12 December.
- Two more review sessions
  - 4:00-6:00 12 Dec. in Chrysler 220
  - 1:00-3:00 15 Dec. in Chrysler 220

Exam topics

- Covers all course material
  - Instruction sets
  - Addressing modes / memory layout for data types and structures
  - Compiling and linking, caller/callee, stack/heap/global/text
  - Floating point
  - Single cycle datapath
  - Multicycle datapath
  - Pipelining and hazards
  - Caches and virtual memory
  - Performance
  - I/O

- Material presented in lecture slides, book, projects

Exam strategy

- Read all questions before starting.
- Remember to show your work – turn in extra sheets you use to reason about the problem.
- **Do not spend all your time on a single question.** If you don't understand a question, move ahead and come back to it later.
- Give yourself a few minutes at the end to check your work and/or attempt to solve easy questions that you have not attacked yet

Caches (W06-Q3a)

You are brought in as a consultant to analyze the cache performance of a 370Systems, Inc. computer. This is a byte-addressable computer with 16-bit addresses. The system uses separate instruction and data caches, and you are asked to only focus on the data cache performance. The cache is write-allocate and write-back. It is two-way set-associative with an LRU replacement policy. The cache size is 16 KB and the block size is 32 bytes. Which is the correct partition of address bits into tag, set index, and block offset?

tag: 3 bits
set index: 8 bits
block offset: 5 bits

Caches (W06-Q3c)

How much memory is required for bookkeeping information for this cache (everything but the actual data; tags, valid and dirty bits, and LRU information)

- How many bookkeeping bits per set

  Each set: 2 dirty + 2 valid + 6 tag + 1 LRU = 11 bits

- How many bookkeeping bits total

  11*256 = 2816 bits

Virtual memory (F07-Q9)

Virtual memory has been added to an LC2K system. Basic properties of the LC2 ISA have not been changed; the machine is still word addressable with a 16 bit virtual address. The page table, TLB, and cache are shown below. All numeric values are shown in hex. The page size is 8192 words. The cache is a unified instruction/data cache that is write allocate and write back.

Consider this LC2K instruction: sw 0 0 15 # 15 is decimal
This instruction is fetched and executed. For the instruction fetch, the program counter contains 0x4004. Register 0 contains 0. The machine language code for this sw instruction is 0x00c0000f. Before this instruction executes, virtual memory locations 0 through 0x10 all contain 0x01c00000, the machine code for a noop Instruction.

Virtual memory (F07-Q9)

Page table			
VPN	PPN	Valid	Dirty
0	3	1	0
1	0	0	0
2	1	1	0
3	1	0	0
4	2	0	0
5	3	0	0
6	2	1	0
7	0	1	0

4 entry fully associative TLB			
Tag	PPN	Valid	Dirty
7	0	1	0
0	3	1	0
~~5~~ 2	~~3~~ 1	~~0~~ 1	0
6	2	1	0

*2-way set associative **physically addressed** cache with 2 word block size*										
index	tag	valid	dirty	word0	word1	tag	valid	dirty	word0	word1
0	0bc	1	0	00c0000f	00c0000f	312	1	0	00c0000f	00c0000f
1	1ef	0	0	00c0000f	00c0000f	1cc	1	0	00c0000f	00c0000f
2	200	1	0	00c0000f	00c0000f	400	1	0	00c0000f	00c0000f
3	3a3	1	1	00c0000f	00c0000f	1dd	1	0	00c0000f	00c0000f
4	400	1	0	00c0000f	00c0000f	200	1	0	00c0000f	00c0000f
5	5ed	0	0	00c0000f	00c0000f	721	1	0	00c0000f	00c0000f
6	6cb	1	0	00c0000f	00c0000f	221	1	1	00c0000f	00c0000f
7	000	1	0	00c0000f	00c0000f	~~654~~	~~0~~	~~0~~	~~00c0000f~~	~~00c0000f~~
						600	1	1	01c00000	00000000

Virtual memory (F07-Q9ab)

a) Circle in the tables each value that is accessed for the fetch and execute of the above sw instruction. If this instruction modifies any of the values in any of the tables (page table, TLB, or cache), cross the old values out and write in the new values next to them.

Fetch: VA: 0x4004, VP#: 2, PA: 0x2004
Store: VA: 0x0F, VP#: 0, PA: 0x600F

b) Circle either HIT or MISS to describe what happens in the cache and the TLB for the instruction fetch and the data access of the given sw instruction:
Instruction fetch: cache: HIT MISS TLB: HIT MISS
Data access: cache: HIT MISS TLB: HIT MISS

Virtual memory (F07-Q9cd)

c) For the given sw instruction, list all the actual physical memory read and write operations that must be performed (do not count accesses that only go to the page table; only count those that require going to the physical memory to read or write data). For each access, give the starting address, whether it is a read or write, and the number of words transferred.

READ 0x600E, 2 words

e) Cache and TLB lookups can be done at the same time if the set index can be extracted entirely from the page offset. If this implementation is modified to use a larger cache, what is the largest cache that could be used that would still allow these parallel lookups? Do not change the block size or associativity. State your answer in words of memory (NOT in bytes).

Current set index is 3bits, block offset is 1 bit. To fit in the page offset, we can expand the set index up to 12bits. That would lead to: 2^12 sets * 4 words/set = 2^14 words

Virtual memory (F07-Q9f)

f) It is determined that acceptable performance can only be achieved with a cache that is twice as large as your answer in part (e). To accomplish this and still allow cache and TLB lookups to happen in parallel, what change to you need to make? (circle the one best answer below):

(i) Double the block size
(ii) Halve the block size
(iii) Quadruple the block size
(iv) Double the associativity
(v) Halve the associativity
(vi) Halve the associativity and double the block size
(vii) Halve the associativity and halve the block size
(viii) None of the above changes are sufficient

Caller/callee (W07-Q3)

Consider the following function.

```
int foo(void) {
   int h, i, j, k;
   j = 100;
   bar1();
   k = j * 3;
   i = 0;
   while (i<10){
      … = k;
      h = i*3;

      bar2();
      k=…
      … = h;
      i += 1;
   }
}
```

Caller/callee (W07-Q3)

a) Assume that you have 2 caller-saved and 2 callee-saved registers. Pick the best register assignment for h, i, j, k that can lead to the minimal number of executed save and restore instructions. Assume that each variable requires its own register

Caller-saved: j, k Callee-saved: h,i

b) Now, assuming foo() is called 20 times, how many caller save/restore instruction pairs will be executed in foo()?

20 for caller registers across bar1().

There would also be 40 callee saves.

Storage

- Consider a hard disk with the following characteristics:
  - Average seek time: 12 ms.
  - Rotation rate: 7,200 RPM.
  - Transfer time: bounded only by rotation rate.
  - Controller overhead: 1 ms.
  - Sectors per track: 8.
- Neglect wait time and caching by the hard disk. The controller overhead is imposed for each access.
- On average, how much will adding a second hard disk containing an exact copy of the first decrease total block read access time? The orientations of the two disks are not synchronized, i.e., all relative orientations for the two disks are equally probable.

Drops average rotational delay from 1/2 to 1/3 rotation.
Seek + rot/3 + rot/8 + 1 ms.