Final Exam Review

EECS 370 – Introduction to Computer Organization - Winter 2016

Profs. Valeria Bertacco & Reetu Das

EECS Department
University of Michigan in Ann Arbor, USA

© Bertacco-Das, 2016
The material in this presentation cannot be
copied in any form without our written permission

Exam: Time

- **Wednesday April 20, 2016 – 10:30am-12:30pm**

- No Michigan time, exam starts at 10:30am sharp
  - Show a few minutes early to your assigned room

- Closed book/notes
  - No phones, computers, calculators, smart watches, nothing electronic
  - You are only allowed
    a) #2 pencils,
    b) eraser and
    c) one letter-size two-sided sheet of paper with any notes you wish to have
- **Bring your Mcard to the exam**

Exam: Location

- Check your email for the location
  You should have received an email from either myself or Prof. Winsor with your exam location

- Go to your assigned room!
  Exams taken in the wrong room show up as missing from the correct room's roster.

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, Data and control hazards
  - Caches and virtual memory
  - Performance

- Material presented in lecture slides, book, projects

The good news

- Questions you may encounter:
  - Short answers on concepts from the class
  - Caller/callee register assignment
  - Data layout and alignment
  - Cache access stream simulation
  - Show program execution on a pipeline
  - Modify datapaths to support a new instruction, or feature
  - Performance of everything
  - Virtual memory simulation
  - Hierarchical Page Tables

Preparing for the exam

- **Review sessions**
  - Today's lecture - Profs
  - **Monday 4/18 6:00pm – 8:00pm – Chrysler 220 – GSIs/Ias**

- **Office Hours**
  - Professors OH's
  - GSI/IAs OH's

- **No Discussions this week**

Exam strategy

- Expect 8-13 questions
- Pace yourself
- 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
- Most importantly, **show your work** !!!

What's the best way to review for the exam?

Practice,
Practice,
Practice

Plenty of choices: Old Exams, Class Problems, Midterm Review Problems

Select Practice Questions

Just because it is not covered in this review it does not mean that it is not important!

Select Practice Questions

Problem 1

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~~ 600	~~0~~ 1	~~0~~ 1	~~00c0000f~~ 01c00000	~~00c0000f~~ 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

Select Practice Questions

Problem 2

You need to design a 3-level hierarchical page table. Physical memory is with 8KB (byte-addressable); virtual memory addresses are 17-bits long. A page is 32 bytes and each page table entry is 2 bytes. The superpage table and each of the 2nd level page tables require precisely one page of storage.

Determine the bit positions for the following fields in a virtual address and a physical address. Page offset field has been completed as an example.

Virtual Address			
Super-Page Table	2nd Level Page Table	3rd Level Page Table	Page Offset
Bits:	Bits:	Bits:	Bits:

Physical Address	
Physical Page Number	Page Offset
Bits:	Bits:

You need to design a 3-level hierarchical page table. Physical memory is with 8KB (byte-addressable); virtual memory addresses are 17-bits long. A page is 32 bytes and each page table entry is 2 bytes. The superpage table and each of the 2nd level page tables require precisely one page of storage.

Determine the bit positions for the following fields in a virtual address and a physical address. Page offset field has been completed as an example.

Virtual Address			
Super-Page Table	2nd Level Page Table	3rd Level Page Table	Page Offset
Bits: 16-13	Bits: 12-9	Bits: 8-5	Bits: 4-0

Physical Address	
Physical Page Number	Page Offset
Bits:12-5	Bits:4-0

Assume that at the beginning of a program's execution only the superpage table is in main memory. For the program shown below, calculate the total size of all page tables (including the super page table, 2nd level and 3rd level page tables) in main memory after executing the for loop. Assume the array BigArray is allocated at the virtual address 0x0 and 'char' = 1 byte. Show your work.

```
char BigArray[2048*32];
for (i = 0; i < 2048; i += 16){
     sum += BigArray[i*32];
}
```

Total 2nd Level Page Tables?

Total 3rd Level Page Tables?

Total Size in memory?

```
char BigArray[2048*32];
for (i = 0; i < 2048; i += 16){
     sum += BigArray[i*32];
}
```

Total 2nd Level Page Tables?

Total 3rd Level Page Tables?

Total Size in memory?

```
char BigArray[2048*32];
for (i = 0; i < 2048; i += 16){
     sum += BigArray[i*32];
}
```

Total 3rd Level Page Tables? 128

Total 2nd Level Page Tables? 8

Total Size in memory? (1+8+128)*32 = 4384

Select Practice Questions

Problem 3

Caller/callee (W07-Q3)

Consider the following function.

```
int foo() {
    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 (these are only for caller registers)