EECS 470 *Final Exam*

Winter 2015

Name: ______________________________ unique name: ____________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

____

Scores:

Page #	Points
2	/8
3	/18
4	/13
5	/15
6	/12
7	/12
8	/7
9&10	/15
Total	**/100**

NOTES:

- Open White book and EABI only.
- There are **10** pages including this one.
- Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
- Don't spend too much time on any one problem.
- You have about 120 minutes for the exam.
- **Be sure to show work and explain what you've done when asked to do so.** Getting partial credit without showing work will be rare.

1) Multiple choice/fill-in-the-blank. Pick the best answer.
[8 points, -2 per wrong/blank answer, min 0]

a) In a hash cache and skew cache we use some hashing function to determine the index to use when access the cache. Those accesses typically use an XOR. Why don't we generally use some other gate such as a NAND?
   - ***Because the XOR gate is typically faster than a NAND gate.***
   - ***Because a NAND gate would make some sets more likely to be accessed than others.***
   - ***Because the XOR gate typically uses less power than a NAND gate.***
   - ***It's just what people typically use—either XOR or NAND should work equally well.***

b) Say you have a single application on an embedded system that runs every second for 0.9 seconds and is otherwise idle. Assuming it's an option, to best reduce power you should
   - ***Turn the processor off (or nearly off) for the 0.1s when idle***
   - ***Scale the voltage and frequency down to about 0.9 times what it is.***
   - ***Scale the voltage and frequency down to about 0.729 times what it is.***
   - ***Scale the current down to about 0.729 times what it is.***

c) You would expect a direct-mapped 128-byte cache with 32-byte cache lines to get a hit about

   _________% of the time on a memory access with a stack distance of 2.

d) The ***BTB / gshare / RAS / tournament*** predictor has problems correctly predicting the address of

   load / store / call / return instructions.

2) *Caches and virtual memory* **[18 points, 6 points each]**

a) Consider the following C-code segment:

```
int X[4096];            // each element is 4 bytes in size
for(i=0;i<10000000;i++)
     for(j=0;j<A;j=j+B)
          X[j]++;
```

Assume that all values other than the data in array "X" are kept in registers during this loop. For this code, what would be the expected _hit-rate_ for the various values of A and B if the data cache were 4 KB with 32-byte lines and was two-way associative?
[-2 per wrong/blank box, min 0]

	B=2	B=6
A=2048		
A=1024		

b) Consider a memory system with 32 KB pages where the virtual memory size is 4 GB and physical memory addresses are 40 bits. If you have a 32-entry fully-associative TLB, how many bits are:

- Used for the entire TLB tag store (just the tags, not status bits)? **[3]**

- Used for the entire TLB "data" store? **[3]**

c) Say you are working on a processor that has a 256 KB write-back data cache with 32-byte cache lines that is virtually addressed. In order to avoid problems associated with a virtually addressed cache, you have proposed invalidating the cache on every context switch. Clearly that will make the next process run slower for a while as the cache warms up. But your boss is worried it might make even very short high-priority context switches (a dozen instructions say which need to be done quickly) be very slow in the worst case (more than just due to cache misses!). Why could that happen?

3) Power **[7 points]**

a) If we had a processor that used 200W and ran at 2GIPS about what performance and power consumption would you expect if we moved the voltage and frequency up by 4%? **[4, 2 each]**

Power: _________________W

Performance: _________________ GIPS

b) Consider an ALU and a cache both of which draw 2 Watts of power. If you scale the frequency and voltage up by 5%, which of the two devices would be likely to draw more power? Explain your answer. **[3]**

4) An academic paper entitled "Exploring Efficient SMT Branch Predictor Design" concluded that:

"… branch prediction accuracy is less important in an SMT system than in a traditional superscalar processor. Misprediction ratios that increased by factors of two or three only caused slowdowns of as little as 2%."

If a single-thread running on the same processor would see a slowdown of 30%+ given the same misprediction rates on the same benchmarks, what could explain why the multi-threaded applications were largely unaffected by the very poor misprediction rates? **[6 points]**

5) You work at "Cool Computers Inc" and have been tasked with evaluating replacements for your company's current 4 KB direct-mapped VIPT L1 data cache. The virtual memory system you use has 32-bit physical and virtual addresses and uses a 4 KB page size. Assuming you must stay with the same virtual memory system and continue to use a VIPT cache, label each of the following options as being "viable" or "non-viable". *For those you label as non-viable, provide a brief explanation as to why it isn't viable.* **[8 points, 2 each]**

   a) 4-way associative 16KB cache.

   b) 2-way associative 4KB cache.

   c) Direct-mapped 4KB hash cache.

   d) Direct-mapped 8KB cache.

6) You've been working on a high-end embedded-systems design (1GHz processor, 1GB memory). During development the data cache has been kept turned off (to ease debugging). When the cache is turned on, you find that performance *drops* by a factor of 3. You know that the cache works correctly and greatly improves performance of other applications running on the same system. You know that you've not made any fundamental error. Why can adding a data cache slow down an application by this much? Your solution must be reasonable given the scenario provided. **[7 points]**

7) Write a Verilog module to arbitrate the output of three functional units to two CDB buses. If a functional unit requests a CDB but none is available then, and only then, you should stall it. Your code must be synthesizeable. You may pick any prioritization that you wish but if two or more functional units want to use the CDB, at least 2 must be able to. **[12 points]**

```
module cdb_arb (
  input logic  [2:0]        fu_request_i, //Functional unit x wants CDB
  input logic  [2:0] [4:0]  fu_prf_i,     //Which PRF entry unit x has
                                          //as its destination
  input logic  [2:0] [63:0] fu_data_i,    //FU x’s result.
  output logic [1:0]        cdb_valid_o,  //Data on CDB x is valid
  output logic [1:0] [4:0]  cdb_prf_o,    //CDB y’s dest. PR entry.
  output logic [1:0] [63:0] cdb_data_o,   //CDB y’s result.
  output logic [2:0]        fu_stall_o )  //Stall the functional unit
                                          //if it didn't get the CDB
                                          //the cycle
```

8) Consider a processor that is part of a multi-processor system. Its instruction and data caches are each 16 KB direct-mapped caches with 32-byte cache lines. The data cache is write-back, the instruction cache is write-through. It supports the MESI protocol as described in class. The word size (for instructions and data) is 4 bytes.

Now say you have a program running on that processor. All you know about the program is that it performs 100,000 loads, 20,000 stores and fetches 400,000 instructions (5,000 of which are unique). Answer the following questions. **Show your work**. **[12 points]**

a) Assuming both caches start with all lines marked invalid and all other processors are idle, what is the smallest number of read lines (BRLs) that could generated when this program runs? **[4]**

b) Assuming both caches start with (potentially useful) instructions/data and all other processors are idle, what is the smallest number of read lines (BRLs) that could generated when this program runs? **[4]**

c) Assuming both caches start with (potentially useful) instructions/data and all other processors are idle, what is the ***largest*** number of bus transactions that could generated when this program runs? **[4]**

9) Bus vs. directory-based systems. **[7 points]**

a) Say we have 4 processors each of which has a 1MB private writeback L1 cache with 64-byte cache blocks. The main memory size is 4GB (total) and the system uses the MESI coherence protocol as described in class. How many state-bits do we need to maintain the coherence protocol? Where do those state-bits reside? Show your work and ignore any transient states. **[4]**

b) Now say we are using a directory-based system rather than a bus-based one with the same caches as above. Each of the 4 processors has 1GB of memory (so still 4GB total and the system uses the MESI coherence protocol as described in class. How many state-bits do we need to maintain the coherence protocol? Where do those state-bits reside? Show your work and ignore any transient states (*This problem is quite hard*). **[3]**

10) Consider the state of a machine implementing the R10K scheme with an RRAT. **[15 points]**

RAT	
Arch Reg #	**Phy. Reg #**
0	12
1	4
2	1
3	3
4	8

ROB					
Buffer Number	**PC**	**Executed?**	**Dest. PRN**	**Dest ARN**	
0	60	N	0	1	← HEAD
1	64	N	1	2	
2	66	Y	2	1	
3	70	Y	--	--	
4	74	N	4	1	
5	78	Y	3	3	
6					← TAIL
7					
8					

RRAT	
Arch Reg #	**Phy. Reg #**
0	12
1	11
2	10
3	9
4	8

RS							
RS#	**Op Type**	**Op1 Ready?**	**Op1 PRN/value**	**Op2 Ready?**	**Op2 PRN/value**	**Dest PRN**	**ROB**
0	+	N	1	N	1	4	4
1	*	Y	10	Y	10	0	0
2	+	N	0	Y	10	1	1
3							
4							

PRF			
Phy Reg #	**Value**	**Free**	**Valid**
0	1	N	N
1	2	N	N
2	20	N	Y
3	18	N	Y
4	5	N	N
5	6	Y	N
6	7	Y	N
7	8	Y	N
8	9	N	Y
9	10	N	Y
10	11	N	Y
11	44	N	Y
12	13	N	Y

KEY:

- **Op1 PRN/value** is the value of the first argument if "Op1 ready?" is yes; otherwise it is the Physical Register Number that is being waited upon.
- **Op2 PRN/value** is the same as above but for the second argument.
- **Dest. PRN** is the destination Physical Register Number.
- **Dest. ARN** is the destination Architectural Register Number.
- **ROB** is the associated ROB entry for this instruction.
- **Free/Valid** indicates if the PRF entry is currently available for allocation and if the valid in it is valid. A free entry should be marked as invalid.

Say that the instruction in ROB #3 is a branch and it was mis-predicted: the next PC should have been 100. Now, say that the instruction in memory location 100 is R1=R2+R1, that in location 104 is R4=R1*R3, and 108 is R3=R4+R2. Update the machine to the state where the branch has left the RoB, and the instruction at location 100 has committed, the instruction at 104 has dispatched but not sent to a functional unit yet, and the instruction at 108 has gone as far as it can given the state of the other instructions. When selecting a PRF to use, select the lowest numbered one that is free. ***Be sure to update the head and tail pointers!***

You may use this one or the one on the next page but be sure to cross out (with a BIG X) the one you don't want graded.

(This is a copy of the previous page)

RAT	
Arch Reg #	**Phy. Reg #**
0	12
1	4
2	1
3	3
4	8

ROB					
Buffer Number	**PC**	**Executed?**	**Dest. PRN**	**Dest ARN**	
0	60	N	0	1	← HEAD
1	64	N	1	2	
2	66	Y	2	1	
3	70	Y	--	--	
4	74	N	4	1	
5	78	Y	3	3	
6					← TAIL
7					
8					

RRAT	
Arch Reg #	**Phy. Reg #**
0	12
1	11
2	10
3	9
4	8

RS							
RS#	**Op Type**	**Op1 Ready?**	**Op1 PRN/value**	**Op2 Ready?**	**Op2 PRN/value**	**Dest PRN**	**ROB**
0	+	N	1	N	1	4	4
1	*	Y	10	Y	10	0	0
2	+	N	0	Y	10	1	1
3							
4							

PRF			
Phy Reg #	**Value**	**Free**	**Valid**
0	1	N	N
1	2	N	N
2	20	N	Y
3	18	N	Y
4	5	N	N
5	6	Y	N
6	7	Y	N
7	8	Y	N
8	9	N	Y
9	10	N	Y
10	11	N	Y
11	44	N	Y
12	13	N	Y

KEY:
- **Op1 PRN/value** is the value of the first argument if "Op1 ready?" is yes; otherwise it is the Physical Register Number that is being waited upon.
- **Op2 PRN/value** is the same as above but for the second argument.
- **Dest. PRN** is the destination Physical Register Number.
- **Dest. ARN** is the destination Architectural Register Number.
- **ROB** is the associated ROB entry for this instruction.
- **Free/Valid** indicates if the PRF entry is currently available for allocation and if the valid in it is valid. A free entry should be marked as invalid.

Say that the instruction in ROB #3 is a branch and it was mis-predicted: the next PC should have been 100. Now, say that the instruction in memory location 100 is R1=R2+R1, that in location 104 is R4=R1*R3, and 108 is R3=R4+R2. Update the machine to the state where the branch has left the RoB, and the instruction at location 100 has committed, the instruction at 104 has dispatched but not sent to a functional unit yet, and the instruction at 108 has gone as far as it can given the state of the other instructions. When selecting a PRF to use, select the lowest numbered one that is free. ***Be sure to update the head and tail pointers!***

You may use this one or the one on the previous page but be sure to cross out (with a BIG X) the one you don't want graded.