EECS 470 *Midterm Exam*

Winter 2015

Name: ____ANSWER KEY__________________________ 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	**/20**
3	**/15**
4	**/9**
5	**/15**
6	**/11**
7	**/15**
8 & 9	**/15**
Total	***/100***

NOTES:

- Open book and Open notes
- Calculators are allowed but not ones with communication support (Bluetooth, Cell phones, etc.)
- Don't spend too much time on any one problem.
- You have about 120 minutes for the exam.
- There are ***9*** pages including this one.
- Be sure to show work and explain what you've done when asked to do so.

1) Multiple choice/fill in the blank **[20 points, -2 per blank or wrong answer]**

a) You have a local history predictor where the BHT is indexed by 8 bits of the PC and each entry has 4 bits of history. The PHT is a standard 4-state predictor. We would need

_____1024________ bits of storage for the BHT and _____32____ bits of storage for the PHT.

b) A four-way superscalar processor could ideally achieve an IPC of ____4_____.

c) In the original version of Tomasulo's algorithm, the RAT points to a ***reservation station / RoB entry / PRF entry***.

d) Test-and-set is a(n) ___atomic_____ operation, meaning that the test (read) and set (write) happen back-to-back without any operation occurring between the read and write.

e) A processor implementing the R10K scheme has 16 RoB entries, 4 RSes, and 40 PRF entries. It implements an architecture with 32 architected registers. In the steady state, when a branch mispredicts, it could free *at most* ***1 / 2 / 4 / 8 / 16 / 32 / 40*** PRF entries.

f) A processor implementing the R10K scheme has 16 RoB entries, 4 RSes, and 48 PRF entries. It implements an architecture with 32 architected registers. Assuming all PRF entries are in use, the fewest number of RAT entries than could be different than the RRAT is ***1 / 2 / 4 / 8 / 16 / 32 / 40***.

g) Given a 2-KB, eight-way associative, cache with 16-byte cache lines and a 32-bit

address space there will be ___4_____ bits used for the index. If that same cache were

direct-mapped you'd need __**21**______ bits to be used for the tag.

h) Say you have an ISA where all instructions are 32-bits and which has 32 general purpose registers and all immediate values are 16-bits. If your instruction set consisted of nothing other than instructions that used one GPR and one immediate, you could have up to ***256 / 512 / 1024 / 2048 / 4096*** instructions total in your ISA.

i) If a given application were perfectly parallelizable for 90% of its execution time on one processor and perfectly serial for the rest of the time, you would expect the application to

have a speedup of ___**250**___% on 3 processors.

j) Adding more RoB entries to a given processor running the P6 scheme is likely to improve CPI because it reduces the number of stalls due to ***data hazards / structural hazards/ cache misses / control hazards***.

2) Explain the following sentence from McFarling's "Combining Branch Predictors" paper **[7 points]**

As we would expect, gselect-best performs better than either bimodal or global prediction since both are essentially degenerate cases.

The gselect predictor uses a concatenation of bits from the branch PC and global history register as the index. gselect is a degenerate case of bimodal and global history predictor because when we use all the bits of PC and 0 bits from global history it is bimodal and when we use all bits of global history register and 0 from PC it is a global history predictor. So both bimodal and global history are manifestations of gselect. Gselect-best is the combination of PC and global history bits that gives the best prediction rates. The authors in the paper run a set of simulations to find out the best combination of PC and global history bits. Since Gselect-best uses the best combination of PC and global history bits, it will perform at least as well as a bimodal or global history predictor.

3) Say that in the "R10K" algorithm, you have 32 RoB entries, 6 RS entries, 48 physical registers and the ISA supports 16 architected registers. How many bits would you need for the RRAT (just the pointers in the table, not valid bits, the free list or anything else)? You must show your work to receive any credit. **[4 points]**

16 lines x 6 bits/line = 96 bits

4) Circle each the following that are ***reasons*** we don't build processors with massive architected register files (say thousands of architected registers)? **[4 points, -2 per wrong/missing circle]**

- *They would likely cause more spills and fills to occur.*
- *They would likely increase the CPU's clock period.*
- *They would likely reduce the number of instructions we can encode in a 32-bit instruction.*
- *They would make having a RoB nearly useless.*
- *They would likely reduce the data cache hit rate.*

5) You go to a talk on computer architecture and the speaker makes the following statement:

> *Our study shows that, as expected, our 3-way superscalar processor can only achieve a CPI approaching 1/3 if the frontend (fetch, decode, dispatch) is actually 4-wide.*

Why does the frontend need to be wider than the backend to achieve high performance? Your answer must be 30 words or less to receive any credit. **[5 points]**

- **Things like branch mispredictions require instruction squashing, which means more instructions go through frontend than get committed from ROB.**
- **(less-good answer) If the RS has more instructions to choose from for issue, it can better utilize the EX units, ROB, etc.**

6) The select logic in an out-of-order processor sometimes must choose from several ready instructions to decide what to issue to the execute stage. It was claimed in class that this can make a big difference in real processors, but rarely does in our class project.

For real processors, which of the following three heuristics for selecting which instruction to issue do you believe will lead to the highest performance? Briefly justify your choice (1-2 sentences).
[4 points]

- Top – Select the instruction that appears in the lowest-numbered reservation station
- Eldest – Select the eldest instruction (in program order) in the reservation stations
- Youngest – Select the youngest instruction (in program order) in the reservation stations

Eldest. The risk of inadvertent starvation is a real problem. By choosing the oldest instruction, you prevent dependent instructions from waiting even longer in the RS. You also avoid the structural hazard of having the ROB fill up.

7) The company "Processor's 'R Us" is trying to improve the performance of their branch resolution logic. Their baseline processor design achieves an IPC of 0.8 over a suite of benchmark applications. Their proposed change shortens the branch execution pipeline, shaving 2 cycles off the average branch mispredict penalty. However, the design change increases the clock period by 5%. Over the entire benchmark suite, 15% of instructions are branches. Unfortunately they do not know the prediction accuracy of the branch predictor.

For what ***range*** of *prediction* accuracies will this change result in a performance improvement? Clearly show how you arrived at your answer. **[9 points]**

0 % < P < ~80 %

8) Consider a local pattern history predictor where The BHT has 16 entries, each with 3 bits of history. The predictors in the PHT are each 1 bit. What steady-state prediction rate would this predictor get on a branch that had the following pattern? You can assume that there are no other branches.
[6 points, 2 each]

- 3 taken, 1 not taken repeating forever (T,T,T,NT,T,T,T,NT, etc.)
  100%
- 4 taken, 1 not taken repeating forever (T,T,T,T,NT,T,T,T,T,NT) etc.
  60%
- 5 taken, 1 not taken repeating forever (T,T,T,T,T,NT,T,T,T,T,T,NT) etc.
  66%

Write a SystemVerilog module which implements a 3-bit saturating up counter. The counter takes the following inputs: enable, reset, and clock. It generates the following outputs: counter[2:0]. Reset is a synchronous reset signal should reset the counter to zero no matter the value of enable. When enabled the counter should increment by one on the rising edge of clock unless it has saturated. Your code will be graded for using proper style and following our coding guidelines.
[11 points]

```
module counter  (  input clock,
                   input reset,
                   input enable,
                   output logic [2:0] counter
                   );

    always_ff @(posedge clock) begin
        if (reset)
            counter <= #1 3'b0;
        else if (enable && counter < 3'b111)
            counter <= #1 counter + 3'b1;
    end
```

10) Consider the following pseudo-assembly code:

```
           r3=100000
           r5=0
           r6=0
     bob:  r1=MEM[r3+0]                 // THE LOAD
           if(r1>0) goto ONE            // Branch 1
           r5=r5+1
           r6=r6+r1
           r6=r6/2
     ONE:  if(r1>=0) goto TWO           // Branch 2
           r6=r6+r3
           r6=r6+1
     TWO:  r3=r3+4
           if(r5<50000) goto bob        // Branch 3
```

The predictors all use the least significant bits of the PC other than the word-offset. Predictors and patterns are all initialized to all zeros (not taken).

You are to consider how different branch predictors will behave on this code under different circumstances.

- **Case 1:** The data loaded from memory follows the pattern (-1, 1) repeating forever. So (-1, 1, -1, 1, etc.)
- **Case 2:** The data loaded from memory follows the pattern (-1, -1, 0, 1) repeating forever. So (-1, -1, 0, 1, -1, -1, 0, 1, etc.)

You are now to consider 3 branch predictors:

- **Predictor 1:** A bi-modal predictor with 8 entries, each 1 bit in size.
- **Predictor 2:** A bi-modal predictor with 4 entries, each 2 bits in size.
- **Predictor 3:** A local pattern history predictor. The BHT has 16 entries, each with 2 bits of history. The predictors are each 1 bit.

What are the expected prediction *rates* for each of the following (percentage of time right)? Your answers must be correct within 1.0%. **[15 points, -1 per wrong or blank box, min 0]**

	Case 1			**Case 2**		
	Predictor 1	*Predictor 2*	*Predictor 3*	*Predictor 1*	*Predictor 2*	*Predictor 3*
Branch 1	50	50	100	25	50	50
Branch 2	0	50	100	50	50	25
Branch 3	50	50	100	25	25	75

RAT	
Arch Reg #	Phy. Reg #
0	6
1	~~3~~ 1
2	~~5~~ 0
3	~~11~~ 9
4	10

ROB					
Buffer Number	PC	Executed ?	Dest. PRN	Dest ARN	
~~0~~	~~20~~	~~N Y~~	4	~~1~~	~~□ HEAD~~
~~1~~	~~24~~	~~N Y~~	~~5~~	~~2~~	
~~2~~	~~28~~	~~Y~~	~~12~~	~~1~~	
~~3~~	~~32~~	~~Y~~	~~--~~	~~--~~	
4	~~36~~	~~N~~	~~11~~	~~3~~	
~~5~~	~~40~~	~~Y~~	~~3~~	~~1~~	□~~TAIL~~
6	100	Y	0	2	<— HEAD
7	104	N	1	1	<— TAIL
8					

RRAT	
Arch Reg #	Phy. Reg #
0	6
1	~~7~~ 12
2	~~8~~ 5
3	9
4	10

RS							
RS#	Op Type	Op1 Ready?	Op1 PRN/value	Op2 Ready?	Op2 PRN/value	Dest PRN	ROB
0	~~*~~	~~Y~~	~~11~~	~~Y~~	~~-1~~	4	~~0~~
1	~~+~~	~~Y~~	~~13~~	~~N~~	~~5~~	~~11~~	4
2	~~+~~	~~Y~~	~~13~~	~~Y~~	~~13~~	~~0~~	~~6~~
3	~~+~~	~~N~~	4	~~N~~	4	~~5~~	~~1~~
4	+	Y	26	Y	26	1	7

PRF			
Phy Reg #	Value	Free	Valid
0	26	N	Y
1	5	N	N
2	6	Y	N
3	26	Y	N
4	-11	Y	N
5	-22	N	Y
6	10	N	Y
7	11	Y	N
8	12	Y	N
9	-1	N	Y
10	14	N	Y
11	15	Y	N
12	13	N	Y