EECS 470 *Midterm 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	**/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.
- **There are two "answer areas" for the last problem. Clearly mark which one you want graded or we will grade the first one.**

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

_____________ bits of storage for the BHT and _____________ bits of storage for the PHT.

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

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) ______________ 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 ________ bits used for the index. If that same cache were

direct-mapped you'd need ________ 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 __________% 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.

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]**

4) <u>Circle</u> each the following that are ***<u>reasons</u>*** 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]**

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

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]**

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.)

- 4 taken, 1 not taken repeating forever (T,T,T,T,NT,T,T,T,T,NT) etc.

- 5 taken, 1 not taken repeating forever (T,T,T,T,T,NT,T,T,T,T,T,NT) etc.

9) 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]**

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.

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

- **<u>Predictor 1:</u>** A bi-modal predictor with 8 entries, each 1 bit in size.
- **<u>Predictor 2:</u>** A bi-modal predictor with 4 entries, each 2 bits in size.
- **<u>Predictor 3:</u>** 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						
Branch 2						
Branch 3						

11) Consider the following state of a machine implementing what we've called the "R10K" algorithm. **[15 points]**

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

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

RRAT	
Arch Reg #	**Phy. Reg #**
0	6
1	7
2	8
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							
3	+	N	4	N	4	5	1
4							

PRF			
Phy Reg #	**Value**	**Free**	**Valid**
0	4	Y	N
1	5	Y	N
2	6	Y	N
3	26	N	Y
4	8	N	N
5	9	N	N
6	10	N	Y
7	11	N	Y
8	12	N	Y
9	-1	N	Y
10	14	N	Y
11	15	N	N
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. Say that the instruction in memory location 100 is R2=R1+R1 and that the instruction in 104 is R1=R2+R2. Update the machine to the state where the branch has left the RoB, and the instructions at memory 100 has been dispatched and executed, but not committed while the one at location 104 has been dispatched but not executed. Use the lowest numbered PRF entries first. ***Be sure to update the head and tail pointers!***

On the following page is an extra copy of this problem. 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 state. You may use this one or the previous one but be sure to cross out (with a BIG X) the one you don't want graded.

Consider the following state of a machine implementing what we've called the "R10K" algorithm. **[15 points]**

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

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

RRAT	
Arch Reg #	**Phy. Reg #**
0	6
1	7
2	8
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							
3	+	N	4	N	4	5	1
4							

PRF			
Phy Reg #	**Value**	**Free**	**Valid**
0	4	Y	N
1	5	Y	N
2	6	Y	N
3	26	N	Y
4	8	N	N
5	9	N	N
6	10	N	Y
7	11	N	Y
8	12	N	Y
9	-1	N	Y
10	14	N	Y
11	15	N	N
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. Say that the instruction in memory location 100 is R2=R1+R1 and that the instruction in 104 is R1=R2+R2. Update the machine to the state where the branch has left the RoB, and the instructions at memory 100 has been <u>dispatched</u> and <u>executed</u>, but not committed while the one at location 104 has been <u>dispatched</u> but <u>not executed</u>. Use the lowest numbered PRF entries first. ***Be sure to update the head and tail pointers!***