EECS 470 *Midterm Exam*

Fall 2013

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	**/15**
3	**/12**
4	**/12**
5	**/12**
6	**/16**
7	**/7**
8	**/6**
9	**/20**
Total	***/100***

NOTES:

- Open book and Open notes
- Calculators are allowed but not if it has 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 ***10*** pages including this one.
- There are two copies of the last problem. Cross out the one you don't want graded.
- Be sure to show work and explain what you've done when asked to do so.
- The problem on page 4 may be long and/or difficult. You might want to do it last.

1) Circle the correct answer **[15 points, -2 per blank or wrong answer]**

a. In the R10K algorithm, entries in the RAT points to a ***reorder buffer entry / a reservation station /*** A PHYSICAL REGISTER ***/ an execution unit***.

b. Say you have a processor using the R10K algorithm. It has 16 RoB entries, 32 architected registers (none of which hold a fixed value like R31 does on the Alpha), and 64 PRF entries. Once the processor has been running for a while, the *fewest* PRF entries that will ever be free is ***0 /*** 16 ***/ 24 / 32 / 64*** and the *most* that will ever be free is ***0 / 16 / 24 /*** 32 ***/ 64***.

c. Say a processor with a 36-bit address space (byte addressed) has an L1 cache with a 2KB data store which is fully associative. Assuming the cache lines are each 16 bytes, the tag store is 512 ***/ 1280 / 2968 / 4096 / 5552*** bytes (including only the tags themselves).

d. You would expect that adding more reservation stations would increase ***the CPI / the clock frequency*** / THE FANOUT OF THE CDB for a given processor.

e. In the "R10K" algorithm, we will free ***all PRF entries / no PRF entries /*** THOSE PRF ENTRIES WHICH AREN'T POINTED TO BY THE RRAT ***/ those PRF entries in the RAT that are overwritten by the RRAT*** when a branch mispredict occurs.

f. "Power constrained computing" refers to devices where the primary concern is related to THE DEVICE GETTING TOO HOT ***/ the device using battery power too quickly / the device needing a higher voltage than we can easily supply***.

g. A cylindrical wire which is 1cm long and 50nm in diameter will have A HIGHER ***/ a lower / the same*** resistance as one which is 0.5cm long and 100nm in diameter.

h. Say you had an ISA where all instructions are 32-bits. Say that there are 64 physical registers, 8 reservation stations, 48 RoB entries and 32 architected registers and all instructions write to a destination register.
   - If your instruction set consisted of nothing other than instructions that used one GPR and one 16-bit immediate, you could have up to ***512 / 1024 /*** 2048 ***/ 4096 / 8192*** instructions.
   - You would never have a structural hazard solely due to the ***reservation stations /*** REORDER BUFFER ENTRIES ***/ physical register file.***

2) Circle the correct answer. For this problem the processors are all non-superscalar.
[5 points, -3 for each blank/wrong answer, minimum 0]

a. If, in the "P6" algorithm, the RAT had only one write port, what structural hazard would be introduced (circle one)?
- There would occasionally be a conflict between dispatch and commit.
- There would occasionally be a conflict between dispatch and execution complete.
- There would occasionally be a conflict between execution complete and dispatch.
- There would never be a structural hazard due to this issue.

b. If, in the "R10K" algorithm, the PRF had only one write port, what structural hazard would be introduced (circle one)?
- There would occasionally be a conflict between dispatch and commit.
- There would occasionally be a conflict between dispatch and execution complete.
- There would occasionally be a conflict between execution complete and dispatch.
- There would never be a structural hazard due to this issue.

PROBLEM WAS THROWN OUT AS IT WAS UNCLEAR IF THE FREE AND/OR VALID LISTS WERE PART OF THE RAT OR PART OF THE PRF. ALL ANSWERS WERE TAKEN.

3) Below is a list of statements. For each entry on the list indicate which of the three out-of-order algorithms we discussed in class the statement is *true* for by putting an X in the appropriate box and otherwise leaving the box blank. A given statement may be true for all three algorithms, some combination of algorithms or none.
[7 points, -1 per wrong box, minimum 0]

Statement	Original Tomasulo's Algorithm	P6	R10K
Provides precise exception handling		X	X
Has a trivial bound on "window size"[1]		X	X
Only need to look for a source operand on the CDB or one other place.	X		X
Instructions may modify architectural state out-of-order			
Uses reservation stations	X	X	X
Suffers no performance impact from RAW hazards			

[1] THIS QUESTION WAS CONFUSING TO A FEW PEOPLE. "TRIVIAL BOUND" MEANS A BOUND THAT IS SIMPLE OR EASY TO FIND IN THIS CASE IT'S THE ROB SIZE. THERE IS NO FIXED BOUND FOR THE ORIGINAL ALGORITHM.

4) Consider the following pseudo-assembly code:

```
       r2=0
       r4=0
       r5=0
 bob:  r3=(r2 mod 2)   // remainder when r2 is divided by 2
       if(r3==0) goto next     // Branch 1
       r4=r4+1
next:  r5=r5+4
       r2=MEM[r5+0]            // Load
       if(r4<50000) goto bob   // Branch 2
```

"bob" has an address of 0x1000. The predictors all use the least significant bits of the PC other than the word-offset. Predictors are all initialized to "0" or "00" which is "not-taken" and "strongly not-taken" respectively.

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

- **Case 1:** The data from the load will be "1" the first time, "2" the second, "3" the third etc.
- **Case 2:** The data from the load will be random. (each instance is independent, with no bias toward even or odd numbers)
- **Case 3:** The data is even the first five times in a row and then odd, and then even five times in a row, and then odd, then even etc.

You are now to consider 3 branch predictors:

- **Predictor 1:** A PC-based predictor with 4 entries each 1 bit.
- **Predictor 2:** A PC-based predictor with 8 entries each a 2-bit saturating counter.
- **Predictor 3:** A local pattern history predictor. The BHT has 16 entries, each with 3 bits of history. The predictors are each 1 bit.

What are the expected mis-predict *rates* for each of the following? Your answer must be correct within 0.5%. **[12, -1 per wrong or blank box, min 0]**

	Case 1		**Case 2**		**Case 3**	
	Branch 1	*Branch 2*	*Branch 1*	*Branch 2*	*Branch 1*	*Branch 2*
Predictor 1	50	50	50	50	1/6	1/6
Predictor 2	50	0	50	0	1/6	0
Predictor 3	0	0	50	1/16	1/6	1/6

5) Consider a set of code where there are two classes of instructions.
   - *"Short"* instructions are not dependent on any other instruction and can execute in 4 cycles.
   - *"Long"* instructions are not dependent on any other instruction and can execute in 40 cycles.
   - *"Dependent"* instructions are (only) dependent on the instruction in front of them and take 4 cycles to execute.

   Say you have a machine which can issue one instruction per cycle, finish execution of one instruction per cycle, and retire one instruction per cycle. This machine implements what we have called the "P6" algorithm and it frees its RS when the instruction is sent to its functional unit. The machine has an RS size of 8 and a RoB size of 16. You may assume the machine otherwise has unlimited resources (execution units etc.) *You must show/explain your work to get credit!* **[12 points]**

   a. What is the best CPI this machine could achieve if the program being run consisted of only "*long*" instructions? **[4]**

      THE LONG INSTRUCTIONS WILL IMMEDIATELY LEAVE THE RS AS SOON AS THEY ENTER. BUT THEY WILL OCCUPY THE ROB UNTIL THEY RETIRE. SO 16 WILL ENTER, THERE WILL BE A STALL OF ~24 CYCLES AND THEN AS THOSE 16 COMMIT THERE WILL BE 16 NEW ONES COMING IN AND AGAIN STALLING FOR ~24 CYCLES. SO C=40, I=16 AND CPI=2.5

   b. What is the best CPI this machine could achieve if the program being run consisted of only "*dependent*" instructions? **[4]**
      4: EACH INSTRUCTION WOULD HAVE TO WAIT UNTIL THE ONE IN FRONT OF IT COMPLETED.

   c. What is the best CPI this machine could achieve if the program being run consisted of groups of 100 instructions, where the first 99 were "dependent" and the last was "short". Assume there are a large number of these groups. (So the code is 99 dependent, 1 short, 99 dependent, 1 short, etc.) **[4]**

      ~3.72. THE RS IS GOING TO BE THE LIMITING FACTOR HERE. FOR A SHORT TIME TWO DEPENDENT CHAINS WOULD BOTH BE MAKING FORWARD PROGRESS. THAT WOULD HAPPEN FOR (ABOUT) 4*7 CYCLES. SO (400-28)/100=3.72.

6) We have an architecture that we are told has a CPI of 1.2 over a given set of benchmarks, where the only reason the CPI isn't 1.0 is due to branch mis-predicts. Someone has come up with a way to cut the branch mis-predict penalty by 2 cycles, but at a cost of increasing the clock period by 5%. We know 20% of instructions are branches. What would be the speedup of this new scheme assuming the prediction rate were 90%? You must show your work to receive credit. **[6 points]**

SPEEDUP=TOLD/TNEW.

2% OF ALL INSTRUCTIONS ARE BRANCH MISPREDICTS. THAT AND THE CPI OF 1.2 IMPLY THE MISPREDICT PENALTY IS 10 CYCLES. SO THE NEW SCHEME WOULD HAVE A CPI OF 1.16. HOWEVER THE CYCLE TIME IS 1.05 AS MUCH. THAT GIVES 1.218 "TIME UNITS" PER INSTRUCTION IN THE NEW SCHEME COMPARED TO 1.2 IN THE OLD. SPEED UP IS THEN 0.985—THAT IS THE NEW SCHEME IS SLOWER.

7) Consider the pipelined processor you did as part of project 3 but with the structural hazards removed (so you can fetch and load/store in the same cycle). Say 30% of all instructions were loads, 20% were branches, and 10% were stores. Assume the program is quite long (millions and millions of instructions). In addition, 25% of all instructions are data dependent on the instruction in front of them, while the rest are dependent on the instruction after that. Branches are taken 25% of the time. **[10 points]**

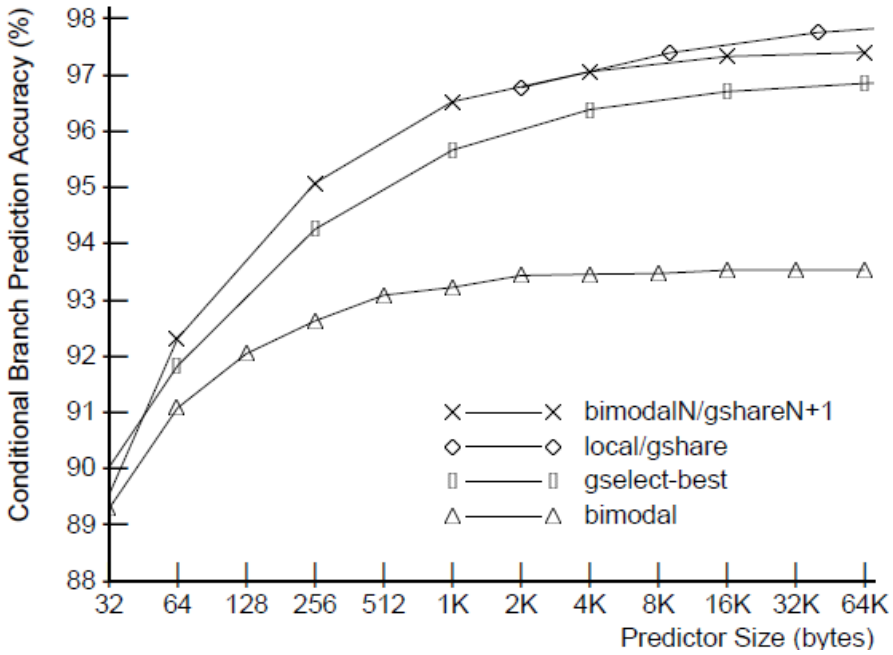
a. What would you expect the CPI to be? **[4]**

1+[DATA HAZARDS STALLS]+[CONTROL HAZARD STALLS]=
1+[.3*.25*1]+[.2*.25*3]=1+0.075+0.15=1.25

b. If we instead resolved the branch in decode, what would you expect the CPI to be? Assume you can add forwarding paths to decode as needed. **[6]**

1+[DATA HAZARDS STALLS]+[CONTROL HAZARD STALLS]=
1+[NEW DHS + OLD DHS]+[CONTROL HAZARD STALLS]=
1+[BRANCH DEPENDENT ON NEXT]+[BRANCH DEPENDENT ON LOAD 2 AWAY]+OLD DHS+[CONTROL HAZARD STALLS]=
1+(.2*.25)+(.2*.3*.75)+.075+.15=1.32
(NOTE: BRANCHES DEPENDENT ON A LOAD IMMEDIATELY IN FRONT OF THEM STALL TWICE, THAT IS DONE BY THE OLD DHS PLUS THE EXTRA ONE FOR DEPENDENT ON THE NEXT INSTRUCTION—THERE ARE A NUMBER OF WAYS TO FORMULATE THIS).

8) Examine the following graph and then answer the questions found below.
[7 points]



a) Which of the four predictors shown above (if any) use *no* global information?
[3, no partial credit]

BIMODAL DOESN'T USE GLOBAL INFORMATION. A FEW FOLKS ARGUED THAT IT DID DUE TO ALIASING AND THEY GOT CREDIT. THAT SAID, GLOBAL GENERALLY MEANS THE ENTIRE HISTORY, NOT JUST HOW A FEW OTHER BRANCHES BEHAVE.

b) In your own words, explain what the local/gshare predictor is. Your answer must be less than 30 words in length. **[4]**

IT USES A TOURNAMENT SELECTOR TO CHOOSE BETWEEN A LOCAL-HISTORY PREDICTOR AND A GSHARE PREDICTOR.

8. Consider the following programs

Program A	**Program B**
R1=MEM[R2+0]	R1=MEM[R2+0]
R2=R3+4	R2=R3+4
R3=R1+R7	R3=R4+R1
R4=R2+6	R4=R2+6
R5=R4+R3	**R5=R2+R4**
R6=R8+R6	R6=R9+R10

Say we have an out-of-order machine with 3 RSs and 5 ROB entries. Further, say the first load stalls for a long time. Both programs will come to a halt waiting for the load to finish. For each program indicate the last instruction that will be placed in the ROB previous to the load finishing. You are to assume an instruction stays in its RS until it completes execution. **[6 points]**

BOTH ARE LIMITED BY THE ROB.

There are two copies of this problem. Please cross out the one you don't want graded, otherwise we will grade the one on page 9

9. In this problem we are using the R10K algorithm with an RRAT. Say the ROB, RS, RAT, PRF, and RRAT are as follows: NOTE: THE INFORMATION IN THE RS COULD BE IN ANY RESERVATION STATION.

RAT

Arch Reg. #	Phys. Reg. #
0	1
1	~~3~~ 2
2	4
3	~~5~~ 10
4	9
5	~~8~~ 5

ROB

Buffer Number	PC	Done with EX?	Dest. Arch Reg #	Dest. Phy. Reg #
0	~~16~~	~~N~~	~~1~~	~~3~~
1	~~20~~	~~N~~	~~2~~	~~0~~
2	~~24~~	~~Y~~	~~--~~	~~--~~
3	~~28~~	~~Y~~	~~2~~	~~4~~
4	~~32~~	~~N~~	~~3~~	~~5~~
5	80	N	1	2
6	84	Y	2	4
7	88	N	5	5

RRAT

Arch Reg. #	Phys. Reg. #
0	1
1	~~2~~ 3
2	~~11~~ 0
3	10
4	9
5	8

Head of ROB=~~0~~ 5

Tail of ROB=4 7

RS

RS	Op type	Op1 ready	Op1 PRN/value	Op2 ready	Op2 PRN/value	Dest. PRN	ROB
0	+	~~Y~~ N	~~-1~~ 2	Y	~~-6~~ 0	~~3~~ 5	~~0~~ 5
1	~~+~~	~~N~~	~~3~~	~~Y~~	~~-2~~	~~0~~	~~1~~
2	~~+~~	~~Y~~	~~4~~	~~Y~~	~~5~~	~~5~~	~~4~~
3	+	Y	-9	Y	-9	2	5

Phy. Register File

Phy. Reg	Value	Free	Valid
0	~~1~~ -9	N	~~N~~ Y
1	2	N	Y
2	3	N	~~Y~~ N
3	~~4~~ -7	N	~~N~~ Y
4	~~-1~~ -72	N	Y
5	-2	N	N
6	-3	Y	N
7	-4	Y	N
8	8	N	Y
9	-9	N	Y
10	0	N	Y
11	-5	~~N~~ Y	~~Y~~ N

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.
- **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

Say that the branch in RoB entry 2 was mispredicted and should have branched to "top". Set the machine state so that the following instructions are dispatched and so that A has not yet started execution, while B and C have progressed as far as they can

```
top:  R1=R2+R4        // A
      R2=R4*R5        // B
      R5=R1+R3        // C
```

Fill the different structures to get to the point described. The PRF will always allocate the lowest numbered free PRF entry. The PC of instruction A is 80 and each instruction is 4 bytes in size.

[20 points]

There are two copies of this problem. Please cross out the one you don't want graded, otherwise we will grade the one on page 9

There are two copies of this problem. Please cross out the one you don't want graded, otherwise we will grade the one on page 9

9. In this problem we are using the R10K algorithm with an RRAT. Say the ROB, RS, RAT, PRF, and RRAT are as follows:

RAT	
Arch Reg. #	**Phys. Reg. #**
0	1
1	3
2	4
3	5
4	9
5	8

ROB				
Buffer Number	**PC**	**Done with EX?**	**Dest. Arch Reg #**	**Dest. Phy. Reg #**
0	16	N	1	3
1	20	N	2	0
2	24	Y	--	--
3	28	Y	2	4
4	32	N	3	5
5				
6				
7				

RRAT	
Arch Reg. #	**Phys. Reg. #**
0	1
1	2
2	11
3	10
4	9
5	8

Head of ROB=0

Tail of ROB=4

RS							
RS	**Op type**	**Op1 ready**	**Op1 PRN/value**	**Op2 ready**	**Op2 PRN/value**	**Dest. PRN**	**ROB**
0	+	Y	-1	Y	-6	3	0
1	+	N	3	Y	-2	0	1
2	+	Y	4	Y	5	5	4
3							

Phy. Register File			
Phy. Reg	**Value**	**Free**	**Valid**
0	1	N	N
1	2	N	Y
2	3	N	Y
3	4	N	N
4	-1	N	Y
5	-2	N	N
6	-3	Y	N
7	-4	Y	N
8	8	N	Y
9	-9	N	Y
10	0	N	Y
11	-5	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.
- **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

Say that the branch in RoB entry 2 was mispredicted and should have branched to "top". Set the machine state so that the following instructions are dispatched and so that A has not yet started execution, while B and C have progressed as far as they can

```
top: R1=R2+R4        // A
     R2=R4*R5        // B
     R5=R1+R3        // C
```

Fill the different structures to get to the point described. The PRF will always allocate the lowest numbered free PRF entry. The PC of instruction A is 80 and each instruction is 4 bytes in size.

[20 points]

There are two copies of this problem. Please cross out the one you don't want graded, otherwise we will grade the one on page 9