EECS 470 *Midterm Exam* ANSWER KEY

Winter 2011

Name: _____ unique name: _____

Sign the honor code:

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

_____

Scores:

#	Points
Page 2	/12
Page 3	/12
Page 4	/12
Page 5	/14
Page 6	/6
Page 7	/10
Page 8	/8
Page 9	/8
Pages 10 & 11	/18
Total	/100

NOTES:

- Open book and Open notes
- 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.
- There are ***11*** pages including this one.
- **Be sure to show work and explain what you've done when asked to do so.**
- **The last page has two "answer areas". Clearly mark which one you want graded or we will grade the first one.**

1. Fill-in-the-blank or circle the *best* answer **[12 points, -2 per wrong/blank, minimum 0]**

   a. Given a 16-KB, four-way associative cache with 64-byte lines, you will need __6____ bits to index the cache. If virtual and physical addresses were both 32-bits in size, the cache would need about **1KB / 4KB / 6KB / 8KB** to store all the tags.

   b. In the algorithm we've called T2, an instruction which is **COMMITTING / completing execution/ issuing** will write a value to the ARF. In T3 an instruction which is **committing / COMPLETING EXECUTION / issuing** will write a value to the PRF.

   c. You would expect a wire 1cm long with a 200nm^2 cross-sectional area to have a **higher / LOWER / identical** resistance than a wire 2cm long and 100nm^2 cross-sectional area.

   d. The period of a 200MHz clock is ____5_____ns.

   e. Consider a processor with 8 Reservation Stations, 32 RoB entries, 32 architected registers, and 64 physical registers. Ignoring state bits, the RAT would use **48 / 64 / 160 / 192 / 256 / 384** bits total.

   f. Say you have an ISA where all instructions are 32-bits and which has 16 general purpose registers and all immediate values are 18-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.

2. Each of the following questions ask you to either put a checkmark (an X) next to a statement or leave it unmarked. **[12 points, -3 for each incorrectly marked (or unmarked) answer, minimum 0]**
    a. Consider the following statements. Place a checkmark next to those statements that are generally TRUE when a Tomasulo scheduler is added to a pipeline which previously used in-order scheduling.

        _X_ The pipeline is able to better exploit instruction level parallelism.

        _X_ The pipeline sees an increased utilization of functional units.

        ___ The pipeline no longer has a need for register renaming.

        _X_ The pipeline can more easily hide memory latency.

    b. Place a checkmark next to the statements about instruction level parallelism (ILP) that are generally TRUE.

        _X_ The addition of register renaming exposes additional ILP.

        ___ It is a waste of resources to add support for thread-level parallelism to a machine that can effectively exploit ILP.

        _X_ You generally find more ILP in scientific/floating point code than in general purpose/integer code.

        _X_ Adding dynamic branch prediction to a pipeline with no branch prediction will expose additional ILP.

3. Consider a local history predictor where the Branch History Table has 1024 entries each 3 bits in length and the Pattern History Table consists of a standard 2-bit predictor. The BHT has not tags. **[12 points]**

   a. How many bits of memory are used in the BHT? The PHT? **[3]**

      BHT bits=__3072___ **PHT bits= __16___**

   b. Say that a single branch is being taken 5 times in a row and then is not taken 2 times in a row and repeats this pattern forever (T,T,T,T,T,N,N,T,T,T,T,T,N,N, etc.) Assuming there is no aliasing of any kind, what will be the steady-state prediction rate for this branch? Justify your answer. **[6]**

      6/7, THE T,T,N CASE WILL BE MISPREDICTED.

   c. Assuming that: all instructions are 32-bits in size, all instructions are aligned, and the machine is byte-addressable, what is the *minimum distance* (in bytes) two instructions must be apart from each other in order to alias in the BHT? **[3].**

      Distance between instructions for the BHT to alias = __4096____ bytes

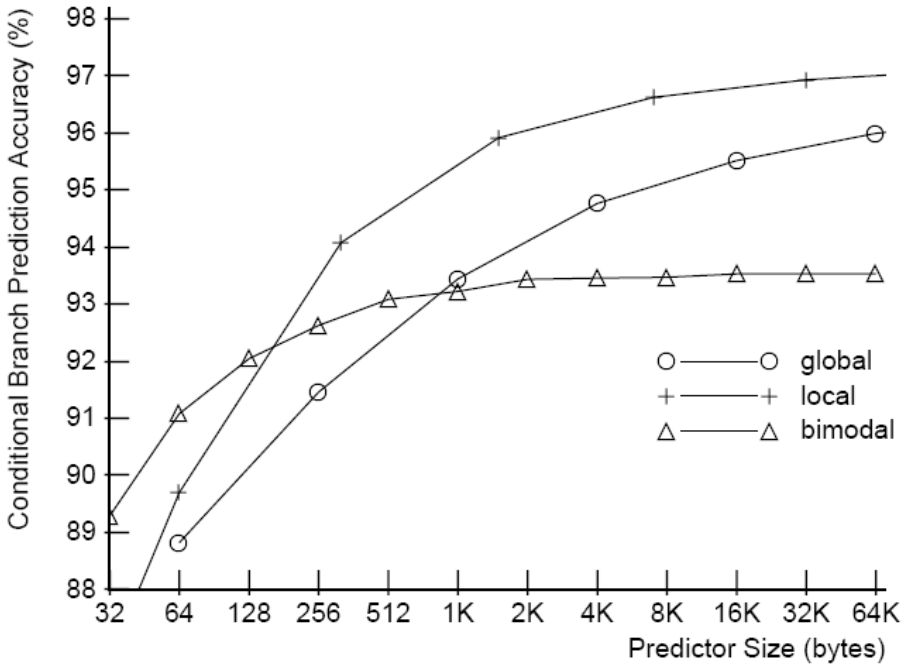
YES, THAT DOES ASSUME THE BITS USED ARE THE LEAST SIGNIFICANT OTHER THAN THE OFFSET.

Given the following design changes to a simple out-of-order pipeline (and assuming no other changes to the pipeline or workload), what would be the effect on the i) number of instructions committed (N_{inst}), ii) the cycles-per-instruction (CPI), iii) clock period (t_{clk}), and iv) silicon area cost (A_{cost}). For each possible effect, indicate one of the following: no change (Ø), equal or greater (↑), equal or less (↓), or not enough information to determine (?). Provide the best answer.
[14 points, -.5 per wrong or blank answer]

Design Change	**N_{inst}**	**CPI**	**t_{clk}**	**A_{cost}**
Shrink the silicon process from 250nm to 130nm	Ø	Ø↑	↓	↓
Double the number of reservation stations	Ø	↓	↑	↑
Switch from using a local dynamic branch predictor to a global one	Ø	?	?	?
Add compile-time optimization to the compiler	↑?↓	?	Ø	Ø
Double the number of registers available to the compiler	↓	↓	↑	↑
Add a multiply instruction to the pipeline (which previously it didn't have)	↓	↑	↑?	↑
Double the number of pipeline stages.	Ø	↑	↓	↑

IN SOME CASES, MORE THAN ONE ANSWER WAS ACCEPTED. ALL ACCEPTED ANSWERS ARE LISTED.

4. Answer the following questions about predictors. **[6 points]**



a. *Briefly* define the following terms in the context of this graph: **[3]**

Global: PREDICTION IS BASED ON THE GLOBAL HISTORY REGISTER. THAT IS THE TAKEN/NOT TAKEN PATTERN OF THE MOST RECENT X BRANCHES.

Local: PREDICTION IS BASED ON THE LOCAL HISTORY OF THE BRANCH (PLUS POTENTIAL ALIASING). THAT IS THE TAKEN/NOT TAKEN PATTERN OF THE MOST RECENT X TIMES THIS BRANCH WAS EXECUTED.

Bimodal: PREDICTION IS BASED ON THE PC OF THE BRANCH.

b. Gshare is best understood as combining which of the two above predictor types? *Briefly* explain your answer. **[3]**

THE GLOBAL AND BIMODAL PREDICTORS. GSHARE INVOLVES XORING THE PC AND THE GLOBAL HISTORY REGISTER TO FIND A PREDICTION.

6. Misprediction CPI penalties **[10 points]**
   a. Give an expression to calculate average CPI of a branch instruction (CPI_{br}) for an in-order pipeline, assuming that branches take one cycle to execute when not mispredicted. *You are to assume for this problem that branch predictor and BTB mispredictions never occur at the same time*. Make your expression a function of (possibly all) of the following machines characteristics: **[4]**
      - T_{mp} = penalty, in cycles, for a branch or BTB misprediction
      - p_{br} = probability of a branch direction misprediction
      - p_{BTB} = probability of a BTB target misprediction
      - N_{br} = size of the branch predictor (in entries)
      - N_{BTB} = size of the BTB (in entries)

CPI_{br}= $1+(p_{pr} + p_{BTB})*T_{mp}$

   b. Using the assumptions of part a, what is the average CPI for branch instructions (CPI_{br}) for the following designs:
      i) A pipeline using predict-not-taken branch prediction and no BTB, and a 3-cycle branch misprediction latency. On average 65% of all branches are taken on this design while running. **[3]**

CPI_{br}= **2.95**

      ii) A pipeline using a 256-entry Gshare branch predictor with a 92% accuracy, and a 512-entry BTB with a 82% target prediction accuracy, and a 10-cycle branch/BTB mispredict latency. **[3]**

CPI_{br}= **3.6**

7. Answer the following questions about the algorithm we have called T3 (or MIPS R10000) using what we have called back-pointers. **[8 points]**

   a. When does an instruction write to the PRF? **[1]**

   WHEN IT COMPLETES EXECUTION.

   b. When does an instruction free a PRF entry? How does it select which one to free? **[3]**

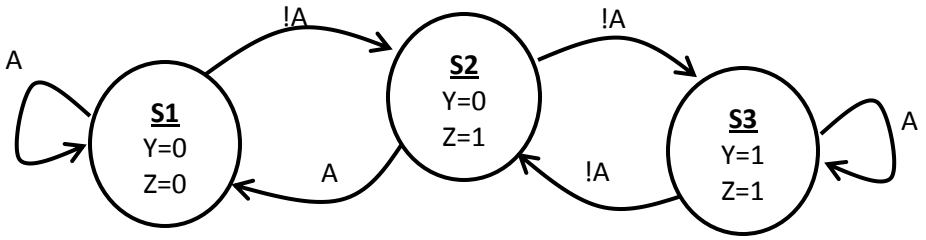
   A) WHEN IT COMMITS. IT FREES THE ENTRY THAT IT OVERWROTE IN THE RAT AT ISSUE (THE VALUE IN THE "BACK POINTER"). B) WHEN A MISPREDICTED BRANCH COMMITS IT FREES ALL PRF ENTRIES THAT DON'T END UP IN THE RAT AFTER THE RAT IS "FIXED" (SEE BELOW)

   WE TOOK ANSWERS THAT ONLY ADDRESSED "A".

   c. Explain exactly what would be involved in "fixing" the RAT once a mispredicted branch hits the head of the RoB. **[4]**

   YOU WALK THROUGH THE ROB FROM TAIL TO HEAD, PUTTING THE BACK POINTER VALUE INTO THE APPROPRIATE RAT ENTRY (WHERE IT CAME FROM—THE LOCATION INDEXED BY THE DESTINATION ARCHITECTED REGISTER OF THE CURRENT INSTRUCTION).

   YOU COULD INSTEAD STATE THAT YOU USE THE VALUE CLOSEST TO THE HEAD FOR EACH RAT ENTRY. THAT'S EQUIVALENT TO THE ABOVE.



8. Consider the above Moore-type state machine where A is an input and Y and Z are outputs. Without removing or changing any of the code below, you are to finish the implementation of this state machine in legal, and well written Verilog in a way that meets the EECS 470 programming style guidelines. *State S2 is the reset state, while S1 should be encoded as 2'b01 and S3 as state 2'b10.* **[8 points]**

```
module statemachine(A,clock,reset,Y,Z);
      input A, clock, reset;
      output wire Y,Z;
      reg [1:0] state, next_state;

      always @(posedge clock)
      begin
            if(reset)
                  state <= #1 2'b00;
            else
                  state <= #1 next_state;
      end
   `define S1 2'b01
   `define S2 2'b00
   `define S3 2'b10
   always@*
   begin
      if(state==`S2)
         next_state=(A ? `S1: `S3);
      else if(state==`S1)
         next_state=(A ? `S1: `S2);
      else
         next_state=(A ? `S3: `S2);
   end

   assign Y=(state== `S3);
   assign Z=(state!= `S1);

endmodule
```

COMMON ERRORS:

- USE ALWAYS BLOCK TO ASSIGN TO Y AND Z.
- NOT BE SURE NEXT STATE IS DEFINED ON ALL PATHS (EVEN THE "ILLEGAL STATE!).

9. Consider the following state of a machine implementing what we've called Tomasulo's third (or the R10K algorithm) with a retirement RAT.

RAT

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

ROB

Buffer Number	PC	Executed?	Dest. PRN	Dest ARN	
0	0	N	0	1	~~← HEAD~~
1	4	N	1	1	
2	8	Y	2	3	
3	12	Y	-	-	
4	16	N	3	1	
5	20	Y	9	2	
6	24	Y	10	4	~~← TAIL~~
7	80	N	11	2	← HEAD
8	84	N	12	0	← TAIL

RRAT

Arch Reg #	Phy. Reg #
0	8
1	~~7~~ 1
2	6
3	~~5~~ 2
4	4

RS

RS#	Op Type	Op1 Ready?	Op1 PRN/value	Op2 Ready?	Op2 PRN/value	Dest PRN	ROB
0	~~+~~	~~N~~	~~0~~	~~Y~~	~~4~~	~~3~~	~~4~~
1	+	Y	14	Y	4	11	7
2	+	N	11	Y	2	12	8
3	~~+~~	~~N~~	~~0~~	~~Y~~	~~2~~	~~1~~	~~1~~
4	~~*~~	~~Y~~	~~3~~	~~Y~~	~~4~~	~~0~~	~~0~~

PRF

Phy Reg #	Value	Free	Valid
0	23	~~N~~ Y	N
1	~~-1~~ 14	N	~~N~~ Y
2	4	N	Y
3	11	~~N~~ Y	N
4	4	N	Y
5	3	~~N~~ Y	~~Y~~ N
6	0	N	Y
7	1	~~N~~ Y	~~Y~~ N
8	2	N	Y
9	8	~~N~~ Y	~~Y~~ N
10	16	~~N~~ Y	~~Y~~ N
11	12	~~Y~~ N	N
12	14	~~Y~~ N	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.
- **Dest. ARN** is the destination Architectural Register Number.

Say that the instruction in ROB #3 is a branch and it was mis-predicted: the next PC should have been 80. Say that the instruction in memory location 80 is R2=R1+R4 and in 84 is R0=R2+R0. Update the machine to the state where the branch has left the RoB, and the instructions at memory 80 and 84 have issued but not completed execution. When faced with an arbitrary decision, just be sure to make a legal choice. ***Be sure to update the head and tail pointers!*** **[18]**

On the following page is an extra copy of this state. 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.

9. Consider the following state of a machine implementing what we've called Tomasulo's third (or the R10K algorithm) with a retirement RAT.

RAT

Arch Reg #	Phy. Reg #
0	8
1	3
2	9
3	2
4	10

ROB

Buffer Number	PC	Executed?	Dest. PRN	Dest ARN	
0	0	N	0	1	← HEAD
1	4	N	1	1	
2	8	Y	2	3	
3	12	Y	-	-	
4	16	N	3	1	
5	20	Y	9	2	
6	24	Y	10	4	← TAIL
7					
8					

RRAT

Arch Reg #	Phy. Reg #
0	8
1	7
2	6
3	5
4	4

RS

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

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.

PRF

Phy Reg #	Value	Free	Valid
0	23	N	N
1	-1	N	N
2	4	N	Y
3	11	N	N
4	4	N	Y
5	3	N	Y
6	0	N	Y
7	1	N	Y
8	2	N	Y
9	8	N	Y
10	16	N	Y
11	12	Y	N
12	14	Y	N

Say that the instruction in ROB #3 is a branch and it was mis-predicted: the next PC should have been 80. Say that the instruction in memory location 80 is R2=R1+R4 and in 84 is R0=R2+R0. Update the machine to the state where the branch has left the RoB, and the instructions at memory 80 and 84 have issued but not completed execution. When faced with an arbitrary decision, just be sure to make a legal choice. ***Be sure to update the head and tail pointers!*** **[18]**