EECS 470 Midterm Exam Answers
Fall 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:

<table>
<thead>
<tr>
<th>#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 2</td>
<td>/15</td>
</tr>
<tr>
<td>Page 3</td>
<td>/9</td>
</tr>
<tr>
<td>Page 4</td>
<td>/12</td>
</tr>
<tr>
<td>Page 5</td>
<td>/13</td>
</tr>
<tr>
<td>Page 6</td>
<td>/9</td>
</tr>
<tr>
<td>Page 7</td>
<td>/12</td>
</tr>
<tr>
<td>Page 8</td>
<td>/12</td>
</tr>
<tr>
<td>Page 9 &amp; 10</td>
<td>/18</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>

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 10 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 [15 points, -2 per wrong/blank, minimum 0]

   a. Given a 16KB, two-way associative cache with 16-byte lines, you will need _____ **9** bits to index the cache. If the cache started out entirely invalid, it would get **0 / 1 / 2 / 3 / 4** hits on the following address stream: 0x22222, 0x32223, 0x22220, 0x82228, 0x32220, 0x82220

   b. Which of the following is true:
      - **Power constrained computing refers to battery life limitations in mobile devices.**
      - **Register spills and fills are a result of not having enough physical registers**
      - **The Register Alias Table in the P6 scheme points to a RS entry**
      - **Gshare generally outperforms qselect if the predictor memory structures are the same size.**

   c. As you increase size of the RoB, you expect the CPI of the processor on a given workload to go **up / down**, while you expect the clock period to go **up / down**.

   d. Say you have an ISA where all instructions are 32-bits and which has 32 general purpose registers and all immediate values are 14-bits. If your instruction set consisted of nothing other than instructions that used two GPRs and one immediate, you could have up to **256 / 512 / 1024 / 2048 / 4096** instructions total in your ISA.

   e. In the R10K scheme, an instruction will write to the PRF when it is **committing / completing execution / dispatching**. In P6 scheme an instruction will write to the ARF when it is **committing / completing execution / dispatching**.

   f. The **wakeup** process can be best described as addressing **true dependencies / false dependencies / control hazards / structural hazards**. The **select** process can best be described as addressing **true dependencies / false dependencies / control hazards / structural hazards**.
2. Write a SystemVerilog module which implements the following schematic. You are to keep the signal names the same as they are in the provided figures. Your code should be reasonably efficient. Minor syntax errors will be ignored. [9 points]

```
module temp
(
    input Clk,
    input A, B, C, D,
    input [1:0] S,
    output X1,
    output logic X2
);

    wire mux1 = S[0] ? B : A;
    wire mux2 = S[1] ? D : C;

    assign X1 = mux1 & mux2;

    always_ff @(posedge Clk) begin
        X2 <= #1 X1;
    end
endmodule
```
3. Short answer [12 points]
   a. Consider the pipeline you were to implement for your third assignment, but assume that the structural hazard has been removed. A given program consists of 10% loads, 15% stores, 10% branches and the rest are ALU operations. If 40% of the branches are not-taken and 30% of all instructions are dependent on the instruction immediately before them, what is the expected CPI of the processor on this program? Show your work [4]

   \[ 1 + 0.6(0.1)(3) + 0.3(0.1)(1) = 1.21 \]

   b. Consider a local history predictor where the Branch History Table has 64 entries each 8 bits in length and the Pattern History Table consists of a standard 2-bit predictor. How many bits of memory are used in the BHT? The PHT? [4]

   \[
   \begin{array}{c}
   \text{# bits in the BHT} \quad 512 \\
   \text{# bits in the PHT} \quad 512 \\
   \end{array}
   \]

   c. Place a checkmark next each statements about instruction level parallelism (ILP) that is generally TRUE. [4 points, -2 wrong check/not checked line, minimum 0]

   ___\x___ Better branch predictors effectively expose additional ILP.

   ___\x___ Increasing the PRF size in the R10K scheme (while keeping the RoB and RS constant is unlikely to significantly expose any additional ILP

   ____ Increasing the RS size in the P6 scheme while keeping the RoB size constant is unlikely to significantly expose any additional ILP.

   ____ The 5-stage pipeline you did for project 3 will see no performance improvement on programs with significant ILP vs. those with nearly no ILP.
4. More short answer [13 points]
   a. Functions are surprisingly difficult for the branch predictors we’ve discussed to deal with. What is it about functions that cause problems for our predictors? Be specific about what part of prediction is difficult (direction, address) and what exactly it is about functions that cause problems. [5]

   Branch address predictors have problems with return statements from functions. A BTB predicts that a branch will go to wherever it branched to last time. But if a function is getting called from multiple places, the return will go to different places each time.

   b. McFarling’s “Combining Branch Predictors” paper discusses a local/gshare predictor. Explain what that is. [4]

   It consists of a local history predictor, a gshare predictor and a tournament selector that chooses between the two.

   c. In the R10K algorithm, explain exactly when and how the RAT is updated by a committing instruction [4]

   If the committing instruction is a mispredicted branch, the RRAT and its freelist are copied into the RAT and its freelist.

   If the committing instruction writes to a register, the PRF overwritten in the RRAT is freed in the RAT.
5. Say we have the following code segment in pseudo-assembly:

```
If(R1==0) goto SKIP
R5=R5/R3
R6=R6+1
If(R6==-1) goto SKIP
R5=R5+1
SKIP:
```

And say we’ve got the following instruction available to us: `CMOV (Rx, Ry, Rz)` Where the instruction sets Ry=Rz if and only if Rx! = 0. [9 points]

a. Rewrite the above code to remove all branches. You may only use R7, R8, and R9 as “scratch” registers: all other registers are holding live values. For comparisons you may use instructions such as R1=(R5==R6) or R1=(R5>0). You may not read or write memory. [5]

//If R1==0, we divide by 1 and do nothing.
R7=1
CMOV(R1, R7, R3)  //if(R1==0, R7=1, else R7=R3)
R7=R5/R7
CMOV(R1,R5,R7)   //if(R1==0, R5 unchanged, else R5=R7)
R7=R6+1
CMOV(R1,R6,R7)   //if(R1==0, R6 unchanged, else R6=R6+1
R7=R6+1
R7=R1 & R7       //if R1==0 or R6=-1 we should do nothing.
R8=R5+1
CMOV(R7,R5,R8)

b. Under what circumstances would you expect removing the above branches and replacing them with CMOVs would improve performance? [4]

Most simply when the branches are predicted poorly and the penalty for a branch is fairly high. More specifically, the CMOV code will always take 10 instructions, so when the average pass through the original code has more than 5 stalls.
6. Consider a non-superscalar processor implementing the R10K algorithm. This processor has 32 architected registers, 64 RoB entries, 96 physical registers and 16 reservation stations. List all input and output that will be used to implement the RRAT by filing in the table below. [12 points]

<table>
<thead>
<tr>
<th>Description of signal</th>
<th>Input/Output</th>
<th>Why it’s needed</th>
<th>How many bits?</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>input</td>
<td>To maintain state</td>
<td>1</td>
</tr>
<tr>
<td>reset</td>
<td>input</td>
<td>To clear all entries</td>
<td>1</td>
</tr>
<tr>
<td>inst_commit</td>
<td>input</td>
<td>Valid when instruction is committing from the head of the ROB</td>
<td>1</td>
</tr>
<tr>
<td>inst_arch_reg</td>
<td>input</td>
<td>The architected register index of the committing instruction</td>
<td>5</td>
</tr>
<tr>
<td>inst_phys_reg</td>
<td>Input</td>
<td>The physical register index of the committing instruction</td>
<td>7</td>
</tr>
<tr>
<td>prf_invalidate_valid</td>
<td>output</td>
<td>To tell PRF to free the physical register that has been overwritten</td>
<td>1</td>
</tr>
<tr>
<td>prf_invalidate_index</td>
<td>output</td>
<td>Physical register index of the overwritten register</td>
<td>7</td>
</tr>
<tr>
<td>phys_reg_idx_array</td>
<td>output</td>
<td>The current state of the RRAT, for the RAT to read in the event of mispredict</td>
<td>7*32</td>
</tr>
</tbody>
</table>

There were a few other reasonable answer here. But this is probably the best.
7. Consider the following pseudo assembly code

\[
\begin{align*}
  r2 &= 0 \\
  r4 &= 0 \\
  r5 &= 0 \\
  \text{bob:} & \quad r3 = (r2 \% 2) \\
  & \quad \text{if}(r3 == 0) \text{goto next} // \text{Branch 1} \\
  & \quad r4 = r4 + 1 \\
  \text{next:} & \quad r5 = r5 + 4 \\
  & \quad r2 = \text{mem}[r5 + 0] // \text{Load} \\
  & \quad \text{if}(r4 < 500000000) \text{goto bob} // \text{Branch 2}
\end{align*}
\]

“bob” has an address of 0x1008. The predictors will use the least significant bits of the PC other than the word-offset. Predictors are all initialized to not taken or strongly not taken in the case of bimodal.

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

- **Case 1**: The data from the load is even 5 times in a row and then odd once, then even 5 times in a row and then odd once, etc.
- **Case 2**: The data from the load is odd 2 times in a row, then even 2 times in a row, then odd 2 times in a row, then even 2 times in a row, etc.
- **Case 3**: The data from the load is odd 3 times in a row and then even, then odd 3 times in a row, then even, etc.

You are now to consider 3 branch predictors

- **Predictor 1**: A PC-based predictor with 4 entries each 1 bit in size.
- **Predictor 2**: A global history predictor. The branch history register has 3 bits of history. The predictors are each 2-bit saturating counters.
- **Predictor 3**: A local pattern history predictor. The BHT has 16 entries each with 3 bits of history. The predictors are each 2-bit saturating counters.

What are the expected prediction rates for each of the following? Your answer must be correct within 0.5%.

[12 points, -1 per blank or wrong entry, minimum 0]

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Branch 1</td>
<td>Branch 2</td>
<td>Branch 1</td>
</tr>
<tr>
<td>Predictor 1</td>
<td>5/6</td>
<td>5/6</td>
<td>½</td>
</tr>
<tr>
<td>Predictor 2</td>
<td>5/6</td>
<td>1</td>
<td>½</td>
</tr>
<tr>
<td>Predictor 3</td>
<td>5/6</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
8. Consider the following state of a machine implementing what we’ve called the R10K algorithm with a retirement RAT.

<table>
<thead>
<tr>
<th>RAT</th>
<th>ROB</th>
<th>RRAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch Reg #</td>
<td>Phy. Reg #</td>
<td>Buffer Number</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>N</td>
</tr>
</tbody>
</table>

- **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 60. Say that the instruction in memory location 60 is R2=R1+R4 and in 64 is R0=R0+R2. Update the machine to the state where the branch has left the RoB, and the instructions at memory 60 and 64 have dispatched but not started execution. When selecting a PRF use the lowest numbered physical register available, otherwise when making an arbitrary decision, just be sure it is legal. Be sure to update the head and tail pointers! [18]