# EECS 470 *Midterm Exam Answers*

Winter 2018

Name: _____ unique name: _____

Sign the honor code:

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

_____

---

Scores:

## 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 ***9*** 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" for the last question. Clearly mark which one you want graded or we will grade the first one.**

1. Fill-in-the-blank or circle the best answer **[17 points, -2 per wrong/blank, minimum 0]**

   a. When using the R10K algorithm, if you have B reorder buffer entries, S reservation stations, and A architected registers you can be sure that you will not have a use for more than ***B+S / B+A / A+S*** physical registers.

   b. In the P6 scheme an instruction might find its data in one of three places. Those three places are __**ARF**_____, _____**CDB**_____, or _____**ROB**____. In the R10K scheme there are ***two / three / four*** places an instruction might find its data.

   c. Given an 8-KB four-way associative cache with 16-byte cache lines and a 32-bit address space there will be __***7***___ bits used for the index. If that same cache were fully-associative you'd need __***0***____ bits to be used for the index.

   d. The BTB generally does a poor job with predicting the target address of a ***function call / function return / PC-relative branch***.

   e. A cylindrical metal wire which is 1mm long and has a diameter of 10nm will have the ***higher / lower / the same*** resistance as a wire made of the same material which is 2mm long and has a diameter of 20nm.

   f. Adding more reservation stations will reduce the amount of time that the processor stalls due to ***structural hazards / data hazards / control hazards*** but may decrease the ***number of branch delay slots / clock period / clock frequency***.

   g. In the R10K scheme, if you have 32 RoB entries, 16 architected registers, 6 reservation stations and 48 PRFs, you should never have a structural hazard due to the lack of a ***PRF entry / reservation station / RoB entry / floating point divide.***

2. Consider the pipelined processor you did as part of project 3 (including the structural hazard). Say 20% of all instructions were loads, 25% were branches, and 10% were stores and that the program was quite long (millions and millions of instructions). 20% of all instructions are data dependent on the instruction in front of them, and branches are taken 60% of the time. What would you expect the CPI to be? Show your work. **[6 points]**

   **1+dependent load stalls+structural stalls+control hazards**
   **Dependent Load stalls=(.2)(1)(.2)**
   **Structural hazard stalls =.3(1)**
   **Control hazards = .25 * .6 (3)**
   **CPI is then 1.79**

3. Answer each of the following questions in 40 words or less. **[11 points]**
   a. Why don't we let stores write to memory as soon as their address and data are known? **[3]**

      **Because we don't want to modify architectural state speculatively.**

   b. In the P6 scheme, under what circumstances does a committing instruction update the RAT? Be precise and be sure to consider all types of instructions. **[4]**

      - **When an instruction that wrote an architected register commits and the RAT points to its ROB entry.**
      - **When a mispredicted branch commits**

4. Consider a set of code where there are three 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 50 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 keeps an instruction in its RS until that instruction has finished executing. The machine has an RS size of 12 and a RoB size of 64 unless otherwise noted. 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? **[3]**

      **50/12 =4.1667.**

   b. What is the best CPI this machine could achieve if the program being run consisted of only "*short*" instructions? **[3]**

      **1**

   c. What is the best CPI this machine could achieve if the program being run consisted of only "*dependent*" instructions? **[3]**

      **4**

   d. What is the best CPI this machine could achieve if the program being run consisted of groups of 50 instructions, where the first 49 were "dependent" and the last was "short"? Assume there are a large number of these groups. **[3]**

      **(50*4-11*4)/50 = 39*4/50=3.121**

5. Consider the following pseudo-assembly code:

```
       r2=0
       r4=0
       r5=0
 bob:  r3=(r2 mod 3)   // remainder when r2 is divided by 3
       if(r3==0) goto next       // Branch 1
       r6=r6+1
next:  r5=r5+4
       r2=MEM[r5+0]              // Load
       if(r6<1000000) 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. Global history is initialized as if all branches had been not taken.

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 random. That is it could be any value that fits in a 32-bit integer with equal probability.

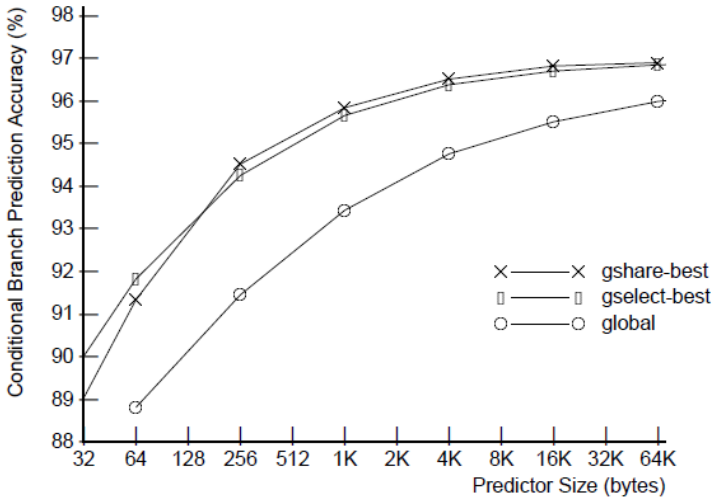
You are now to consider 3 branch predictors:

- **Predictor 1:** A bimodal predictor with 4 entries each a 2-bit saturating counter.
- **Predictor 2:** A bimodal predictor with 8 entries each 1 bit.
- **Predictor 3:** A Gshare predictor with 4 entries each 1 bit.

What are the expected *mispredict rates* for each of the following? You only need to worry about steady state values (assume 1000000 is a really big number.)
**[15 points, -1.5 per wrong or blank box, min 0]**

| | **Case 1** | | **Case 2** | |
|---|---|---|---|---|
| | *Branch 1* | *Branch 2* | *Branch 1* | *Branch 2* |
| **Predictor 1** | **2/3** | **0** | **2/3** | **0** |
| **Predictor 2** | **2/3** | **0** | **4/9** | **0** |
| **Predictor 3** | **1** | **1/3** | **14/27** | **2/9** |

6. The graph to the right comes from the "Combining Branch Predictors" paper that was assigned as a reading for the course. In your own words, describe what "best" means in the context of both gselect-best and gshare-best. **[7 points]**




a. "gselect-best" **[3]**

For gselect you need to choose how many bits to use from the PC and how many from the global history. Gselect-best is the result for whatever choice maximizes performance.

b. "gshare-best" **[4]**

In this case we may choose to not use all the global history bits. Gshare-best is the result of using as few or many of those bits as to maximize performance.

7. MOV instructions in modern x86 architectures can often be implemented without actually writing the value from one register to another. Instead, the Map Table simply renames the architectural register of the destination to the same physical register as the source at dispatch. This is often called Move Elimination. Move Elimination effectively takes the MOV instruction out of the execution pipeline as it doesn't need to be issued. For this question, assume you are implementing this feature on top of an R10k microarchitecture. **[13 points]**

   a. Name one advantage to this technique other than speeding up the MOV instruction. **[2]**

   - Doesn't require an RS.
   - Lower power (less copies)
   - Less on the CDB

   b. Does the MOV instruction need to broadcast on the CDB? Why or why not? **[5]**

   No need to broadcast because MOV is always resolved before the dependent instructions. Only the instruction that handles the source register will have to broadcast.

   c. Consider the following code running on a processor where Move Elimination has been implemented. When can you free the physical register that is assigned to $r0 in the first instruction? Explain your answer. **[6]**

```
ADD $r4 $r2 $r0        //r0 = r4 + r2
MOV $r0 $r1            //r1 = r0
MOV $r1 $r13           //r13 = r1
ADD $r9 $r1 $r1        //r1 = r9 + r1
ADD $r6 $r0 $r0        //r0 = r6 + r0
MUL $r13 $r3 $r7       //r7 = r13 * r3
ADD $r0 $r1 $r13       //r13 = r0 + r1
ADD $r0 $r13 $r1       //r1 = r0 + r13
MUL $r7 $r1 $r0        //r0 = r7 * r1
```

When ADD $r0 $r1 $r13 retires, because all architected registers that are dependent on the initial mapping of $r0 in the first instruction have been overwritten so the value is no longer needed..

8. Consider the following state of a machine implementing what we've called Tomasulo's third algorithm. **[19 points]**

**RAT**

| Arch Reg # | Phy. Reg # |
|---|---|
| 0 | 7 |
| 1 | 6 |
| 2 | ~~3~~ 11 |
| 3 | 4 |
| 4 | ~~9~~ ~~8~~ 5 |

**ROB**

| Buffer Number | PC | Executed? | Dest ARN | Dest. PRN | Replaced Phy. Reg# (Back pointer) |
|---|---|---|---|---|---|
| 0 | ~~12~~ | ~~N~~ | ~~0~~ | ~~0~~ | ~~1~~ |
| 1 | ~~16~~ | ~~Y~~ | ~~1~~ | ~~6~~ | ~~2~~ |
| 2 | ~~20~~ | ~~N~~ | ~~0~~ | ~~7~~ | ~~0~~ |
| 3 | ~~24~~ | ~~Y~~ | ~~--~~ | ~~--~~ | ~~--~~ |
| 4 | ~~80~~ | ~~N~~ | ~~4~~ | ~~8~~ | ~~5~~ |
| 5 | ~~84~~ | ~~Y~~ | ~~4~~ | ~~9~~ | ~~8~~ |
| 6 | 28 | N | 2 | 12 | 3 |
| 7 | 32 | N | 2 | 11 | 12 |
| 8 | | | | | |

Initial Head=0

Initial Tail=6

Final Head= 6

Final Tail= 8

**RS**

| RS# | Op Type | Op1 Ready? | Op1 PRN/value | Op2 Ready? | Op2 PRN/value | Dest PRN | ROB |
|---|---|---|---|---|---|---|---|
| 0 | ~~Add~~ | ~~Y~~ | ~~1~~ | ~~Y~~ | ~~2~~ | ~~0~~ | ~~0~~ |
| 1 | ~~Mult~~ | ~~N~~ | ~~7~~ | ~~Y~~ | ~~4~~ | ~~8~~ | ~~4~~ |
| 2 | ~~Add~~ | ~~N~~ | ~~0~~ | ~~Y~~ | ~~2~~ | ~~7~~ | ~~2~~ |
| 3 | Add | Y | 3 | Y | 6 | 12 | 6 |
| 4 | Add | Y | 6 | N | 12 | 11 | 7 |

**PRF**

| Phy Reg # | Value | Free | Valid |
|---|---|---|---|
| 0 | ~~1~~ 1 | ~~N~~ Y | ~~N~~ ~~Y~~ N |
| 1 | 2 | ~~N~~ Y | ~~Y~~ N |
| 2 | -1 | ~~N~~ Y | ~~Y~~ N |
| 3 | 4 | N | Y |
| k4 | 5 | N | Y |
| 5 | 6 | N | Y |
| 6 | 20 | N | Y |
| 7 | ~~8~~ 3 | N | ~~N~~ Y |
| 8 | 9 | ~~N~~ Y | N |
| 9 | 10 | ~~N~~ Y | ~~Y~~ N |
| 10 | 11 | Y | N |
| 11 | 12 | ~~Y~~ N | N |
| 12 | 13 | ~~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.
- **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, *No register should be valid and free!*

a. Say that the instruction in ROB #3 is a branch and it was mis-predicted: the next PC should have been 28. Say that the instruction in memory location 28 is R2=R0+R4 and in 32 is R2=R4+R2. Update the machine to the state where the branch has left the RoB, and the instructions at memory locations 28 and 32 have dispatched but not executed. When allocating a new PRF entry, pick the *highest* number available. Otherwise, when faced with an arbitrary decision, just be sure to make a legal choice. **[16]**

b. What was the assembly instruction that is found at memory location 80? Your answer should be of the form:"r1=r2+r3" or something similar. *Your answer may not include an immediate*. **[3]**

R4=R0*R2

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. *Also, be sure to answer the following question.*