EECS 470 Winter '24

Homework 3

Due Wednesday 2/14 at 10pm – No late homework accepted.

Please note that you will not get this back in time for the exam. We'll post answers almost immediately.

Name: _____ unique name: _____

Assignments that are difficult to read after scanning will lose *at least* 50% of the possible points and we may not grade them at all. This assignment is worth a bit more than 1% of your grade in the class and is graded out of 100 points.

---

1. Say we have a BTB used in parallel with a tournament branch predictor. The BTB is tagged (so no aliasing) while the predictor is not tagged at all. During the fetch stage, say we encounter a branch that is in the BTB and is predicted taken by both the 2-bit local and 2-bit global predictors of the tournament predictor. If the branch is actually taken which of the following might be changed? Explain why each may be changed (and if so, when) or why it would never change. **[14]**
   a. BTB target address
   b. BTB tag
   c. Local branch predictor
   d. Local branch history table
   e. Global branch predictor
   f. Tournament predictor
   g. Global history register

2. Consider the following statement:
   > "Because of register renaming there is much less of a need for having a large set of general-purpose registers – the physical register file can be much larger than the architected register file and not have any impact on the ISA."

   Explain when having a larger set of architected registers can help but having a larger set of physical registers would not. **[12]**

3. Consider the following programs

Program A	**Program B**
R1=MEM[R2+0]	R1=MEM[R2+0]
R2=R4+4	R2=R1+4
R3=R2+R7	R3=R4+R4
R4=R1+6	R4=R2+6
R1=R2+R3	R1=R2+R3
R6=R2+R6	R6=R9+R10
R7=R6+19	R7=R3+19
R8=R3+R6	R8=R9+R3

   Say we have an out-of-order machine (Using the P6 scheme) with 3 RSs and 6 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 is issued to its functional unit. Be sure to clearly show/explain your work. **[16]**

4. Consider the following tables that represent the state of a processor that implements what we have called the P6 scheme.

RAT	
Arch Reg. #	**ROB# (-- if in ARF)**
0	--
1	--
2	--
3	--
4	--
5	--

ROB				
Buffer Number	**PC**	**Done with EX?**	**Dest. Arch Reg #**	**Value**
0				
1				
2				
3				
4				
5				
6				
7				
8				

RS						
RS#	**Op type**	**Op1 ready?**	**Op1 RoB/value**	**Op2 ready?**	**Op2 RoB/value**	**Dest ROB**
0						
1						
2						
3						
4						

ARF							
	Reg#	**0**	**1**	**2**	**3**	**4**	**5**
	Value	6	5	4	3	2	1

Place the following instructions into the processor:

```
R3=R2+R1        // A PC=40, each instruction is 4 bytes.
R1=R1*R3        // B
R3=R1+R3        // C
R2=R4*R2        // D
R5=R3+R4        // E
```

Show the state of the above tables if instruction A has retired, instruction B has not finished executing, while instructions C, D and E have progressed as far along as possible given the state of A and B. *Be sure to label the head and tail of the ROB.* Please place instruction A in slot 0 of the ROB and B in slot 1. When arbitrary decisions need to be made, you are to just make them. ***Clearly cross out any data that is no longer valid.*** **[20]**

5. Consider a gshare predictor where:
   - The global history to date is 1010, where ***the left-most bit is the most recent***.
   - The instructions are 32-bits long each.
   - Memory is byte-addressed.
   - The address bits used are the four least significant bits of the PC other than the word offset and they are used with the LSB on the right.
   - The underlying predictor is a two-bit up/down saturating counter (00 is strongly not taken).
   - Updates to the predictor and global history are done immediately after the branch is predicted and the update is with the "right" answer. (Yes, this is almost impossible.)
   - The current predictor table is:

Index	Predictor	Index	Predictor
0	11	8	00
1	11	9	10
2	00	10	11
3	11	11	10
4	01	12	11
5	00	13	00
6	10	14	01
7	01	15	01

   - The code being run is:

```
A:  R1=R1+R2
    If(R2<=3) goto C
    R3=R3-1
C:  If(R3<=0) goto D
    goto A
D:  halt
```

   For the above pseudo-assembly code assume that:
   - The label "A" is at address 0x100.
   - The first conditional branch is always **taken**.
   - The second conditional branch is **not taken** the first time it is encountered, and **taken** each time thereafter.

   Update the above table to the state it will be in when the halt instruction is encountered. You should assume unconditional branches are neither predicted nor do they influence the predictor. **[20]**

6. Multiple choice/fill in the blank. *Select the best answer.*
   [18 points, -2 per wrong or blank answer, minimum 0]

   a. In the algorithm we are calling R10K, when using an RRAT 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.

   b. If, in the algorithm we are calling P6, the RAT had only one port for writing values, you could only have one instruction that writes a register ***dispatch / issue / complete execution / retire*** per cycle.

   c. If, in the algorithm we are calling R10K, the PRF had only one write port, you could only have one instruction that writes a register ***dispatch / issue / complete execution / retire*** per cycle.

   d. In the original Tomaulo's algorithm, the RAT points to ***a reorder buffer entry / a reservation station / a physical register / an execution unit***.

e. The branch target buffer is a(n) ***address / confidence / direction*** predictor that has particular problems with ***function calls / conditional branches / returns from functions / indirect branches***. To help with that case we might add a(n) ***Translation Lookaside Buffer (TLB) / Reorder Buffer (RoB) / Alias Table List Address Selector (ATLAS) / Return Address Stack (RAS)***.

f. In the algorithm we are calling R10K, if you have 32 RoB entries, 16 architected registers, 8 RS entries and a 4KB instruction-cache, you will not have a use for more than ***8 / 16 / 32 / 48 / 64*** PRF entries.

g. Given a 4-KB four-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. In the P6 algorithm, a processor with 2 RSes, 16 RoB entries, and 8 architected registers would have a RAT whose size in bits is ***10 / 16 / 20 / 32 / 40 / 96 / 128*** ignoring state bits (like valid etc.). If that same design were using the algorithm we've called R10K, the RAT would have ***10 / 16 / 20 / 32 / 40 / 96 / 128*** bits (again ignoring state bits).