CS 305
Autumn 2016
Quiz 3
Oct 5, 2016
Time Limit: 90 Minutes

IIT Roll No.: Umesh Bellur

This exam contains 6 pages (including this cover page) and 7 problems. Check to see if any pages are missing. **Enter all requested information on the top of this page, and put your IIT Roll Number clearly on the top of every page, in case the pages become separated.**

You may *not* use any books or a calculator on this exam. The last page of this paper has the MIPS instructions for your reference.

You are required to show your work on each problem on this exam. The following rules apply:

- **If you use any well known results you must indicate which one.**
- **Organize your work,** in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.
- **Mysterious or unsupported answers will not receive full credit.** A correct answer, unsupported by calculations, explanation, or algebraic work will receive no credit; an incorrect answer supported by substantially correct calculations and explanations will still receive partial credit.

Do not write in the table to the right.

Ques	Points	Score
1	2	
2	3	
3	3	
4	8	
5	8	
6	6	
7	4	
Total:	34	

1. (2 points) What is needed to resolve the following hazard without stalling the pipeline?

```
xor R12, R13, R12
ld R2, R3 #load contents of R3 to R2.
add R13, R4, R12
```

Mem/WB forwarding for R12 from the XOR to the Add instruction

No partial

2. (3 points) What is the logic to check if we need to forward from MEM to EX (Mem/WB forwarding)? (Rs, Rt are the sources and Rd is the destination.)

$$\text{If } ((\text{Mem/WB.RegWrite} = 1) \wedge \xrightarrow{} 0.5$$
$$(\text{Mem/WB.Rd} = \text{ID/EX.Rs}) \wedge \xrightarrow{} 1$$
$$((\text{EX/Mem.Rd} \neq \text{ID/EX.Rs}) \vee \xrightarrow{} 1$$
$$(\text{EX/Mem.RegWrite} = 0)) \xrightarrow{} 0.5$$
$$)$$

3. (3 points) What is the slowdown for processors that does one instruction per cycle and has a branch predictor that is 85% accurate with a 75-cycle misprediction cost? (Assume 35% of instructions are branches). Compute slowdown relative to perfect prediction.

for perfect prediction (No stalls or other dependencies)

$\#$ of cycles = $\#$ of Instructions = IC

for our situation

$= 0.65 * IC + 0.35 * IC (0.85 \times 1 + 0.15 \times 75)$

$= 0.65 * IC + IC * (0.35 \times 0.85 + 0.35 \times 0.15 \times 75)$

4. Consider this sequence of MIPS instructions.

```
a) lw $t1, 40($t2)
b) add $t2, $t3, $a0
c) add $t1, $t1, $t2
d) sw $t1, 20($t2)
```

(a) (2 points) Identify all of the data dependencies in the above instructions, whether or not they cause any hazards or stalls. The instructions are labeled (*a*) – (*d*), which you must use in your answer where you should state where the value was generated and where it is needed.

0.5 ← (a) generates $t1 used in (c)
1 ← (b) " $t2 " in (c) & (d)
0.5 ← (c) " $t1 " " (d)

(b) (3 points) Suppose we execute these instructions on a processor with a 5-stage pipeline with forwarding as described in class. Are there any hazards in the above sequence of instructions that will require the pipeline to stall because they can't be handled by the forwarding mechanisms? If so, where are they and why is the stall needed?

No Stalls needed with ~~full~~ forwarding

No partial

(c) (3 points) Now lets assume we're executing these instructions on a new, prototype processor, also with our usual 5-stage pipeline. Unfortunately in this new processor the hazard circuitry is completely broken and it does not detect hazards or properly forward results or insert stalls when necessary. We need to modify the code and insert nop instructions to delay the execution of later instructions when necessary. Re-write the above code and insert the **minimum number of nop instructions** needed to avoid hazards that would otherwise require forwarding or stalls to produce the correct results. You may not reorder the original instructions ? just insert nops where they are needed. (To insert a nop, just write nop on a line by itself. The assembler will know how to translate that into a machine instruction that does nothing except delay for a cycle.)

(a)
(b)
nop
nop } → 1.5
(c)
nop
nop } → 1.5
(d)

~~±4~~

5. Pipeline performance. Suppose we have the following functional units with the given latencies in a processor: IF - 2ns, ID - 2ns, EX - 3ns, MEM - 6ns, WB - 2ns.

   (a) (1 point) If we use these units to build a single-cycle implementation, how long (how many ns) does it take to execute a single instruction?

   $\Sigma = 15$ ns

   (b) (1 point) If we use these units to build our usual 5-stage pipeline processor, what is the shortest possible clock period?

   6 ns

   (c) (2 points) How many nanoseconds does it take to execute N instructions using this pipeline, where N is some arbitrary large number?

   $6 * (4 + N)$ ↳ cycles to fill

   No partial pts

   (d) (2 points) What is the speedup or slowdown of this pipelined processor over the single-cycle implementation for one instruction? What is the speedup in execution time for N instructions?

   1 ignoring fill delay 1 ins/6 ns vs 1 ins/15 ns or Speedup = 15/6 = 2.5

   1 For only 1 inst - Slowdown = 2

   (e) (2 points) How does the speedup calculated in the previous question (for N instructions) compare with the maximum speedup we could get from a 5-stage pipeline implementation? If it is not as good as it might be, explain what the problem is and suggest what might be done to improve it.

   1 { Max Speedup in throughput → 5 for a 5 stage pipeline (when balanced perfectly).

   1 { The 'mem' stage could be split into 2 3 ns stages to balance it

6. (6 points) Given the following MIPS code:

```
a  LOOP:  lw $t1, 0($a0) // 1
b  lw $a0, 0($t1) // 2
c  addi $a1, $a1, -1 // 3
d  bne $a1, $zero, LOOP // 4
e  add $v0, $a0, $zero // 5
f  addi $sp, $sp, 8 // 6
```

Assume that the pipeline processor implements ONLY Ex/Mem forwarding and a predict not-taken branch predictor. Draw the pipeline diagram for the code assuming you start out with $\$a1 = 2$. Include instructions, if any, that are fetched incorrectly. Use the letters F, D, E, M, and W to represent pipe stages Fetch, Decode, Ex, Mem and Writeback, S for stall and X for the noop that replaces an instruction when it's squashed (for all its remaining stages).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a	F	D	E	M	W			1												
b		F	D	S	S	E	M	W												
c			F	S	S	D	E	M	W											
d	1			S	S	F	D	E	M	W			1							
e	1				S	S	F	D	X	X	X									
f								F	X	X	X	X								
a									F	D	E	M	W		~~1~~					
b										F	D	S	S	E	M	W				
c											F	S	S	D	E	M	W			
d												S	S	F	D	E	M	W		
e									1				S	S	F	D	E	M	W	
f														S	S	F	D	E	M	W

~~[illegible]~~ 20 cycles

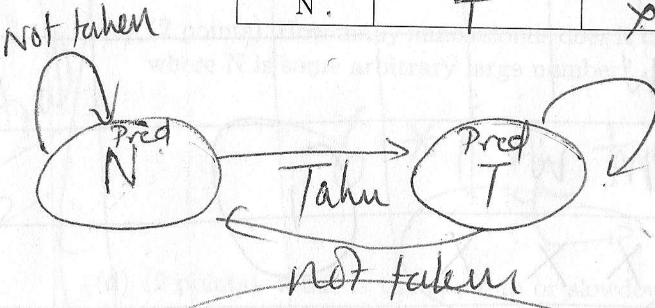
7. (4 points) One particular branch (i.e., one specific PC) has the following actual outcomes: T, T, T, N, T, N, T, T, T, N. Show the predictions for both a one-bit counter and a two-bit counter. For the two-bit counter, use T for strongly taken, t for weakly taken, n for weakly not-taken, and N for strongly not taken. Finally fill in the accuracy (percentage of predictions that were correct) at the bottom of the table. The first prediction is done for you.

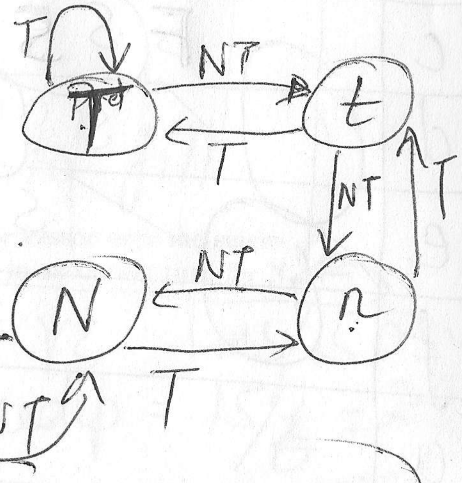
1.5

1.5

2

Outcome	1 Bit Predictor	Correct?	2 Bit Predictor	Correct?	State
T	N	no X	n	no	t
T	T	y	t	y	T
T	T	y	T	y	T
N	T	~~y~~ no	T	no	t
T	N	X	t	y	T
N	T	X	T	no	t
T	N	X	t	✓	T
T	T	✓	T	✓	T
T	T	✓	T	✓	T
N	T	X	T	no	t





accuray = 4/10 = 40%

0.5

accurey = 6/10 = 60%

0.5