CMPE 110 Computer Architecture
Fall 2015, Homework #2 Solution

Computer Engineering
UC Santa Cruz

November 3, 2015

Question 1. Pipelining Hazards (18 points)

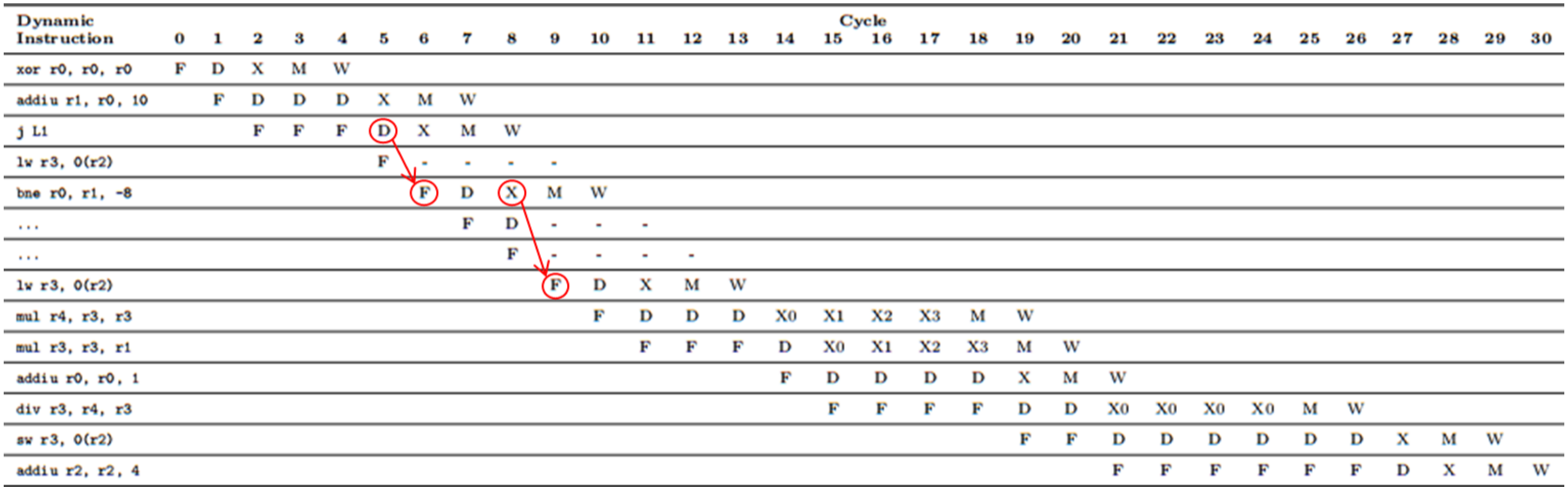
Solution.

Question 1.A Stalling for Structural Hazards (6 points)

See the pipeline diagram on the next page.

CPI Calculation:

- Before loop begins, instructions 1-3 (3 instructions) are called, taking 10 cycles total (C0-C9).
- After loop begins, a single iteration of the loop (instructions 4-11, 8 instructions) adds 21 cycles (C10-C30) to the pipeline.
- All 10 iterations of the loop would require $10 \times 21 = 210$ cycles.
- When the loop ends, the `bne` instruction is executed once more, adding 1 cycle.
- Add the above three points together and we get $10 + 210 + 1 = 221$ cycles for the entire program.
- The program has a total of $3 + 10 \times 8 + 1 = 84$ instructions executed.
- CPI $= \frac{221\ cycles}{84\ instructions} = 2.63$ cycles/instruction



Dynamic																Cycle															
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
`xor r0, r0, r0`	F	D	X	M	W																										
`addiu r1, r0, 10`		F	D	D	D	X	M	W																							
`j L1`			F	F	F	D	X	M	W																						
`lw r3, 0(r2)`						F	-	-	-	-																					
`bne r0, r1, -8`							F	D	X	M	W																				
...								F	D	-	-	-																			
...									F	-	-	-	-																		
`lw r3, 0(r2)`										F	D	X	M	W																	
`mul r4, r3, r3`											F	D	D	D	X0	X1	X2	X3	M	W											
`mul r3, r3, r1`												F	F	F	D	X0	X1	X2	X3	M	W										
`addiu r0, r0, 1`															F	D	D	D	D	X	M	W									
`div r3, r4, r3`																F	F	F	F	D	D	X0	X0	X0	X0	M	W				
`sw r3, 0(r2)`																				F	F	D	D	D	D	D	D	X	M	W	
`addiu r2, r2, 4`																						F	F	F	F	F	F	D	X	M	W

Question 1.B Bypassing for Data Hazards (6 points)

See the pipeline diagram on the next page.

CPI Calculation:

- Before loop begins, instructions 1-3 (3 instructions) are called, taking 8 cycles total (C0-C7).
- After loop begins, a single iteration of the loop (instructions 4-11, 8 instructions) adds 17 cycles (C8-C24) to the pipeline.
- All 10 iterations of the loop would require $10 \times 17 = 170$ cycles.
- When the loop ends, the `bne` instruction is executed once more, adding 1 cycle.
- Add the above three points together and we get $8 + 170 + 1 = 179$ cycles for the entire program.
- The program has a total of $3 + 10 \times 8 + 1 = 84$ instructions executed.
- CPI $= \frac{179\ cycles}{84\ instructions} = 2.13$ cycles/instruction

Dynamic	Cycle																								
Instruction	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
`xor r0, r0, r0`	F	D	X	M	W																				
`addiu r1, r0, 10`		F	D	X	M	W																			
`j L1`			F	D	X	M	W																		
`lw r3, 0(r2)`				F	-	-	-	-																	
`bne r0, r1, -8`					F	D	X	M	W																
`...`						F	D	-	-	-															
`...`							F	-	-	-	-														
`lw r3, 0(r2)`								F	D	X	M	W													
`mul r4, r3, r3`									F	D	D	X0	X1	X2	X3	M	W								
`mul r3, r3, r1`										F	F	D	X0	X1	X2	X3	M	W							
`addiu r0, r0, 1`												F	D	D	D	D	X	M	W						
`div r3, r4, r3`													F	F	F	F	D	X0	X0	X0	X0	M	W		
`sw r3, 0(r2)`																	F	D	D	D	D	X	M	W	
`addiu r2, r2, 4`																		F	F	F	F	D	X	M	W

Question 1.C Compiler Optimization for Control Hazards (6 points)

New re-ordered code:

```
1       xor r0, r0, r0
3       j L1
2       addiu r1, r0, 10
5 loop: mul r4, r3, r3
6       mul r3, r3, r1
7       addiu r0, r0, 1
8       div r3, r4, r3
9       sw r3, 0(r2)
10      addiu r2, r2, 4
11 L1:  bne r0, r1, -7
4       lw r3, 0(r2)
```

See the pipeline diagram on the next page.

CPI Calculation:

- Before loop begins, instructions 1-3 (3 instructions) are called, taking 7 cycles total (C0-C6).
- After loop begins, a single iteration of the loop (instructions 4-11, 8 instructions) adds 15 cycles (C7-C21) to the pipeline.
- All 10 iterations of the loop would require $10 \times 15 = 150$ cycles.
- When the loop ends, the `bne` and `lw` instructions are executed once more, adding 2 cycles.
- Add the above three points together and we get $7 + 150 + 2 = 159$ cycles for the entire program.
- The program has a total of $3 + 10 \times 8 + 2 = 85$ instructions executed.
- CPI $= \frac{159\ cycles}{85\ instructions} = 1.87$ cycles/instruction

Dynamic	Cycle																					
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
`xor r0, r0, r0`	F	D	X	M	W																	
`j L1`		F	D	X	M	W																
`addiu r1, r0, 10`			F	D	X	M	W															
`bne r0, r1, -7`				F	D	X	M	W														
`lw r3, 0(r2)`					F	D	X	M	W													
`...`						F	-	-	-	-												
`mul r4, r3, r3`							F	D	X0	X1	X2	X3	M	W								
`mul r3, r3, r1`								F	D	X0	X1	X2	X3	M	W							
`addiu r0, r0, 1`									F	D	D	D	D	X	M	W						
`div r3, r4, r3`										F	F	F	F	D	X0	X0	X0	X0	M	W		
`sw r3, 0(r2)`														F	D	D	D	D	X	M	W	
`addiu r2, r2, 4`															F	F	F	F	D	X	M	W

Question 2. Datapath Bypassing (18 points)

Solution.

Question 2.A Resolving Data Hazards by Stalling (6 points)

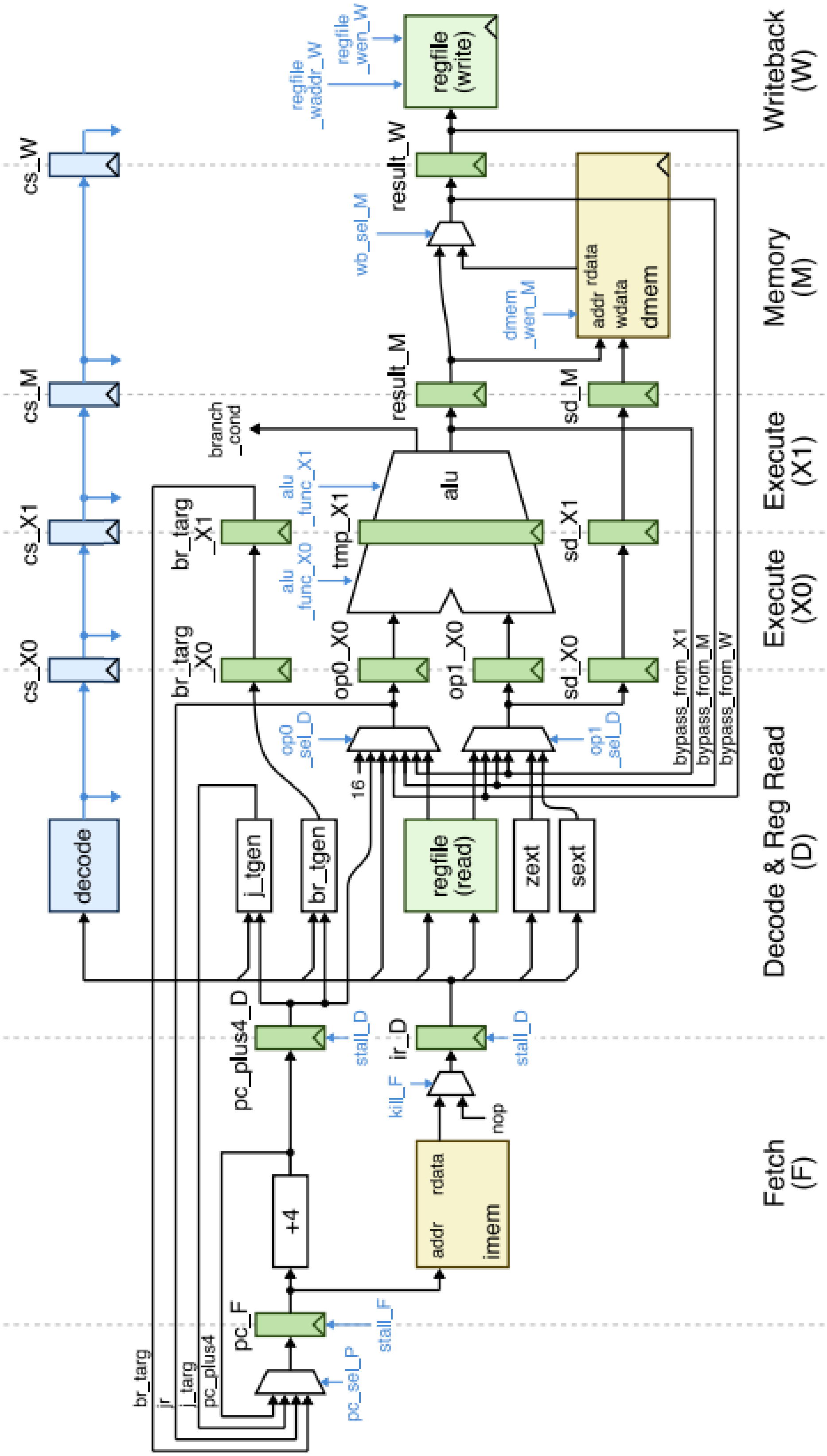
Dynamic Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`bne r1, r0, done`	F	D	X0	X1	M	W											
`lw r5, 0(r2)`		F	D	X0	X1	M	W										
`lw r6, 0(r3), 10`			F	D	X0	X1	M	W									
`addu r7, r5, r6`				F	D	D	D	D	X0	X1	M	W					
`addiu r8, r4, 4`					F	F	F	F	D	X0	X1	M	W				
`sw r7, 0(r8)`									F	D	D	D	D	X0	X1	M	W

Question 2.B Implementing Data Bypassing (6 points)

Modified datapath is shown on the next page. It is not possible to completely remove stalling using only bypassing because the pipeline has a load-use dependency where an lw instruction may produce a value (in the M stage) needed in the following instructions ALU operation. Additionally, dependent back-to-back integer operations will have to stall no matter what because of the two-stage execution.

Question 2.C Bypassing Execution (6 points)

Dynamic Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13
`bne r1, r0, done`	F	D	X0	X1	M	W								
`lw r5, 0(r2)`		F	D	X0	X1	M	W							
`lw r6, 0(r3), 10`			F	D	X0	X1	M	W						
`addu r7, r5, r6`				F	D	D	D	X0	X1	M	W			
`addiu r8, r4, 4`					F	F	F	D	X0	X1	M	W		
`sw r7, 0(r8)`								F	D	D	X0	X1	M	W


Fetch (F)
Decode & Reg Read (D)
Execute (X0)
Execute (X1)
Memory (M)
Writeback (W)
br_targ
jr
j_targ
pc_plus4
pc_sel_P
pc_F
stall_F
+4
addr
rdata
imem
kill_F
nop
ir_D
stall_D
pc_plus4_D
stall_D
decode
j_tgen
br_tgen
regfile (read)
zext
sext
16
op0 _sel_D
op1 _sel_D
bypass_from_X1
bypass_from_M
bypass_from_W
cs_X0
br_targ _X0
op0_X0
op1_X0
sd_X0
alu _func_X0
cs_X1
br_targ _X1
tmp_X1
sd_X1
alu _func_X1
alu
branch _cond
cs_M
result_M
sd_M
dmem _wen_M
addr
wdata
rdata
dmem
wb_sel_M
cs_W
result_W
regfile _waddr_W
regfile _wen_W
regfile (write)

Question 3. Branch Prediction (10 points)

Solution.

Each stall in your instruction increases the CPI of that instruction by one. Since CPI is an average, repeated occurrences of stalls do not contribute any further to CPI.

CPI = Base CPI + Stalls

In this system, we assume bypassing is implemented. Therefore, the only stalls are due to branch mispredictions and load-use latency.

CPI = Base CPI + Load Stalls + Branch Stalls

Loads comprise 12% of instructions and 30% of the time they cause a 2-cycle stall. Branches comprise 25% of instructions and 55% of the time they mispredict, which causes a 4-cycle penalty.

$\text{CPI} = 1 + (0.12 \times 0.30 \times 2) + (0.25 \times 0.55 \times 4)$

$\text{CPI} = 1.62$

Question 4. Out-of-Order Execution (18 points)

Solution.

In this problem, we denote the Reservation Station as R_s and the Reorder Buffer as R_o.

Question 4.A Out-of-Order with no Bypassing (6 points)

See the pipeline diagram on the second to last page.

Question 4.B Out-of-Order with Bypassing (6 points)

See the pipeline diagram on the last page.

Question 4.C Program Latency (6 points)

No Bypassing (Question 4.A):

- As shown in the pipeline diagram on the penultimate page, the first iteration of the loop starts at Cycle 4 and ends on cycle 30. This means a total of 27 cycles are needed for the first iteration.
- The `bne` instruction at the start of the next iteration only misfetches 2 instructions (rather than 4) so only 2 instructions are fetched and then squashed.
- These two squashed instructions do not count toward the cycle count because they overlap with instructions from the previous iteration that are still executing.
- Therefore, the `bne` instruction from the second iteration onward only adds 1 cycle to the overall cycle count.
- After this branch is resolved, the remainder of the second iteration onward behaves like Cycles 13-30, which is 18 cycles.
- Therefore, each iteration after the first adds 19 cycles to the total cycle count.
- The total cycles in the loop are $27 + 9 \times 19 = 198$ cycles.

With Bypassing (Question 4.B):

- As shown in the pipeline diagram on the last page, the first iteration of the loop starts at Cycle 4 and ends on cycle 23. This means a total of 20 cycles are needed for the first iteration.
- These two squashed instructions after the `bne` of the second iteration onward do not count toward the cycle count because they overlap with instructions from the previous iteration that are still executing.
- Therefore, the `bne` instruction from the second iteration onward only adds 1 cycle to the overall cycle count.
- After this branch is resolved, the remainder of the second iteration onward behaves like Cycles 11-23, which is 13 cycles.
- Therefore, each iteration after the first adds 14 cycles to the total cycle count.
- The total cycles in the loop are $20 + 9 \times 14 = 146$ cycles.

Dynamic Instruction															Cycle																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
`xor r0, r0, r0`	F	D	X	M	W																										
`addiu r1, r0, 10`		F	D	R_s	R_s	X	M	W																							
`j L1`			F	D	X	M	R_o	R_o	W																						
`lw r3, 0(r2)`				F	-	-	-	-																							
`bne r0, r1, -8`					F	D	R_s	R_s	X	M	W																				
...						F	D	X	M	-																					
...							F	D	X	-	-																				
...								F	D	-	-	-																			
...									F	-	-	-	-																		
`lw r3, 0(r2)`										F	D	X	M	W																	
`mul r4, r3, r3`											F	D	R_s	R_s	X0	X1	X2	X3	M	W											
`mul r3, r3, r1`												F	D	R_s	R_s	X0	X1	X2	X3	M	W										
`addiu r0, r0, 1`													F	D	X	M	R_o	R_o	R_o	R_o	R_o	W									
`div r3, r4, r3`														F	D	R_s	R_s	R_s	R_s	R_s	R_s	X0	X0	X0	X0	M	W				
`sw r3, 0(r2)`															F	D	R_s	R_s	R_s	R_s	R_s	R_s	R_s	R_s	R_s	R_s	R_s	X	M	W	
`addiu r2, r2, 4`																F	D	R_s	R_s	X	M	R_o	R_o	R_o	R_o	R_o	R_o	R_o	R_o	R_o	W

Dynamic	Cycle																							
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
`xor r0, r0, r0`	F	D	X	M	W																			
`addiu r1, r0, 10`		F	D	X	M	W																		
`j L1`			F	D	X	M	W																	
`lw r3, 0(r2)`				F	-	-	-	-																
`bne r0, r1, -8`					F	D	X	M	W															
`...`						F	D	-	-	-														
`...`							F	-	-	-	-													
`lw r3, 0(r2)`								F	D	X	M	W												
`mul r4, r3, r3`									F	D	R_s	X0	X1	X2	X3	M	W							
`mul r3, r3, r1`										F	D	R_s	X0	X1	X2	X3	M	W						
`addiu r0, r0, 1`											F	D	X	M	R_o	R_o	R_o	R_o	W					
`div r3, r4, r3`												F	D	R_s	R_s	R_s	X0	X0	X0	X0	M	W		
`sw r3, 0(r2)`													F	D	R_s	R_s	R_s	R_s	R_s	R_s	X	M	W	
`addiu r2, r2, 4`														F	D	R_s	X	M	R_o	R_o	R_o	R_o	R_o	W