CS 347, Spring 1998
Lab 3 Prologue Solutions

Notation

In the following diagrams, the letters F, D, E, M, and W represent pipeline stages Instruction Fetch, Instruction Decode, Execute, Memory Access, and Write-Back respectively. Forwarding is shown using special subscript and superscript notation. When data is forwarded from a pipeline stage, the register to which the forwarded data will eventually be written is shown as a subscript following the stage, e.g. $E_{(\$1)}$ shows that the data that will be written to register 1 is forwarded from the Execute stage. A superscript version of this notation is used to represent the stage to which data is forwarded, e.g. $^{(\$1)}E$ in the next column represents EX-EX forwarding.

1. Correct Version, Stall mode, `pipe1.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
ldq $4, 64($3)	F	F	F	D	E	M	W					
addq $3,0x4,$5				F	D	E	M	W				
addq $4,$5,$2					F	D	D	D	E	M	W	
ldq $5,44($2)						F	F	F	D	D	D	E
addq $5,0x8,$3									F	F	F	D
ldq $7,56($3)												F
stq $7,8($3)												

Instr	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}
ldq $4, 64($3)												
addq $3, 0x4,$5												
addq $4,$5,$2												
ldq $5,44($2)	M	W										
addq $5,0x8,$3	D	D	E	M	W							
ldq $7,56($3)	F	F	D	D	D	E	M	W				
stq $7,8($3)			F	F	F	D	D	D	E	M	W	

2. Buggy Version, Stall mode, `pipe1.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
ldq $4, 64($3)	F	F	F	D	E	M	W					
addq $3,0x4,$5				F	D	E	M	W				
addq $4,$5,$2					F	D	D	D	E	M	W	
ldq $5,44($2)						F	F	F	D	E	M	W
addq $5,0x8,$3									F	D	D	D
ldq $7,56($3)										F	F	F
stq $7,8($3)												

Instr	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}
ldq $4, 64($3)												
addq $3, 0x4,$5												
addq $4,$5,$2												
ldq $5,44($2)												
addq $5,0x8,$3												
ldq $7,56($3)												
stq $7,8($3)												

Describe where the bug occurs and its effect on the program:

At cycle c_9, ldq $5, 44($2) must stall in the instruction decode stage until addq $4, $5, $2 writes back its result to register 2. In the buggy version, this stall does not take place. Therefore the ldq instruction reads the value 0 from register 2 in the decode stage, resulting in an incorrect address calculation. The end result is an unaligned memory access exception which halts the simulation after cycle c_{11}.

3. Correct Version, Forwarding mode, `pipe1.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
ldq \$4, 64(\$3)	F	D	E	$M_{(\$4)}$	W									
addq \$3,0x4,\$5		F	D	$E_{(\$5)}$	M	W								
addq \$4,\$5,\$2			F	D	$^{(\$4,\$5)}E_{(\$2)}$	M	W							
ldq \$5,44(\$2)				F	D	$^{(\$2)}E$	$M_{(\$5)}$	W						
addq \$5,0x8,\$3					F	D	D	$^{(\$5)}E_{(\$3)}$	$M_{(\$3)}$	W				
ldq \$7,56(\$3)						F	F	D	$^{(\$3)}E$	$M_{(\$7)}$	W			
stq \$7,8(\$3)								F	D	$^{(\$3)}E$	$^{(\$7)}M$	W		

4. Buggy Version, Forwarding mode, `pipe1.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
ldq $4, 64($3)	F	D	E	$M_{(\$4)}$	W									
addq $3,0x4,$5		F	D	$E_{(\$5)}$	M	W								
addq $4,$5,$2			F	D	$^{(\$4,\$5)}E_{(\$2)}$	M	W							
ldq $5,44($2)				F	D	$^{(\$2)}E$	$M_{(\$5)}$	W						
addq $5,0x8,$3					F	D	D	$^{(\$5)}E_{(\$3)}$	M	W				
ldq $7,56($3)						F	F	D	$^{(\$3)}E$	$M_{(\$7)}$	W			
stq $7,8($3)								F	D	D	$^{(\$7)}E$	M	W	

Describe where the bug occurs and its effect on the program:

The buggy version stalls incorrectly when a stq instruction follows a ldq instruction. The correct version uses MEM-MEM forwarding to transfer data in register 7 between the last two instructions. The buggy version incurs a one cycle performance penalty.

5. Correct Version, Stall mode, `pipe2.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
addq $31, 0x10, $3	F	D	E	M	W							
bne $3, targ1		F	D	D	D	E	M	W				
addq $3, $3, $4			F	F	F	F	F					
addq $3, $4, $5												
addq $3, $5, $6												
targ1: addq $3, $3, $7								F	D	E	M	W
beq $7, targ2									F	D	D	D
addq $3, $7, $1										F	F	F
targ2: addq $3, $7, $2												

Instr	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}
targ1: addq $3, $3, $7												
beq $7, targ2	E	M	W									
addq $3, $7, $1	F	F	D	E	M	W						
targ2: addq $3, $7, $2			F	D	E	M	W					

6. Buggy Version, Stall mode, `pipe2.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
addq $31, 0x10, $3	F	D	E	M	W							
bne $3, targ1		F	D	D	D	E	M	W				
addq $3, $3, $4			F	F	F	F	D	E	M	W		
addq $3, $4, $5							F					
addq $3, $5, $6												
targ1: addq $3, $3, $7								F	D	E	M	W
beq $7, targ2									F	D	D	D
addq $3, $7, $1										F	F	F
targ2: addq $3, $7, $2												

Instr	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}
targ1: addq $3, $3, $7												
beq $7, targ2	E	M	W									
addq $3, $7, $1	F	D	E	M	W							
targ2: addq $3, $7, $2		F	D	E	M	W						

Describe where the bug occurs and its effect on the program:

The buggy version of the simulator fails to cancel the instruction following the branch when the branch is taken. This results in a value being incorrectly written to register 4.

7. Correct Version, Forwarding mode, `pipe2.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
addq \$31, 0x10, \$3	F	D	$E_{(\$3)}$	$M_{(\$3)}$	W									
bne \$3, targ1		F	D	$^{(\$3)}E$	M	W								
addq \$3, \$3, \$4			F	D	$^{(\$3)}E$									
addq \$3, \$4, \$5				F	D									
addq \$3, \$5, \$6					F									
targ1: addq \$3, \$3, \$7						F	D	$E_{(\$7)}$	$M_{(\$7)}$	W				
beq \$7, targ2							F	D	$^{(\$7)}E$	M	W			
addq \$3, \$7, \$1								F	D	$^{(\$7)}E$	M	W		
targ2: addq \$3, \$7, \$2									F	D	E	M	W	

8. Buggy Version, Forwarding mode, `pipe2.O`

Instr	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
addq \$31, 0x10, \$3	F	D	$E_{(\$3)}$	$M_{(\$3)}$	W									
bne \$3, targ1		F	D	$^{(\$3)}E$	M	W								
addq \$3, \$3, \$4			F	D	$^{(\$3)}E$									
addq \$3, \$4, \$5				F	D									
addq \$3, \$5, \$6					F	D	E	M	W					
targ1: addq \$3, \$3, \$7						F	D	$E_{(\$7)}$	$M_{(\$7)}$	W				
beq \$7, targ2							F	D	$^{(\$7)}E$	M	W			
addq \$3, \$7, \$1								F	D	$^{(\$7)}E$	M	W		
targ2: addq \$3, \$7, \$2									F	D	E	M	W	

Describe where the bug occurs and its effect on the program:

The buggy version of the simulator fails to cancel the third instruction following the branch when the branch is taken. This results in a value being incorrectly written to register 6.