COE 501: Computer Architecture

Problem Set 5: Advanced Pipelining

Solution

1) (10 pts) Consider the following so-called DAXPY loop used in Gaussian elimination. The loop implements the vector operation **Y = a * X + Y** for vectors **X** and **Y** of length 100. Initially, R1 is the address of X[0], R2 = address of Y[0], and R3 = address of X[100]. Here is the MIPS code:

```
loop:  L.D     F2, 0(R1)        ; load F2 = X[i]
       MUL.D   F3, F1, F2       ; F3 = a * X[i]
       L.D     F4, 0(R2)        ; load F4 = Y[i]
       ADD.D   F4, F3, F4       ; F4 = a * X[i] + Y[i]
       S.D     F4, 0(R2)        ; store Y[i] = F4
       ADDIU   R1, R1, 8        ; R1 = address of next X[i]
       ADDIU   R2, R2, 8        ; R2 = address of next Y[i]
       BNE     R1, R3, loop     ; loop if not done
```

Consider the execution of the above loop on an in-order execution pipeline that has an instruction fetch (IF) stage, an instruction decode (ID) stage, three independent function units: ALU unit (1 pipeline EX stage), FP unit for floating-point addition, subtraction, and multiplication (4 pipeline stages: FP1, FP2, FP3, and FP4), and a memory unit for load and store instructions (2 pipeline stages: A for address calculation and M for memory access). Finally, there is a write-back (WB) stage that writes-back the result of an instruction into the register file. Each cycle, at most one instruction is fetched and at most one is completed. All hazards are detected early in the ID stage. If an instruction in the ID stage cannot issue for execution then it will stall the pipeline.

a) (5 pts) Show the execution of one loop iteration. Assume a full-forwarding network that forwards results between function units. In addition, assume that the BNE instruction is predicted to be always taken (with zero delay) in the IF stage. Draw a timing diagram, showing stall cycles and forwarding. What is the total number of cycles to execute the 100 iterations of the above loop and the average CPI?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
L.D	IF	ID	A	M	WB														
MUL.D		IF	ID	ID	FP1	FP2	FP3	FP4	WB										
L.D			IF	IF	ID	A	M	WB											
ADD.D					IF	ID	ID	ID	FP1	FP2	FP3	FP4	WB						
S.D						IF	IF	IF	ID	ID	ID	ID	A	M					
ADDIU									IF	IF	IF	IF	ID	EX	WB				
ADDIU													IF	ID	EX	WB			
BNE														IF	ID	EX			
L.D															IF	ID	A	M	WB
	14 cycles per iteration = 8 Instruction cycles + 6 stall cycles														Start 2nd Iteration				

Total Cycles = 14 x 100 = 1400 cycles
Average CPI = 14 / 8 = 1.75

b) **(5 pts)** Reorder the instructions in the above loop to reduce the total number of clock cycles. Redraw the timing diagram showing stall cycles and forwarding. Re-compute the average CPI.

One possible reordering is shown below. Because ADDIU R2, R2, 8 is moved above the S.D instruction, the address of Y[i] in the S.D instruction is changed from 0(R2) to -8(R2).

```
loop:  L.D      F2, 0(R1)        ; load F2 = X[i]
       L.D      F4, 0(R2)        ; load F4 = Y[i]
       MUL.D    F3, F1, F2       ; F3 = a * X[i]
       ADD.D    F4, F3, F4       ; F4 = a * X[i] + Y[i]
       ADDIU    R1, R1, 8        ; R1 = address of next X[i]
       ADDIU    R2, R2, 8        ; R2 = address of next Y[i]
       S.D      F4, -8(R2)       ; store Y[i] = F4
       BNE      R1, R3, loop     ; loop if not done
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D	IF	ID	A	M	WB												
L.D		IF	ID	A	M	WB											
MUL.D			IF	ID	FP1	FP2	FP3	FP4	WB								
ADD.D				IF	ID	ID	ID	ID	FP1	FP2	FP3	FP4	WB				
ADDIU					IF	IF	IF	IF	ID	EX	WB						
ADDIU									IF	ID	EX	WB					
S.D										IF	ID	ID	A	M	WB		
BNE											IF	IF	ID	EX			
L.D													IF	ID	A	M	WB
	12 cycles per iteration = 8 instruction cycles + 4 stall cycles												Start 2nd Iteration				

Average CPI = 12 / 8 = 1.5

2) (5 pts) This problem explores hardware register renaming. To eliminate name dependences, WAR, and WAW hazards, the hardware renames registers. Assume the hardware has a pool of temporary registers (called T1 through T63), that are allocated in sequence. Unroll the first two iterations of the above DAXPY loop and rename all the destination registers, starting at T1. Make sure to rename the source registers as well to preserve data dependences.

Solution:

```
loop:   L.D     T1, 0(R1)         ; load T1 = X[i]
        MUL.D   T2, F1, T1        ; T2 = a * X[i]
        L.D     T3, 0(R2)         ; load T3 = Y[i]
        ADD.D   T4, T2, T3        ; T4 = a * X[i] + Y[i]
        S.D     T4, 0(R2)         ; store Y[i] = T4
        ADDIU   T5, R1, 8         ; T5 = address of next X[i]
        ADDIU   T6, R2, 8         ; T6 = address of next Y[i]
        BNE     T5, R3, loop      ; loop if not done

        L.D     T7, 0(T5)         ; load T7 = X[i]
        MUL.D   T8, F1, T7        ; T8 = a * X[i]
        L.D     T9, 0(T6)         ; load T9 = Y[i]
        ADD.D   T10, T8, F4       ; T10 = a * X[i] + Y[i]
        S.D     T10, 0(T6)        ; store Y[i] = T10
        ADDIU   T11, T5, 8        ; T11 = address of next X[i]
        ADDIU   T12, T6, 8        ; T12 = address of next Y[i]
        BNE     T11, R3, loop     ; loop if not done
```

3) (6 pts) Consider the execution of the above DAXPY loop on an out-of-order execution pipeline. A new Issue Stage (IS) is added to the pipeline with sufficient number of reservation stations. The reservation stations are distributed among the function units. As in problem 1, there are three independent functions units: ALU unit (1 pipeline EX stage), FP unit (4 pipeline stages: FP1, FP2, FP3, and FP4), and a memory unit for load and store instructions (2 pipeline stages: A-Unit for address calculation and M-Unit for Data Cache access). In addition, there is a Store Buffer (SB) for store instructions only. Store instructions wait in the store buffer until their data is present. A load instruction is allowed to bypass a previous store waiting in the store buffer if the load address is different from the store address. Instructions are allowed to complete out of program order. Draw a timing diagram showing the execution of the first two iterations of the DAXPY loop on the OOO execution pipeline. Assume that the loop branch is predicted to be always taken.

Solution:

	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
L.D	IF	ID	**IS**	A	M	WB																			
MUL.D		IF	ID	IS	IS	**IS**	FP1	FP2	FP3	FP4	WB														
L.D			IF	ID	**IS**	A	M	WB																	
ADD.D				IF	ID	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	WB									
S.D					IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	**SB**	M								
ADDIU						IF	ID	**IS**	EX	WB															
ADDIU							IF	ID	**IS**	EX	WB	WB													
BNE								IF	ID	**IS**	EX														
L.D									IF	ID	**IS**	A	M	WB											
MUL.D										IF	ID	IS	IS	**IS**	FP1	FP2	FP3	FP4	WB						
L.D											IF	ID	**IS**	A	M	WB	WB								
ADD.D												IF	ID	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	WB	
S.D													IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	**SB**	M
ADDIU														IF	ID	**IS**	EX	WB							
ADDIU															IF	ID	**IS**	EX	WB	WB					
BNE																IF	ID	**IS**	EX						

The arrow indicates the forwarding of the result on the common data bus to a reservation station in the issue stage.

The Store instruction (S.D) waits in the store buffer (SB) until the data is present.

The two L.D instructions in the second iteration are allowed to bypass the S.D instruction of the first iteration because their addresses are different.

A conflicting WB (Write Back) is shown in Red.

4) (7 pts) A new sufficiently large reorder buffer (ROB) and a Commit Stage (C) are now added to the pipeline. Instructions are issued for execution out of program order. However, they commit their results in program order by updating the register file (or memory in the case of a store instruction). Show the timing diagram of the first two iterations of the DAXPY loop, given that the loop branch is predicted to be always taken.

Solution:

	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
L.D	IF	ID	**IS**	A	M	W	C																						
MUL.D		IF	ID	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C																	
L.D			IF	ID	**IS**	A	M	W	rob	rob	rob	rob	C																
ADD.D				IF	ID	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C												
S.D					IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	**SB**	SB	C											
ADDIU						IF	ID	**IS**	EX	W	rob	rob	rob	rob	rob	rob	rob	rob	C										
ADDIU							IF	ID	**IS**	EX	W	W	rob	rob	rob	rob	rob	rob	rob	C									
BNE								IF	ID	**IS**	EX	rob	rob	rob	rob	rob	rob	rob	rob	rob	C								
L.D									IF	ID	**IS**	A	M	W	rob	rob	rob	rob	rob	rob	rob	C							
MUL.D										IF	ID	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	rob	rob	rob	C						
L.D											IF	ID	**IS**	A	M	W	W	rob	rob	rob	rob	rob	rob	C					
ADD.D												IF	ID	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C				
S.D													IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	**SB**	SB	C			
ADDIU														IF	ID	**IS**	EX	W	rob	rob	rob	rob	rob	rob	rob	rob	C		
ADDIU															IF	ID	**IS**	EX	W	W	rob	rob	rob	rob	rob	rob	rob	C	
BNE																IF	ID	**IS**	EX	rob	rob	rob	rob	rob	rob	rob	rob	rob	C

The arrow indicates the forwarding of the result on the common data bus to a reservation station in the issue stage.

Instructions write their results to the Reorder Buffer (ROB). They commit their results in program order (commit stage C).

A conflicting W (Write to Reorder Buffer) is shown in Red.

The Store instruction (S.D) waits in the store buffer (SB) until the commit stage C.

The two L.D instructions in the second iteration are allowed to bypass the S.D instruction of the first iteration because their addresses are different.

At steady state, one instruction is committed per cycle.

5) **(7 pts)** Consider a superscalar pipeline that fetches two instructions, decodes two instructions, dispatches two instructions, writes two results into the ROB, and commits the results of at most two instructions each cycle. Because there are three independent function units, up to three instructions can be issued for execution in a given clock cycle. Redraw the timing diagram of the first two iterations of the DAXPY loop on the superscalar processor, given that the loop branch is predicted to be always taken.

Solution:

```
loop:  L.D     F2, 0(R1)        ; load F2 = X[i]
       MUL.D   F3, F1, F2       ; F3 = a * X[i]
       L.D     F4, 0(R2)        ; load F4 = Y[i]
       ADD.D   F4, F3, F4       ; F4 = a * X[i] + Y[i]
       S.D     F4, 0(R2)        ; store Y[i] = F4
       ADDIU   R1, R1, 8        ; R1 = address of next X[i]
       ADDIU   R2, R2, 8        ; R2 = address of next Y[i]
       BNE     R1, R3, loop     ; loop if not done
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
L.D	IF	ID	**IS**	A	M	W	C																
MUL.D	IF	ID	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C											
L.D		IF	ID	**IS**	A	M	W	rob	rob	rob	rob	C											
ADD.D		IF	ID	IS	IS	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C						
S.D			IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	SB	SB	**SB**	C						
ADDIU			IF	ID	**IS**	EX	W	rob	rob	rob	rob	rob	rob	rob	rob	rob	rob	C					
ADDIU				IF	ID	**IS**	EX	W	rob	rob	rob	rob	rob	rob	rob	rob	rob	C					
BNE				IF	ID	IS	**IS**	EX	rob	rob	rob	rob	rob	rob	rob	rob	rob	rob	C				
L.D					IF	ID	**IS**	A	M	W	rob	rob	rob	rob	rob	rob	rob	rob	C				
MUL.D					IF	ID	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	rob	rob	rob	rob	C			
L.D						IF	ID	**IS**	A	M	W	rob	rob	rob	rob	rob	rob	rob	rob	C			
ADD.D						IF	ID	IS	IS	IS	IS	IS	IS	IS	**IS**	FP1	FP2	FP3	FP4	W	C		
S.D							IF	ID	**IS**	A	SB	SB	SB	SB	SB	SB	SB	SB	SB	**SB**	C		
ADDIU							IF	ID	**IS**	EX	W	W	rob	rob	rob	rob	rob	rob	rob	rob	rob	C	
ADDIU								IF	ID	**IS**	EX	W	rob	rob	rob	rob	rob	rob	rob	rob	rob	C	
BNE								IF	ID	IS	IS	**IS**	EX	rob	rob	rob	rob	rob	rob	rob	rob	rob	C

Two instructions are fetched, decoded, and dispatched each cycle.

At most two results are written into the ROB during the same cycle.

A third write during cycle 11 is a conflicting write W shown in red.

At steady state, two instructions are committed per cycle.