Hardware Security

Homework 2 - Solution

Student Number: Name:

Problem 1. (25pt) Register renaming

Reschedule the loop in the following code to minimize stalls with renaming registers to temporary registers (Tx) to enable parallel execution.

```
Loop:     L.D       F4, 0(R1)
          <Stall>
          MULT.D    F6, F4, F2
          <Stall>
          <Stall>
          <Stall>
          L.D       F4, 0(R2)
          <Stall>
          MULT.D    F8, F4, F2
          <Stall>
          <Stall>
          <Stall>
          S.D       F6, 0(R3)
          S.D       F8, -8(R3)
          DADDUI    R1, R1, #8
          <Stall>
          DADDUI    R2, R2, #8
          <Stall>
          DADDUI    R3, R3, #-16
          <Stall>
          BNE       R1, R4, Loop
```

```
Loop:     L.D       F4, 0(R1)
          L.D       T2, 0(R2)
          MULT.D    F6, F4, F2
          MULT.D    F8, T2, F2
          DADDUI    R1, R1, #8
          DADDUI    R2, R2, #8
          S.D       F6, 0(R3)
          S.D       F8, -8(R3)
          DADDUI    R3, R3, #-16
          <Stall>
          BNE       R1, R4, Loop
```

Problem 2. (25pt) Branch predictor

Consider a 2-bit saturating counter predictor that starts in the strongly not taken state (0 value). For the following branch outcome sequence, show the prediction result, and the state of the predictor.

Outcome	State after branch resolution	Prediction of the next
	0	Not taken
Taken	1	Not taken
Taken	2	Taken
Taken	3	Taken
Not taken	2	Taken
Taken	3	Taken
Taken	3	Taken
Not taken	2	Taken
Not taken	1	Not taken
Not taken	0	Not taken
Not taken	0	Not taken
Taken	1	Not taken
Not taken	0	Not taken

Problem 3. (25pt) Tomasulo algorithm

The following table shows how the Tomasulo algorithms works when there is only one adder, assuming every stage (issue, execute, and write) is done in one cycle. Fill the table below that shows the process when there are two adders.

[Ins: Instruction, Iss: Issue, Ex: Execute, Wr: Write]
[Na: Name, Bs: Busy, L1: Load1, A1: Adder1, A2: Adder2]

```
(1)     L.D       F2, 0(R1)
(2)     ADD.D     F4, F2, F0
(3)     ADD.D     F6, F4, F0
```

	Instruction Status				Reservation Station								Register Status				
1	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√			L1	Y	L					R1	Qi		L1		
	(2)				A1												
	(3)																
2	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√	√		L1	Y	L					R1	Qi		L1	A1	
	(2)	√			A1	Y	A		F0	L1							
	(3)																
3	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√	√	√	L1								Qi			A1	
	(2)	√	√		A1	Y	A	F2	F0								
	(3)																
4	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√	√	√	L1								Qi				A1
	(2)	√	√	√	A1	Y	A	F4	F0								
	(3)	√															
5	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√	√	√	L1								Qi				A1
	(2)	√	√	√	A1	Y	A	F4	F0								
	(3)	√	√														
6	Ins	Iss	Ex	Wr	Na	Bs	Op	Vj	Vk	Qj	Qk	A	F	F0	F2	F4	F6
	(1)	√	√	√	L1								Qi				
	(2)	√	√	√	A1												
	(3)	√	√	√													

1

Instruction Status

Ins	Iss	Ex	Wr
(1)	√		
(2)			
(3)			

Reservation Station

Na	Bs	Op	Vj	Vk	Qj	Qk	A
L1	Y	L					R1
A1							
A2							

Register Status

F	F0	F2	F4	F6
Qi		L1		

2

Instruction Status

Ins	Iss	Ex	Wr
(1)	√	√	
(2)	√		
(3)			

Reservation Station

Na	Bs	Op	Vj	Vk	Qj	Qk	A
L1	Y	L					R1
A1	Y	A		F0	L1		
A2							

Register Status

F	F0	F2	F4	F6
Qi		L1	A1	

3

Instruction Status

Ins	Iss	Ex	Wr
(1)	√	√	√
(2)	√	√	
(3)	√		

Reservation Station

Na	Bs	Op	Vj	Vk	Qj	Qk	A
L1							
A1	Y	A	F2	F0			
A2	Y	A		F0	A1		

Register Status

F	F0	F2	F4	F6
Qi			A1	A2

4

Instruction Status

Ins	Iss	Ex	Wr
(1)	√	√	√
(2)	√	√	√
(3)	√	√	

Reservation Station

Na	Bs	Op	Vj	Vk	Qj	Qk	A
L1							
A1							
A2	Y	A	F4	F0			

Register Status

F	F0	F2	F4	F6
Qi				A2

5

Instruction Status

Ins	Iss	Ex	Wr
(1)	√	√	√
(2)	√	√	√
(3)	√	√	√

Reservation Station

Na	Bs	Op	Vj	Vk	Qj	Qk	A
L1							
A1							
A2							

Register Status

F	F0	F2	F4	F6
Qi				

Problem 4. (25pt) Loop-level parallelism

In the following loop, find all the true dependences, output dependences, and antidependences. Eliminate the output dependences and antidependences by renaming.

```
for (i=0;i<100;i++) {
    A[i] = A[i] * B[i]; /* S1 */
    B[i] = A[i] + c;    /* S2 */
    A[i] = C[i] * c;    /* S3 */
    C[i] = D[i] * A[i]; /* S4 */
}
```

```
for (i=0;i<100;i++) {
    T[i] = A[i] * B[i]; /* S1 */
    B1[i] = T[i] + c;    /* S2 */
    A1[i] = C[i] * c;    /* S3 */
    C1[i] = D[i] * A1[i]; /* S4 */

}
```