

Computer Architecture-I

Assignment 4 Solution

1. The code with hardware register renaming will look as follows:

	Instruction
LOOP:	LD T9, 0 (Rx)
I0:	MULTD T10, F0, T9
I1:	DIVD T11, F0, T9
I2:	LD T12, 0 (Ry)
I3:	ADDD T13, F0, T12
I4:	ADDD T14, T11, T9
I5:	SD T12, 0 (Ry)

2. In this case, the register renaming hardware looks and updates two instructions in the same clock cycle.

```

I0:  MULTD    F5, F0, F2
I1:  ADDD     F9, F5, F4
I2:  ADDD     F5, F5, F2
I3:  DIVD     F2, F9, F0
    
```

Register Rename Table					
Init.		CLK 1		CLK 2	
Available Regs: T9, T10		Available Regs: T11, T12		Available Regs: T13, T14	
0	T0	0	T0	0	T0
1	T1	1	T1	1	T1
2	T2	2	T2	2	T12
3	T3	3	T3	3	T3
4	T4	4	T4	4	T4
5	T5	5	T9	5	T11
6	T6	6	T6	6	T6
7	T7	7	T7	7	T7
8	T8	8	T8	8	T8
9	T9	9	T10	9	T10
10	T10	10	T10	10	T10
...
...
63	T63	63	T63	63	T63

3. For $R1 = 5$, when the following sequence of instructions is executed, the final value in $R1$ is shown below:

```

ADDI    R1,R1,R1 ; Val(R1) = 10
ADDI    R1,R1,R1 ; Val(R1) = 20
ADDI    R1,R1,R1 ; Val(R1) = 40

```

5. Design with Dynamic scheduling:

- a. All the instructions are in the Reservation Station (RS). Gains can come by using register renaming to speed up instructions in successive iterations which are independent in data. It is assumed for all the sub-questions that the functional units are not pipelined. The destination registers for the highlighted instructions can be renamed to avoid a WAW hazard.

	Instruction	Latency
LOOP:	LD F2,0 (Rx)	1+3
I0:	MULTD F2,F0,F2	1+4
I1:	DIVD F8,F2,F0	1+10
I2:	LD F4,0 (Ry)	1+3
I3:	ADDD F4,F0,F4	1+2
I4:	ADDD F10,F8,F2	1+2
I5:	SD F4,0 (Ry)	1+1
I6:	ADDI Rx,Rx,#8	1+0
I7:	ADDI Ry,Ry,#8	1+0
I8:	SUB R20,R4,Rx	1+0
I9:	BNZ R20,LOOP	1+1

- b. The RS can dispatch instructions in the following manner to achieve optimal performance.

Clock	ALU1	ALU2	LD/ST
1	ADDI Rx,Rx,#8		LD T2,0 (Rx)
2	SUB R20,R4,Rx		<Stall>
3			<Stall>
4			<Stall>
5		MULTD F2,F0,T2	LD T3,0 (Ry)
6		<Stall>	<Stall>
7		<Stall>	<Stall>
8		<Stall>	<Stall>
9	ADDD F4,F0,T3	<Stall>	
10	<Stall>	DIVD F8,F2,F0	
11	<Stall>	<Stall>	

12	ADDI Ry,Ry,#8	<Stall>	SD F4,0(Ry)
13		<Stall>	<Stall>
14		<Stall>	<Stall>
15		<Stall>	
16		<Stall>	
17		<Stall>	
18		<Stall>	
19		<Stall>	
20		<Stall>	
21	BNZ R20,LOOP	ADDD F10,F8,F2	
22	<Stall>	<Stall>	
23		<Stall>	

- c. Two instructions are fed to the RS per clock cycle. The given register renamed code can be scheduled as shown. It takes 23 clock cycles.

Clock	ALU1	ALU2	LD/ST
1			LD T2,0(Rx)
2			<Stall>
3			<Stall>
4	ADDI Rx,Rx,#8		<Stall>
5	SUB R20,R4,Rx	MULTD F2,F0,T2	LD T3,0(Ry)
6		<Stall>	<Stall>
7		<Stall>	<Stall>
8		<Stall>	<Stall>
9	ADDD F4,F0,T3	<Stall>	
10	<Stall>	DIVD F8,F2,F0	
11	<Stall>	<Stall>	
12	ADDI Ry,Ry,#8	<Stall>	SD F4,0(Ry)
13		<Stall>	<Stall>
14		<Stall>	<Stall>
15		<Stall>	
16		<Stall>	
17		<Stall>	
18		<Stall>	
19		<Stall>	
20		<Stall>	
21	BNZ R20,LOOP	ADDD F10,F8,F2	
22	<Stall>	<Stall>	
23		<Stall>	

- d. This loop has a true data dependence between the following instructions:

```
LD T2,0(Rx)
MULTD F2,F0,T2
DIVD F8,F2,F0
ADDD F10,F8,F2
```

These instructions are basically determining the loop iteration time of 23 cycles. Adding another ALU might allow scheduling a few more operations concurrently but, does not help in speeding up this bottleneck. Adding another LD/ST unit will give gains considering multiple iterations of this loop. But, the speedup achieved by cutting the latency of the longest instruction (in this case, the DIVD instruction) will give the best speed-up. If the DIVD takes 5 cycles instead of 10, it will give an improvement of 5 cycles in loop iteration time i.e. a speed-up of 1.277.

- e. We want to use speculation in scheduling two copies of the loops in RS. Here, the functional units are assumed to be fully pipelined. The schedule for the two successive iterations is as follows:

Clock	ALU1	ALU2	LD/ST
1	ADDI Rx,Rx,#8		LD T2,0(Rx)
2	SUB R20,R4,Rx		LD T3,0(Ry)
3	BNZ R20,LOOP		<Stall>
4	<Stall>		<Stall>
5	<i>ADDI Rx,Rx,#8</i>	MULTD F2,F0,T2	<i>LD T2,0(Rx)</i>
6	ADDD F4,F0,T3	<Stall>	<Stall>
7	<Stall>	<Stall>	<Stall>
8	<Stall>	<Stall>	<Stall>
9	ADDI Ry,Ry,#8	<Stall>	SD F4,0(Ry)
10	<i>SUB R20,R4,Rx</i>	DIVD F8,F2,F0	<i>LD T3,0(Ry)</i>
11	<i>BNZ R20,LOOP</i>	<Stall>	<Stall>
12	<i>MULTD F2,F0,T2</i>	<Stall>	<Stall>
13	<Stall>	<Stall>	<Stall>
14	<Stall>	<Stall>	
15	<Stall>	<Stall>	
16	<Stall>	<Stall>	
17	<i>DIVD F8,F2,F0</i>	<Stall>	
18	<Stall>	<Stall>	
19	<Stall>	<Stall>	
20	<Stall>	<Stall>	
21	<Stall>	ADDD F10,F8,F2	
22	<Stall>	<i>ADDD F4,F0,T3</i>	

23	<Stall>	<Stall>	
24	<Stall>	<Stall>	
25	<Stall>	<i>ADDI Ry,Ry,#8</i>	<i>SD F4,0(Ry)</i>
26	<Stall>		<Stall>
27	<Stall>		<Stall>
28		<i>ADD F10,F8,F2</i>	
29		<Stall>	
30		<Stall>	

A single iteration would take 23 cycles. Using speculative scheduling, we dispatch two iterations of the loop in 30 cycles as against (23 x 2 = 46 cycles). The instructions for the second iteration are shown in red italics.