Computer Architecture Homework Assignment #2

Grading Policy

TA in charge: Jinkwon Kim
E-mail: coco@kaist.ac.kr
Office: N1 - #922

Questions 1

1.A (25 points)

The candidate answer is described as below:

Cycles	Unscheduled instruction	Schedule instruction
1	DADDIU R4,R1,#800	DADDIU R4,R1,#800
2	LD R2, 0(R1)	LD F2, 0(R1)
3	Stall	LD F6, 0(R2)
4	Stall	DADDIU R1, R1, #8
5	MUL.D F4, F2, F0	MUL.D F4, F2, F0
6	LD F6, 0(R2)	DADDIU R2, R2, #8
7	Stall	DSLTU R3, R1, R4
8	Stall	Stall
9	Stall	Stall
10	Stall	Stall
11	Stall	Stall
12	ADD.D F6, F4, F6	ADD.D F6, F4, F6
13	Stall	Stall
14	Stall	Stall
15	Stall	Stall
16	Stall	BNEZ R3, foo
17	S.D F6, 0(R2)	S.D F6, -8(R2)
18	DADDIU R1, R1, #8	
19	DADDIU R2, R2, #8	
20	Stall	
21	DSLTU R3, R1, R4	
22	Stall	
23	Stall	
24	BNEZ R1, foo	
25	Stall	

- The stall cycles is calculated based on the lecture note.6 22p (Latency usually equals stall cycles when full forwarding is used(latency -1 = stall cycles is also correct answer))
- Example answer:
  - The unscheduled instructions per element takes 24 cycles.
  - The scheduled instruction per element take 16 cycles.
  - 50% performance improvement is achieved.

- **Grading Policy**
- 10 points – unscheduled instruction (-2 point, stall is not correct or stall is missing after BNEZ)
- 15 points – scheduled instruction (-2 point, if stall is not correct or BNEZ stall is not hiding)
- Any order of instruction in scheduled case is accepted if the stall is correct and it does not increase the clock cycle time.(-2 point, if it incurs additional cycles)
- Since the first instruction is not the part of the loop, it can be ignored.

1.B (25 points)

The candidate answer is described as below

Cycles	Unrolling 3 times
1	DADDIU R4,R1,#800
2	L.D F1, 0(R1)
3	L.D F2, 8(R1)
4	L.D F3, 16(R1)
5	MUL.D F4, F1, F0
6	MUL.D F5, F2, F0
7	MUL.D F6, F3, F0
8	L.D F7, 0(R2)
9	L.D F8, 8(R2)
10	L.D F9, 16(R2)
11	DADDIU R1, R1, #24
12	ADD.D F10, F4, F7
13	ADD.D F11, F5, F8
14	**DADDIU R2, R2, #24**
15	**ADD.D F12, F6, F9**
16	DSLTU R3, R1, R4
17	S.D F10, -24(R2)
18	S.D F11, -16 (R2)
19	BNEZ R3, foo
20	S.D F12, -8(R2)

- The 3 unrolled instructions take 19 cycles.
- Therefore, the cycle times per element is 19/3 cycles.

- **Grading policy**
- -2 point, stall is not correct or stall is missing after BNEZ\
- -2 point, if the offset is not correct or register is used wrong.
- -5 point, if the unrolling is not 3 times.
- Since the first instruction is not the part of the loop, it can be ignored.
-

Questions 2

2.A (30 points)

1. assume when issuing the instruction into reservation station, first set the busy flag in that reservation station. (at the same time, it occupy the reservation station)

2. assume reservation station is free at write CDB station, and an instruction can get in there at the same cycle.

3. assume that there is no branch penalty. The next correct instruction can be issued in next cycle.

Note that we focused on stall cycle calculation and functional unit states which are important factor for this question. **Therefore**, **we accept various assumptions about starting pipeline stage, branch penalty, Etc**.

The candidate answer is described as below:

iteration	instruction	issues	executes	Memory access	Write CDB	comment
1	L.D F2,0(R1)	1		2	3	[1-2] L.D buffer
1	MUL.D F4,F2,F0	2	4		19	[3-3] waiting F2 [2-18] FP mult reservation station
1	L.D F6,0(R2)	3		4	5	[3-4] L.D buffer
1	ADD.D F6,F4,F6	4	20		30	[5-5] waiting F6 [5-19] waiting F4 [4-29] FP add reservation station
1	S.D F6,0(R2)	5		31		[6-30] waiting F6 [5-31] S.D buffer
1	DADDIU R1,R1,#8	6	7		8	[6-7] int reservation station
1	DADDIU R2,R2,#8	7	8		9	[7-8] int reservation station
1	DSLTU R3,R1,R4	8	9		10	[8-9] int reservation station
1	BNEZ R3,foo	9	11			[10-10] Waiting R3 [9-11] int reservation station
2	L.D F2,0(R1)	10		12	13	[11-11] Waiting branch [10-12] L.D buffer
2	MUL.D F4,F2,F0	11	19		34	[12-18] FP multiplier busy

						[12-13] waiting F2 [11-33] FP multi reservation station
2	L.D F6,0(R2)	12		13	14	[12-13] L.D buffer
2	ADD.D F6,F4,F6	13	35		45	[14-14] waiting F6 [14-34] waiting F4 [13-44] FP add reservation station
2	S.D F6,0(R2)	14		46		[15-45] Waiting F6 [14-46] S.D buffer
2	DADDIU R1,R1,#8	15	16		17	[15-16] int reservation station
2	DADDIU R2,R2,#8	16	17		18	[16-17] int reservation station
2	DSLTU R3,R1,R4	17	18		19	[17-18] int reservation station
2	BNEZ R3,foo	18	20			[19-19] Waiting R3 [18-20] int reservation station
3	L.D F2,0(R1)	19		21	22	[20-20] waiting branch [19-21] L.D buffer
3	MUL.D F4,F2,F0	20	34		49	[21-22] waiting F2 [21-33] FP multiplier busy [20-48] FP multi reservation station
3	L.D F6,0(R2)	21		22	23	[21-22] L.D buffer
3	ADD.D F6,F4,F6	22	50		60	[23-23] waiting F6 [23-49] waiting F4 [22-59] FP add reservation station
3	S.D F6,0(R2)	23		61		[24-60] waiting F6 [23-61] S.D buffer
3	DADDIU R1,R1,#8	24	25		26	[24-25] int reservation station
3	DADDIU R2,R2,#8	25	26		27	[25-26] int reservation station
3	DSLTU R3,R1,R4	26	27		28	[26-27] int reservation station
3	BNEZ R3,foo	27	29			[27-29] int reservation station [28-28] waiting R3

- **Grading policy**
- -2 point, dependency is not preserved or worng
- -2 point, access the functional unit at the same time (not pipelined)
- -2 point, write CDB cycle is wrong
- -2 point, execute/memory access cycle is wrong
- Any branch stall cycle is accepted (because there is no restirction about it)
- The issue cycle can start at 1 cycle or 3 cycle (if the assumption is correct)

Questions 3.

3.A (10 points)

Since target instruction is stored in the buffer, there is no penalty cycles for unconditional branches. Furthermore, we can save one cycle in this condition because unconditional branch instructions can directly fetched from buffer (eliminate IF stage).

- **Grading policy**
- -2 point, if the answer is wrong

3.B (10 points)

Followings are given condition for this sub-questions.

- Comparing BTB scheme with the branch folding.
- Buffer hit rate: **90%**
- Unconditional branch frequency: **5%**
- Buffer miss penalty: **2 cycles**

Since the branch folding saves -1 cycle for unconditional branch instructions, penalty for each condition is calculated as follows:

Penalty cycles for unconditional branch with branch folding:

- $5\% \times (90\% \times -1 + 10\% \times 2) = 0.05 \times (0.9 \times -1 + 0.1 \times 2) = -0.035$

Penalty cycles for unconditional branch without branch folding:

- $5\% \times (90\% \times 0 + 10\% \times 2) = 0.05 \times (0.9 \times 0 + 0.1 \times 2) = 0.01$

Therefore, **branch folding is more efficient for given conditions**.
For the performance gain, we can find the brake even point using fowling equation.
Since the comparator is not clear, any comparator with correct reason is accepted
Here is some example.

1. $0.05 \times (x\% \times -1 + (1-x)\% \times 2) < 0, x = 66.7\ \%$ (Ideal Case)
2. $0.05 \times (x\% \times -1 + (1-x)\% \times 2) < 0.01$, x= 60 % (unconditinal branch without branch folding when hit rate 90%)
3. $0.05 \times (x\% \times -1 + (1-x)\% \times 2) < 0.05 \times (x\% \times 0 + (1-x)\% \times 2)$, x= 0 % (unconditinal branch without branch folding when hit rate is same (x%))

- **Grading policy**
- 5 points : improvement calcaultation
  - -2 point, if the calculation is wrong.
- 5 points : hit rate calculation (since the comparator can be Ideal case or unconditonal branch without

branch folding)

- -2 point, if the calculation is wrong.
- -2 point, if there is no equation.