Neilesh Sambhu U90484573

10/28/2018

EEL 6764

HW 4

1

3.1

Assume code starts at Loop and not I0.

Instruction	Clock cycles
Loop	1+3=4
I0	1+4=5
I1	1+10=11
I2	1+3=4
I3	1+2=3
I4	1+2=3
I5	1+1=2
I6	1+0=1
I7	1+0=1
I8	1+0=1
I9	1+1=2

Including the branch stall, total clock cycles for one iteration: 37

If we assume that the loop is operating at steady state, the total clock cycle for one iteration is only 36 cycles because the prediction scheme is predict-taken and the one-cycle branch delay slot will ensure that useful work (in the form of the fetching of an independent instruction) is done in the stall cycle.

3.2

```
Loop:   fld f2,0(Rx)
<stall>
<stall>
<stall>
I0:     fmul.d f2,f0,f2
<stall>
<stall>
<stall>
<stall>
I1:     fdiv.d f8,f2,f0
I2:     fld f4,0(Ry)
<stall>
<stall>
<stall>
I3:     fadd.d f4,f0,f4
```

```
<stall>
<stall>
<stall>
<stall>
<stall>
I4:     fadd.d f10,f8,f2
I5:     fsd f4,0(Ry)
I6:     addi Rx,Rx,8
I7:     addi Ry,Ry,8
I8:     sub x20,x4,Rx
I9:     bnz x20,Loop
```

Assume branch prediction scheme is predict-taken and the one-cycle delay slot does not increase the total clock cycles.

Total clock cycles: 26 cycles

3.5

```
Loop:   fld f2,0(Rx)
I6:     addi Rx,Rx,8
<stall>
<stall>
I0:     fmul.d f2,f0,f2
I8:     sub x20,x4,Rx
<stall>
<stall>
<stall>
I1:     fdiv.d f8,f2,f0
I2:     fld f4,0(Ry)
<stall>
<stall>
<stall>
I3:     fadd.d f4,f0,f4
<stall>
<stall>
I5:     fsd f4,0(Ry)
I7:     addi Ry,Ry,8
<stall>
I4:     fadd.d f10,f8,f2
I9:     bnz x20,Loop
<stall>
```

Assume branch prediction scheme is predict-taken and the one-cycle delay slot does not increase the total clock cycles. Assume that we need to have 1 stall immediately after the branch for the latency to determine the branch outcome. **There are multiple orderings/permutations of the moved instructions that will result in the same clock cycle count. However, this ordering will affect the unrolling process. It is difficult to create unrolled code with a minimum clock cycle count because of this dependence on the ordering of the original, unrolled code.**

Total clock cycles: 23 cycles

3.6

b. Assume unrolling takes place with scheduling, as indicated in "Lecture 4 pipelining part 2" (slide 17). Assume we do not need to reassign registers to prevent collisions between the iterations (as indicated in 3.6c). Assume that we need to have 1 stall immediately after the branch for the latency to determine the branch outcome. Assume the branch instruction is considered as part of the n+1th instruction. Assume the addi instructions from the nth iteration whose outputs are used to reference memory can be deleted or replaced with stalls because we can address the n+1th iterations' memory accesses with offsets and then increment by twice the original value in the subsequent addi instruction for the n+1th iteration. For instructions whose outputs are not used to address memory, we do write the nth iterations' instructions, even if their outputs will be overwritten in the subsequent instruction. Assume that we do not need to reorder the instructions (to reduce the stalls) after performing the unrolling.

```
Loop:   fld f2,0(Rx)
fld f2,8(Rx)
<stall>
I6:     addi Rx,Rx,16
I0:     fmul.d f2,f0,f2
I0:     fmul.d f2,f0,f2
I8:     sub x20,x4,Rx
I8:     sub x20,x4,Rx
<stall>
I1:     fdiv.d f8,f2,f0
I1:     fdiv.d f8,f2,f0
I2:     fld f4,0(Ry)
I2:     fld f4,0(Ry)
<stall>
<stall>
I3:     fadd.d f4,f0,f4
I3:     fadd.d f4,f0,f4
<stall>
I5:     fsd f4,0(Ry)
I5:     fsd f4,8(Ry)
I7:     addi Ry,Ry,16
I4:     fadd.d f10,f8,f2
I4:     fadd.d f10,f8,f2
I9:     bnz x20,Loop
<stall>
```

c.

Original: 23 cycles/iteration

Unrolled: 25 cycles/2 iterations = 12.5 cycles/iteration

$$Speedup = \frac{23}{12.5}$$

$$Speedup = 1.84$$

3.11

Let "X" indicate a stall. Assume forwarding from writeback to instruction decode is not possible. Assume that when a data hazard is discovered in the ID stage, the pipeline always stalls through the writeback stage of instruction whose output is needed. Assume we can fetch the correct instruction in the stage immediately after the second execute stage of a branch.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Loop: lw x1,0(x2)	IF	ID	EX	MEM	MEM	MEM	WB																
addi x1,x1,1		IF	ID	X	X	X	X	EX	MEM	WB													
sw x1,0(x2)			IF	X	X	X	X	ID	X	X	EX	MEM	MEM	MEM	WB								
addi x2,x2,4				X	X	X	X	IF	X	X	ID	EX	X	X	MEM	WB							
sub x4,x3,x2					X	X	X	X	X	X	IF	ID	X	X	X	X	EX	MEM	WB				
bnz x4,Loop						X	X	X	X	X	X	IF	X	X	X	X	ID	X	X	EX	EX	MEM	WB

a. If the Loop instruction were to begin immediately after the sub, the Loop instruction's instruction fetch would occur in clock cycle 12. Currently, with the overhead of the branch, the Loop instruction's instruction fetch begins in instruction 22.
$22\ cycles - 12\ cycles = 10\ cycles$
Therefore, the branch overhead is 10 clock cycles.

b. If the Loop instruction were to begin immediately after the sub, the Loop instruction's instruction fetch would occur in clock cycle 12. If we are able to determine a backwards branch in the decode stage, the Loop instruction's instruction fetch would occur in clock cycle 20.
$20\ cycles - 12\ cycles = 8\ cycles$
Therefore, the branch overhead is 8 clock cycles.

c. If the Loop instruction were to begin immediately after the sub, the Loop instruction's instruction fetch would occur in clock cycle 12. During a correct prediction, the Loop instruction's instruction fetch would occur in clock cycle 17.
$17\ cycles - 12\ cycles = 5\ cycles$
Therefore, the branch overhead is 5 clock cycles.

2

a. Assume the Branch History Table has a size of 1.

Index	Bit Starting Value	Bit Ending Value	Prediction/Misprediction
1	0	1	Miss
2	1	1	Not miss
3	1	0	Miss
4	0	1	Miss
5	1	0	Miss
6	0	1	Miss

7	1	1	Not miss
8	1	1	Not miss
9	1	1	Not miss
10	1	0	Miss
11	0	1	Miss
12	1	1	Not miss
13	1	0	Miss

Mispredicted branches: 1,3,4,5,6,10,11,13

a. Assume the Branch History Table has a size of 1.

Index	Bit Starting Value	Bit Ending Value	Prediction/Misprediction
1	10	11	Not miss
2	11	11	Not miss
3	11	10	Miss
4	10	11	Not miss
5	11	10	Miss
6	10	11	Not miss
7	11	11	Not miss
8	11	11	Not miss
9	11	11	Not miss
10	11	10	Miss
11	10	11	Not miss
12	11	11	Not miss
13	11	10	Miss

Mispredicted branches: 3,5,10,13