CS M151B - Homework 6

Kuruvilla Cherian, Jahan
UID: 104436427
Section 1B

March 4, 2017

Loop Unrolling

Part 1

```
Loop:
        LW $t0, 0($s0)
        LW $t1, 0($t0)
        ADD $t1, $s1, $t1
        SW $t1, 0($t0)
        ADDI $s0, $s0, 4
        BNE $s0, $s2, Loop
```

To begin analyzing the number of cycles per iteration of this loop we look for data dependencies. We clearly see the data dependency between the first two load words on the register **$t0**. The ADD instruction is dependent on the second load word on the register **$t1**, which then follows to a data dependency for the store word on the same register. Finally we notice the dependency between the ADDI and the BNE on register **$s0**. Because of full data forwarding in our processor, we are able to schedule the instructions such that there is some overlap in cycles. For example we can schedule the ADD instruction two cycles ahead of the second LW since by that point we can forward the MEM result from LW to the EX stage of ADD.

Utilizing the delay slots we also see that we can add independent instructions to the BNE to reduce the use of unnecessary *nops*. We thus get the scheduled cycles as shown in Figure 1 below.

Cycle	Issue Slot 1 (ALU/Branch)	Issue Slot 2 (LW/SW)
1	ADDI $s0, $s0, 4	LW $t0, 0($s0)
2	NOP	NOP
3	NOP	LW $t1, 0($t0)
4	BNE $s0, $s2, Loop	NOP
5	ADD $t1, $s1, $t1	NOP$_{BNE}$
6	NOP$_{BNE}$	SW $t1, 0($t0)

Figure 1: Instruction scheduling with reordering, full forwarding and branch delay slots

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
ADDI $s0, $s0, 4	IF	ID	EX	MEM	WB					
LW $t0, 0($s0)	IF	ID	EX	MEM	WB					
NOP		IF	ID	EX	MEM	WB				
NOP		IF	ID	EX	MEM	WB				
NOP			IF	ID	EX	MEM	WB			
LW $t1, 0($t0)			IF	ID	EX	MEM	WB			
BNE $s0, $s2, Loop				IF	ID	EX	MEM	WB		
NOP				IF	ID	EX	MEM	WB		
ADD $t1, $s1, $t1					IF	ID	EX	MEM	WB	
NOP$_{BNE}$					IF	ID	EX	MEM	WB	
NOP$_{BNE}$						IF	ID	EX	MEM	WB
SW $t1, 0($t0)						IF	ID	EX	MEM	WB

Figure 2: Pipeline diagram representing the re-scheduled loop

As we see from Figure 1 and Figure 2 by the sixth cycle of the iteration the BNE will resolve the branch by the EX stage, upon which we can begin the next iteration. The only caveat is for the very final iteration, wherein we have to wait for the entire completion of the iteration, that is wait until the 10th cycle of that iteration for the SW to finish it's write back.

Thus we have the following number of cycles for 200 iterations:

$$num\_cycles = 199 \cdot 6 + 1 \cdot 10 = \mathbf{1204\ cycles}$$

2

```
Loop:
        LW $t0, 0($s0)
        LW $t1, 0($t0)
        ADD $t1, $s1, $t1
        SW $t1, 0($t0)
        LW $t2, 4($s0)
        LW $t3, 0($t2)
        ADD $t3, $s1, $t3
        SW $t3, 0($t2)
        ADDI $s0, $s0, 8
        BNE $s0, $s2, Loop
```

When applying loop unrolling we begin by noticing that the ADDI and BNE are the two key components of the entire loop that only needs to run once. Because we simply unroll once, we are only required to repeat the loads, add and stores once more. Because of this unrolling, our dependent registers - such as **$s0** needs to be offset by the proportion by which the overall unroll jumps by. Because we have unrolled once, and thus increased the number of instructions by four instructions, we overall offset by 8 per iteration. Because we see data dependencies between the **$t0** and **$t1** within the unroll, we use *register renaming* to rename the registers in the unroll to **$t2** and **$t3**. Taking advantage of the delay slots and this unroll, we get the following cycle schedule as shown in Figure 3.

Cycle	Issue Slot 1 (ALU/Branch)	Issue Slot 2 (LW/SW)
1	ADDI \$s0, \$s0, 8	LW \$t0, 0(\$s0)
2	NOP	LW \$t2, -4(\$s0)
3	NOP	LW \$t1, 0(\$t0)
4	NOP	LW \$t3, 0(\$t2)
5	ADD \$t1, \$s1, \$t1	NOP
6	BNE \$s0, \$s2, Loop	NOP
7	ADD \$t3, \$s1, \$t3	SW \$t1, 0(\$t0)
8	NOP_{BNE}	SW \$t3, 0(\$t2)

Figure 3: Instruction scheduling with reordering, full forwarding and branch delay slots for unrolled version

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
ADDI \$s0, \$s0, 8	IF	ID	EX	MEM	WB							
LW \$t0, 0(\$s0)	IF	ID	EX	MEM	WB							
NOP		IF	ID	EX	MEM	WB						
LW \$t2, -4(\$s0)		IF	ID	EX	MEM	WB						
NOP			IF	ID	EX	MEM	WB					
LW \$t1, 0(\$t0)			IF	ID	EX	MEM	WB					
NOP				IF	ID	EX	MEM	WB				
LW \$t3, 0(\$t2)				IF	ID	EX	MEM	WB				
ADD \$t1, \$s1, \$t1					IF	ID	EX	MEM	WB			
NOP					IF	ID	EX	MEM	WB			
BNE \$s0, \$s2, Loop						IF	ID	EX	MEM	WB		
NOP						IF	ID	EX	MEM	WB		
ADD \$t3, \$s1, \$t3							IF	ID	EX	MEM	WB	
SW \$t1, 0(\$t0)							IF	ID	EX	MEM	WB	
NOP_{BNE}								IF	ID	EX	MEM	WB
SW \$t3, 0(\$t2)								IF	ID	EX	MEM	WB

Figure 4: Pipeline diagram representing the re-scheduled loop with unrolling

Figure 4 highlights the overall pipeline diagram for the unrolled version of the loop. For all iterations barring the last one, we see the branch gets resolved by the 8^{th} cycle. We must also pay attention to the fact that the number of iterations is reduced by 2, and the very last iteration must run to completion until the store word is written back, giving us the following number of total cycles:

$$num\_cycles = 99 \cdot 8 + 1 \cdot 12 = \mathbf{804\ cycles}$$