Assignment-3: Instruction Level Parallelism

Arnab Das, u1014840

February 25, 2018

In the optimizations below, **assumption** has been made of an **infinite/large enough register space** for both **integer and FP pipelines**.

1 *Question 1*

The Base case with the stalls inserted looks like

```
Loop :  L.D    F1,   0(R1)
        L.D    F2,   0(R2)
        Stall
        MUL    F3,   F1,   F2
        Stall
        Stall
        Stall
        Stall
        S.D    F3,   0(R3)
        DADDUI R1,   R1,   #-8
        DADDUI R2,   R2,   #-8
        DADDUI R3,   R3,   #-8
        BNE    R1,   R4,   Loop
        NOP
```

i. optimized schedule without unrolling

With the optimized schedule without unrolling we get **1 Stall**. The first stall is removed by scheduling the $DADDUI$ for $R1$. Similarly, the other stalls are replaced with useful work done by two more $DADDUI$ for $R2$ and $R3$ respectively. Also, the $S.D$ is moved after the BNE to take advantage of the 1 branch-delay slot. Ultimately we

are left with 1 stall.

```
Loop :  L.D    F1,   0(R1)
        L.D    F2,   0(R2)
        DADDUI R1,   R1,   #-8
        MUL    F3,   F1,   F2
        DADDUI R2,   R2,   #-8
        DADDUI R3,   R3,   #-8
        Stall
        BNE    R1,   R4,   Loop
        S.D    F3,   8(R3)
```

ii. With unrolling

Optimizing with unrolling 2 times, results in **no stall cycles** in the schedule. Here the first stall is removed by executing one more Load for the additionally unrolled statements for the loop. Successive stalls are removed by fetching one more load for the unrolled section of the loop and also the *DADDUI* instructions for iterator decrement. Here as well, the last store is moved after the *BNE* to take advantage of the branch delay slot.

```
Loop :  L.D    F1, 0(R1)
        L.D    F2, 0(R2)
        L.D    F11, -8(R1)
        MUL    F3,   F1,   F2
        L.D    F22,   -8(R2)
        DADDUI R1,   R1,   #-16
        MUL    F33,  F11,  F22
        DADDUI R2,   R2,   #-16
        S.D    F3,   0(R3)
        DADDUI R3,   R3,   #-16
        BNE    R1,   R4,   Loop
        S.D    F33,  8(R3)
```

iii. Software pipelined

In the software pipelined case, the inner kernel without the prologue and the epilogue comprises of no stalls. Here, during the store of the

$i'th$ iteration, the MUL operation of $(i-1)'th$ iteration and the $LOAD$ for the $(i-2)'th$ iteration is executed.

```
Loop :  S.D    F3,   16(R3)
        MUL    F3,   F1,   F2
        L.D    F1,   0(R1)
        L.D    F2,   0(R2)
        DADDUI R1,   R1,   #-8
        DADDUI R2,   R2,   #-8
        DADDUI R3,   R3,   #-8
        BNE    R1,   R4,   Loop
```

2 *Question 2*

Optimized without unrolling

The Base case with stalls will be

```
Loop :  L.D    F2,   0(R2)
        L.D    F3,   0(R3)
        Stall
        MULT.D F1,   F2,   F3
        L.D    F4,   0(R4)
        Stall
        ADD.D  F5,   F4,   F1
        Stall
        Stall
        Stall
        Stall
        S.D    F5,   0(R5)
        DADDUI R2,   R2,   #-8
        DADDUI R3,   R3,   #-8
        DADDUI R4,   R4,   #-8
        DADDUI R5,   R5,   #-8
        BNE    R2,   R1,   Loop
        NOP
```

Optimizing without unrolling results in **no stall** cycles. The mechanism to remove stalls is similar as earlier.

```
Loop :  L.D    F2,  0(R2)
        L.D    F3,  0(R3)
        L.D    F4,  0(R4)
        MUL    F1,  F2,  F3
        ADD    F5,  F4,  F1
        DADDUI R2,  R2,  #-8
        DADDUI R3,  R3,  #-8
        DADDUI R4,  R4,  #-8
        DADDUI R5,  R5,  #-8
        BNE    R2,  R1,  Loop
        S.D    F5,  8(R5)
```

3 *Question 3*

i. Optimized without unrolling

Optimized without unrolling. There 5 stalls in the integer pipeline and 1 stall in the FP pipeline. Also, we have a situation of both pipelines being stalled.

```
Loop :  L.D    F1,  0(R1)
        DADDUI R1,  R1,  #-8
        DADDUI R2,  R2,  #-8        F.MUL  F3,  F1,  F2
        Stall                       Stall
        Stall                       F.ADD  F5,  F3,  F4
        Stall
        Stall
        Stall
        BNE    R1,  R3,  Loop
        S.D    F5,  8(R2)
```

ii. Optimized with loop unrolling

The **limiting case for unrolling with 4 stalls** is shown here which still has **1 stall** concurrently in each pipeline. Then the case for **unrolling 5 times** is shown that removes the stalls .

unroll 4

Loop :	L.D F1, 0(R1)	
	L.D F11, -8(R1)	
	L.D F12, -16(R1)	F.MUL F3, F1, F2
	L.D F13, -24(R1)	F.MUL F31, F11, F2
	DADDUI R1, R1, #-32	F.MUL F32, F12, F2
	DADDUI R2, R2, #-32	F.MUL F33, F13, F2
	Stall	F.ADD F5, F3, F4
	Stall	F.ADD F51, F31, F4
	Stall	F.ADD F52, F32, F4
	Stall	F.ADD F53, F33, F4
	Stall	
	S.D F3, 32(R2)	
	S.D F31, 24(R2)	
	S.D F32, 16(R2)	
	BNE R1, R3, Loop	
	S.D F33, 8(R2)	

Note that, even for unroll of 4, by moving one of the *DADDUI* instructions to the location where both pipelines are stalled, we could remove the stall by making the integer pipeline execute, but that is not an optimal schedule since that introduces an earlier stall where some useful work could have been done.

unroll 5 The issue of both pipelines being stalled at the same time is solved here with **unrolling** it **5 times**.

Loop :	L.D F1, 0(R1)	
	L.D F11, -8(R1)	
	L.D F12, -16(R1)	F.MUL F3, F1, F2
	L.D F13, -24(R1)	F.MUL F31, F11, F2
	L.D F14, -32(R1)	F.MUL F32, F12, F2
	DADDUI R1, R1, #-40	F.MUL F33, F13, F2
	DADDUI R2, R2, #-40	F.MUL F34, F14, F2
	Stall	F.ADD F5, F3, F4
	Stall	F.ADD F51, F31, F4
	Stall	F.ADD F52, F32, F4
	Stall	F.ADD F53, F33, F4
	Stall	F.ADD F54, F34, F4
	S.D F3, 40(R2)	
	S.D F31, 32(R2)	
	S.D F32, 24(R2)	
	S.D F33, 16(R2)	
	BNE R1, R3, Loop	
	S.D F34, 8(R2)	

With unrolling 5 times, it is able to execute 23 instructions in 18 cycles, resulting in an **IPC =** 1.27. Unrolling it further 6 times, results in an **IPC =** 1.35. Attempting until 10 unrolls, gave an **IPC =**1.53, wheres the base case optimized schedule without unrolling had an **IPC = 0.7**.