COMP3211 Tutorial
Week 14
Jeremy Chan
jeremyc@cse.unsw.edu.au

Q01. For Assignment 3, discuss which signals should be stimulated by any testbench that is designed to properly simulate the operation of the multi-cycle version of the datapath? Which signals should you produce waveforms for to provide a minimal yet complete and comprehensive demonstration of the revised design?

To test that the multi-cycle version of data-path is working correctly, the signals that you probably want to look at are the control signals and the outputs and the inputs of the data path registers. These will help you determine whether the correct values are being written at the expected clock cycles.

To demonstrate that it is correctly working, you should at least produce the multi-cycle data path registers that hold the values between clock cycles and the register file values to show that values are being written back.

It would probably be useful to also have the clock signal and reset signal as references.

Q02. What factors limit the performance of a pipelined datapath as the depth of pipelining is increased?

Pipeline CPI = Ideal pipeline CPI + Structural stalls + Data hazard stalls + Control stalls

As the number of pipeline stages increases, the clock cycle period decreases. As we increases the depth of the pipeline there are more operations in flight, and hence it become increasingly difficult to minimize the effect of stalls due to hazards. This can be overcome to some extent by pipeline scheduling techniques but as the issue width increases, this becomes increasingly difficult and the amount of parallelism that can be exploited reaches a limit.

Additional to this, as the amount of logic between pipeline stages decreases, a greater fraction of the cycle time is made up of register setup and hold times and allowances for clock skew.

Deeper pipelines have additional power penalties, resulting from several sources. The most important of these is the simple observation that a deeper pipeline means more operations are in flight every clock cycle, which means more transistors are switching, which means more power!

Q03. The last slide of Lecture 25 (Week 13) contains a program excerpt. Assume this program is to run on the 5 stage pipeline you are familiar with but without enhancements for data or control hazards.

How many cycles are needed to execute this code including the necessary stalls assuming the branch is taken once? (You should assume the branch direction is determined in the execute stage and the PC is updated in the memory access stage.)

Describe the hazards in the code, and describe how you would eliminate or minimize them. Reassess the time needed to execute the code after the enhancements you describe have been incorporated.

The code in lecture 25 is as follows:

```
Loop:   lw    $t0,  0($t1)
        add   $t3,  $t0,  $t1
        sub   $t4,  $t0,  $t2
        or    $t5,  $t3,  $t4
        sw    $t1,  0($t1)
        beq   $t1,  $t0,  loop
Exit:   add   $t3,  $t4,  $t5
```

In this code, we have assumed branch not taken.

```
Loop:   lw    $t0,   0($t1)
        add   $t3,   $t0,   $t1          +2
        sub   $t4,   $t0,   $t2
        or    $t5,   $t3,   $t4          max(+2, +1)
        sw    $t1,   0($t1)
        beq   $t1,   $t0,   loop +3
        lw    $t0,   0($t1)
        add   $t3,   $t0,   $t1          max(+2, +1)
        sub   $t4,   $t0,   $t2
        or    $t5,   $t3,   $t4          +2
        sw    $t1,   0($t1)
        beq   $t1,   $t0,   loop
Exit:   add   $t3,   $t4,   $t5
```

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Lw	F	D	E	M	W																								
Add		*1	*1	F	D	E	M	W																					
Sub					F	D	E	M	W																				
Or						*2	*2	F	D	E	M	W																	
Sw									F	D	E	M	W																
Beq										F	D	E	M	W															
add											F	D	FL(3)																
…												F	FL(3)																
Lw													FL(3)	F	D	E	M	W											
Add															*1	*1	F	D	E	M	W								
Sub																		F	D	E	M	W							
Or																			*2	*2	F	D	E	M	W				
Sw																						F	D	E	M	W			
Beq																							F	D	E	M	W		
Add																								F	D	E	M	W	

1. Load Use Hazard
2. Data Hazard
3. Control Hazard

As can be seen from the above diagram, the total number of cycles is 28. This consists of 13 cycles for instructions execution, 4 for pipeline overhead + 11 caused by hazards. There is a 2 cycle penalty for load hazards, 2 cycles for Data Hazards and 3 cycles for Control Hazards.

A data hazard can be removed by adding a forwarding unit reducing the number of cycles of stalling due to data hazards to 0. A load use and control hazards can be reduced to 1 cycle when a forwarding unit is used. This reduces the total number of cycles of this code to 13 + 3 + 4 (1 cycle for the single control hazard and 2 cycles for the 2 load use hazards). With proper pipeline scheduling, this can be reduced to 17 cycles through the use of branch delay and load delay slots (13 + 4).

The code can be rearranged as follows to use branch delay and load delay slots. Here, we have two branch delay slots and one load delay slot.

```
Loop:   lw    $t0,   0($t1)
        sw    $t1,   0($t1)
        add   $t3,   $t0,   $t1
        beq   $t1,   $t0,   loop
        sub   $t4,   $t0,   $t2
        or    $t5,   $t3,   $t4
Exit:   add   $t3,   $t4,   $t5
```