4. You are attempting to improve a pipelined CPU’s performance by splitting the data memory stage (the slowest stage) into two stages, thereby increasing the clock speed. However, splitting data memory into two stages introduces the possibility of some lw instructions generating two stalls instead of one.

(a) Explain how splitting the data memory section into two stages might result in extra stalls.

For data forwarding to work with a split of data memory, there will need to be 2 stalls for data memory to finish rather than the usual 1 stall with full forwarding. This is because the data pulled from memory will need two cycles to finish thus needing 2 stalls.

(b) Will sw instructions generate any additional stalls? Why or why not?

Typically data is not waiting on memory after a instruction, so generally an SW instruction will not generate any additional stalls. The only command that would need the result after a sw would be a lw, which by the time the sw is finished up the result of the sw will be available.

(c)On average, how many stalls can each lw generate and still have the new CPU outperform the old CPU?

•The clock speed of the current pipelined CPU is 600ps.

•The clock speed of the new pipelined CPU is 580ps.

•25% of the instructions are lw.

•15% of the lw instructions generate a stall on the current pipelined CPU.

600 n > 580 n \* 1 + (.25)(2x + 1y)

Z = (2x + y)

600n > 580n \*(1 + .25Z)

600 > 580 + 145Z

20 > 145 Z

Z < 0.1379

So the lw must generate additional stalls less that 13.79% of the time to be as efficient as the other CPU.

(d) At most, what percentage of Lw instructions can generate a stall before the new CPU fails to outperform the old CPU?

0.1379 > (2x + y)

Y < 0.1379 & x = 0

At most, an lw instruction can generate 1 extra stall only 13.79% of the time (or less) before the new CPU fails to outperform the new CPU.

(e) At most, what percentage of Lw instructions can generate two stalls before the new CPU fails to outperform the old CPU?

0.1379 > (2x + y)

X < 0.0689 & y = 0

At most, an lw instruction can generate 2 extra stalls only about 6.89% of the time (or less) before the new CPU fails to outperform the new CPU.

(f) Suppose that lw instructions generate one stall three times as often as they generate two stalls. What percentage of Lw instructions can generate one stall before the new CPU fails to outperform the old CPU?

Since: 3x = y

X = .034475

Y = .103425

A lw can generate a stall about 10.34% of the time and still be as efficient a CPU as its predecessor.

7. Consider a version of the pipeline from the textbook that has no forwarding. Suppose that this CPU has a cycle time of 250.

(a) Suppose that adding forwarding to this CPU will increase the cycle time to 300, but reduce the CPI from 1.4 to 1.05. What is the speedup of this new pipeline compared to the one without forwarding?

T1 = 250\*n\*(1.4)

T2 = 300\*n\*(1.05)

Speedup = t1/t2

= 250 (1.4) n / 300 (1.05) n

= 350/315

= 1.111

So, the speedup of the new pipeline is greater than the old pipeline by a factor of 1.111.

(b) What is the maximum CPI that will allow the CPU with forwarding to perform better than the CPU without forwarding?

250 (1.4) n > 300 x n

350 > 300 x

1.16667 > x

So the CPI can be as bad as 1.166667 and still perform as well as the old CPU.

(c) A different program has a CPI of 1.25 on the original pipeline. What is the maximum CPI that will allow the CPU with forwarding to run this program faster than the original pipeline?

T = 250\*n\*(1.25)

250 (1.25) n > 300 x n

312.5 > 300 x

1.0416 > x

So the new CPU must have a CPI of 1.0416 or less in order to run this program faster than the original pipeline.

(d) We can see from 7b and 7c that the necessary CPI for improvement depends on the particular program. In general, if a program has a CPI of X on original pipeline, what is the maximum CPI it can have when running on the pipeline with forwarding?

T = 250\*n\*(x)

250 x n > 300 \* CPI \* n

250 x> 300 CPI

5/6x > CPI

So the CPI of the new cpu must be smaller than 5/6x of where x is the old CPI.

(e) Can a program with a CPI of 1.075 possibly run faster on the pipeline with forwarding? Explain why or why not.

5/6 (1.075) = 0.8958

This shows us that the new CPU would always run faster than a forwarding pipeline CPU because the overhead of the 50ps from forwarding resulted in a need of a CPI of 0.8958 or less, which is not possible.

(f) What is the minimum CPI a program must have before it can possibly run faster on the pipeline with forwarding?

5/6x = 1

X = 1.2