Assignment # 2

Project

1. In this problem, we will explore how deepening the pipeline affects performance in two ways: faster clock cycle and increased stalls due to data and control hazards. Assume that the original machine is a 5-stage pipeline with a 1 ns clock cycle. The second machine is a 14-stage pipeline with a 0.5 ns clock cycle. The 5-stage pipeline experiences a stall due to a data hazard every 5 instructions, whereas the 14-stage pipeline experiences 3 stalls every 8 instructions. In addition, branches constitute 20% of the instructions, and the misprediction rate for both machines is 15%.

   (a) What is the speedup of the 14-stage pipeline over the 5-stage pipeline, taking into account only data hazards?

   (b) If the branch mispredict penalty for the first machine is 2 cycles but the second machine is 5 cycles, what are the CPIs of each, taking into account the stalls due to branch mispredictions?

2. A processor implements a five-stage instruction pipeline that implements full forwarding. The following code are executed using this processor and predict always taken is employed as the branch prediction technique.

```
        LD      R1, 24(R5)
        DADD    R2, R4, R1
        BEQ     R2, R6, Dest
        DADDUI  R1, R1, #1
        LD      R4, 4(R5)
Dest:   DADD    R3, R2, R4
```

   (a) Suppose the branch prediction is correct (branch is taken). Draw a diagram to show the pipeline stages for execution of instructions 1, 2, 3 and 6. Show the **forward** and **stalls** in the diagram if they are used. What is the speedup factor of this pipelining method when compared with not using pipelining? We assume every instruction takes 5 cycles on non-pipeline design.

   (b) Redraw the diagram if the prediction is wrong (branch is actually not taken). Show the execution steps from instruction 1 to 6. In this case, what is the speedup factor when compared with not using pipelining?

3. This problem explores the impact of Static and Dynamic Branch Predictions. Assume a 5-stage single-pipeline microarchitecture (Fetch, Decode, Execute, Memory, Write back) and the code is a backwards loop. All operations are 1 cycle except the following: LD or SD instruction takes 3 cycles at the stage of memory (MEM) access; branch instruction takes 2 cycles at the execution (EX) stage. Use forwarding and stalls if needed. Show the phases of each instruction per clock cycle for one iteration of the loop. Indicate at which cycle the first instruction of next loop starts.

```
Loop  LD      R3, 0(R5)
      LD      R1, 0(R3)
      DADDI   R1, R1, #1
      DSUB    R4, R3, R2
      SD      R1, 0(R3)
      BNZ     R4, Loop
      LD      R3, 0(R5) ; This is the first instruction of the next loop
```

   (a) Without branch prediction: draw a diagram to show the pipeline stages for one iteration of the loop. How many clock cycles are required per loop iteration (also called loop length)?

   (b) Assume a static branch predictor, capable of recognizing a backwards branch in the **Decode** stage. Then a backwards branch is assumed always taken. Draw a diagram to show the pipeline stages for one iteration of the loop. How many clock cycles are required per loop iteration?

   (c) Assume a dynamic branch predictor which issues "predicted taken" as the branch instruction is **fetched**. Draw a diagram to show the pipeline stages for one iteration of the loop. How many clock cycles are required per loop iteration?

4. Consider the following code:

```
Loop  L.D     F2, 0(R1)
      L.D     F4, -8(R1)
      MUL.D   F6, F2, F4
      S.D     F6, 0(R2)
      DADDUI  R1, R1, #-16
      DADDUI  R2, R2, #-8
      BNE     R1, R3, Loop
```

   Assume the same latency and initiation interval of Fig C.34. Ignore the delay caused by the branch.

   (a) Show the timing diagram of this code with full forwarding hardware. Use timing diagram like that shown in Fig C.5. How many cycles does this loop take to execute?

(b) Use static scheduling to reduce the total number of stalls in the loop. Show the new timing diagram. In this case, how many cycles does this loop take to execute?

(c) Use loop unrolling and static scheduling so that the pipeline does not stall at all. What is the minimum number of unrolls that is required to remove all the stalls? How many cycles does this loop take to execute? How many new registers that you need to implement this unrolling? Show the new timing diagram, represented like that shown in lecture.

(d) Compare the number of clocks required in each case. Assuming infinite unrolls, what is the minimum number of clocks that is required to implement one iteration of THE ORIGINAL loop?