Name: _____

CS451 Pipeline Performance Homework

1. In theory, what is the maximum speedup you can possibly achieve by converting the book's single-cycle CPU into a 5-stage pipelined CPU? Assume

   - the time of the single-cycle CPU is 925,
   - the pipeline stages are separated by registers that take time 50, and
   - a "perfect world" where the CPU can always be divided into 5 even stages.

2. In theory, what is the maximum speedup you can possibly achieve by converting the book's single-cycle CPU into a pipelined CPU of any depth? Assume

   - the time of the single-cycle CPU is 925,
   - the pipeline stages are separated by registers that take time 50, and
   - a "perfect world" where the CPU can always be divided into k even stages.

3. In addition to the assumptions given in Problem 2, also assume that the CPI for a 5-stage pipeline is 1.035 and that the number of stalls increases by 50% with every additional stage. (In other words, assume that the CPI for a 6-stage pipeline is $1+(.035)(1.5)$, the CPI for a 7-stage pipeline is $1+(.035)(1.5)(1.5)$, etc.)

   (a) Develop a formula for the performance of this k-stage pipeline. Your formula should combine the cycle time formula from Problem 2, with the model for stalls discussed above.

   (b) Use your formula to determine the optimal number of pipeline stages. (The math for this problem gets complex. I suggest using gnuplot, Maple, or Wolframalpha.com to graph the solution and make a close visual guess.)

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.

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

   (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.

Name: _____

(d) **Not due for credit:** At most, what percentage of `lw` instructions can generate a stall before the new CPU fails to outperform the old CPU?

(e) **Not due for credit:** At most, what percentage of `lw` instructions can generate two stalls before the new CPU fails to outperform the old CPU?

(f) **Not due for credit:** 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?

5. The single-cycle CPU, as presented in the Harris and Harris text has a clock period of 925ps. The pipelined CPU presented has a clock period of 500ps. The precise speedup of the pipelined CPU depends on the number of stalls and unfilled branch delay slots. Is it possible for a program to ever take longer on the pipelined CPU than on the single cycle CPU?

   - If it is possible, give an example program that will take longer to run on the pipelined CPU.
   - If it is not possible (1) explain why it is not possible and (2) calculate the longest clock period that would still guarantee that the pipelined CPU is faster for all possible programs.

6. Now repeat problem 5 assuming that at most 25% of the instructions are loads and at most 10% of the instructions are branches.

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?

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

   (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 orignal 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 as a CPI of x on original pipeline, what is the maximum CPI it can have when running on the pipeline with forwarding?

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

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