**Homework 8: Branch Prediction**

**1) You are asked to design a pipelined CPU that has no branch penalties at all: It has no branch delay slots, and does not rely on branch prediction.**

**(a) What must you do to implement such a CPU?**

Determine all the branch outcomes before execution.

**(b) What are the major limitations of this design?**

It would take a lot of hardware to already know the branch outcomes before they are

executed.

**2) Suppose that a machine with a 5-stage pipeline uses branch prediction (i.e., no branch delay slots). 15% of the instructions for a given test program are branches, of which 80% are correctly predicted. The other 20% of the branches suffer a 4-cycle mis-prediction penalty. (In other words, when the branch predictor predicts incorrectly, there are four instructions in the pipeline that must be discarded.) Assuming there are no other stalls, develop a formula for the number of cycles it will take to complete n lines of this program.**

Cycles = n(0.85 + 0.15(0.80 + 0.20 \* 4))

Cycles = 1.09n

**3. Now suppose you are given the option of replacing this processor’s branch prediction scheme with a 1-cycle branch delay system (i.e., there is one branch delay slot after every branch). What percentage of the branch delay slots must be filled in order for the CPU with the branch-delay system to have better performance than the CPU described in question 2?**

Cycles = n(0.85 + 0.15(0.80 \* 2 + 0.20 \* 3))

**4. You have a pipelined CPU that uses branch prediction. The penalty for a mis-predict is**

**4. Your employer proposes doubling the pipeline depth. Doing so will reduce the clock rate by 25%; but will double the branch penalty. Your task is to figure out if this design change will improve performance. The performance improvement depends on two factors: the accuracy of the branch predictor, and the percentage of instructions that are branches. Let b be the percentage of instructions that are branches. Let m be the branch predictor’s mis-predict rate.**

**(a) Develop (and simplify) a formula that describes the performance of the original CPU in terms of b and m.**

N\* P(1+4bm)

**(b) Develop (and simplify) a formula that describes the relative performance of the new CPU in terms of b and m.**

0.75N\*P(1+8bm)

Speedup = N\* P(1+4bm) / 0.75N\*P(1+8bm)

**(c) Develop (and simplify) a formula that determines how accurate the branch predictor must be given the percentage of instructions that are branches. (In other words, generate a formula that solves for m in terms of b.)**

NP(1+4mb) < 0.75NP(1+8mb)

1+4mb < 0.75(1+8mb)

1+4mb < 0.75+2mb

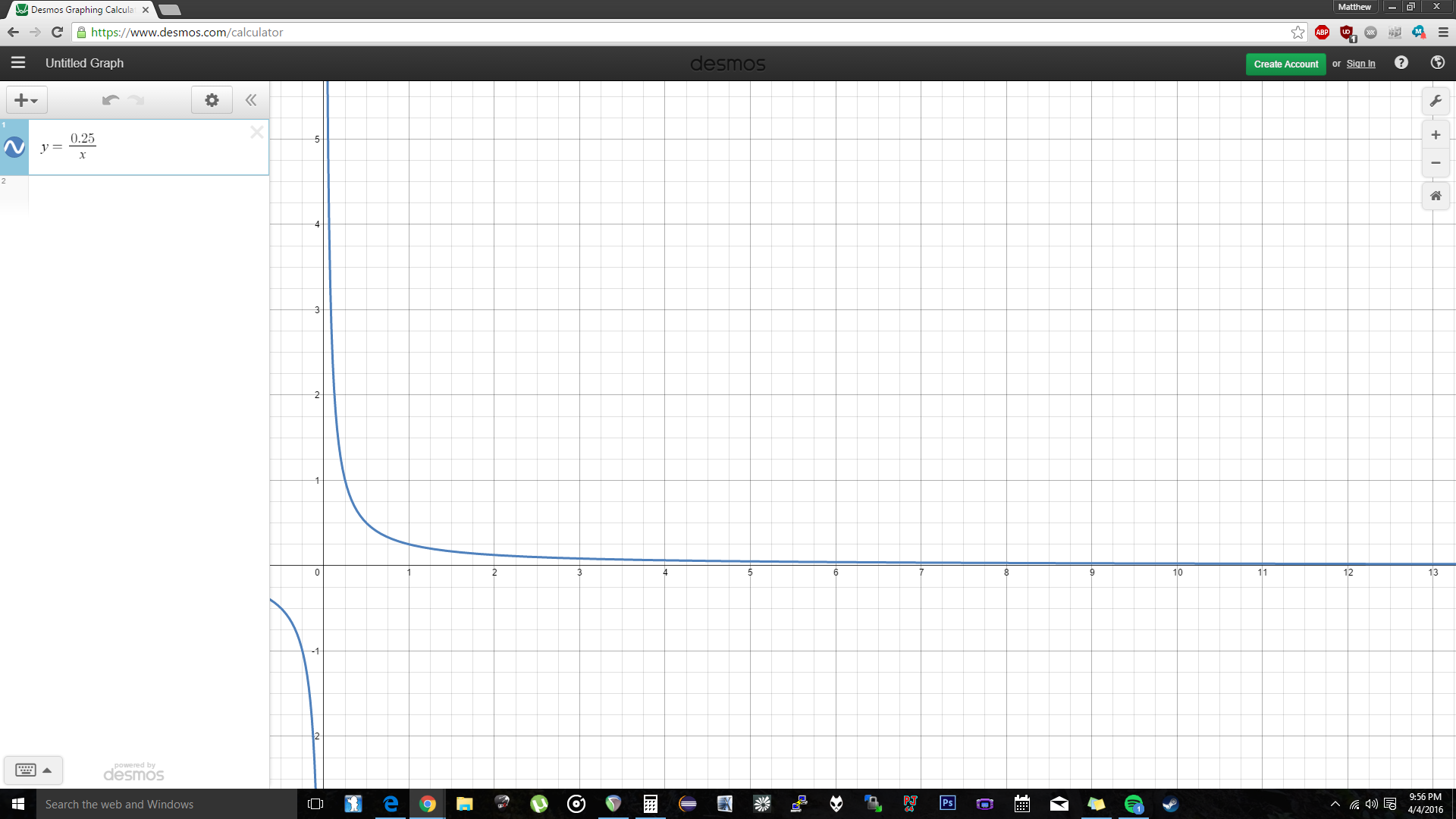
1+2mb < 0.75

2mb < 0.25

mb < 0.125

m < (⅛)/b

Graph of m < (⅛)/b



**(d) At what accuracy does the percentage of branch instructions become irrelevant?**

The percentage of branch instructions becomes irrelevant as b approaches ⅛.

**(e) At what what point does the accuracy of the branch predictor become irrelevant?**

The accuracy of the branch predictor becomes irrelevant whenever it is less than ⅛ b.

**5. Problem 6.36 from Patterson and Hennessy: Compute the predictions for each predictor and branch pattern. Specifically, indicate how each branch will be predicted, then give the hit rate. Use the 2-bit predictor shown in Figure 1.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Predictor** | **TTT** | **NNN** | **TNTNTN** | **TTTNT** | **TTNTTNT** |
| **Always taken** | 100% hit | 0% hit | 50% hit | 80% hit | 0.71% hit |
| **Always not taken** | 0% hit | 100% hit | 50% hit | 20% hit | 0.28% hit |
| **1 bit predictor**  **To predict taken** | 100% hit | 66% hit | 17% hit | 60% hit | 43% hit |
| **2 bit predictor**  **To predict weakly taken** | 100% hit | 66% hit | 50% hit | 80% hit | 0.71% hit |
| **2 bit predictor**  **To predict strongly taken** | 100% hit | 33% hit | 50% hit | 80% hit | 0.71% hit |
| **2 bit predictor**  **To predict weakly not taken** | 66% hit | 100% hit | 0% hit | 60% hit | 14% hit |
| **2 bit predictor**  **to predict strongly not taken** | 33% hit | 100% hit | 50% hit | 20% hit | 0.28% hit |

