CMSC611: Advanced Computer Architecture
Extra Credit Homework 3

All parts of this assignment will use the following data, taken from a (real) run of valgrind on an application. The numbers given are inclusive (including all calls of the function, and any function that it calls). Assume two levels of cache, an L1 hit time of 1 cycle, L2 hit time of 40 cycles, memory access time of 200 cycles, and a branch penalty of 10 cycles. Changes mentioned in any question apply for that question only, and should not be carried forward to later questions.

main():

Event	Short Name	Count
Calls	C_m	1
Instruction Fetch	Ir_m	7,371,916,291
Data Read Access	Dr_m	2,132,375,654
Data Write Access	Dw_m	692,582,065
L1 Instruction Fetch Misses	$I1mr_m$	2,843
L1 Data Read Misses	$D1mr_m$	2,017
L1 Data Write Misses	$D1mw_m$	13,023
L1 Miss Sum	$L1m_m$	17,883
L2 Instruction Read Misses	$ILmr_m$	719
L2 Data Read Misses	$DLmr_m$	322
L2 Data Write Misses	$DLmw_m$	12,617
L2 Miss Sum	LLm_m	13,657
Conditional Branches	Bc_m	321,742,305
Mispredicted Cond. Branches	Bcm_m	3,407,404
Indirect Branches	Bi_m	98,326,304
Mispredicted Ind. Branches	Bim_m	2,541,727
Total Mispredicted Branches	Bm_m	5,949,131

Sphere::intersect()

Event	Short Name	Count
Calls	C_s	9,555,396
Instruction Fetch	Ir_s	5,744,882,374
Data Read Access	Dr_s	1,530,528,738
Data Write Access	Dw_s	477,776,980
L1 Instruction Fetch Misses	$I1mr_s$	6
L1 Data Read Misses	$D1mr_s$	1
L1 Data Write Misses	$D1mw_s$	0
L1 Miss Sum	$L1m_s$	7
L2 Instruction Read Misses	$ILmr_s$	6
L2 Data Read Misses	$DLmr_s$	0
L2 Data Write Misses	$DLmw_s$	0
L2 Miss Sum	LLm_s	6
Conditional Branches	Bc_s	97,279,766
Mispredicted Cond. Branches	Bcm_s	925,966
Indirect Branches	Bi_s	0
Mispredicted Ind. Branches	Bim_s	0
Total Mispredicted Branches	Bm_s	925,966

a) What is the overall branch misprediction rate?

b) What is the misprediction rate within the Sphere::intersect() function?

c) What is the overall L1 miss rate? The overall L2 miss rate?

d) What is the average memory access time (AMAT) in cycles?

e) What is the expected total number of cycles for this program?

f) What is the expected total number of cycles spent in the Sphere::intersect() function?

g) If the clock rate is 2 GHz, what is the total execution time of the program?

h) If we could reduce the total execution time by a billion cycles, what would the overall speedup be?

i) If we could make Sphere::intersect() 1.5x faster what would the overall speedup be?

j) What would the expected overall speedup be if we could cut the L2 hit time to 10 cycles?

k) What would the expected overall speedup be if we could cut the memory access time to 75 cycles?

l) What would the expected overall speedup be if we could cut the branch miss rate in half?

m) The indirect branches are primarily due to virtual function calls. What would the expected overall speedup be if we could refactor the code to eliminate them without increasing the other cycle counts?

n) What would the total execution time be without cache or branch prediction?

Suppose we introduce a new conditional load instruction that can replace a branch followed by a load with a single new instruction. The new instruction does not branch, so cannot be mispredicted, however, it will access its data memory argument even if it doesn't use it, so may increase the data access rate.

o) If **all** of the conditional branches in Sphere::intersect() could use this new instruction, what speedup would you expect in the Sphere::intersect() function? Be sure to account for any changes in instruction count, data access count, and branch count.

p) If you only use this new instruction in Sphere::intersect(), what is your expected total speedup?

q) What is the expected total execution time with the new instruction?