Project 2 - Profiling MiBench with gem5

Spring 2016, CSI 3102

Submission Instructions

I. If you don't obey ***Submission Instructions***, you will get 0 point.

II. Submit only one file per team.

III. Submitted file's name must be: *project2_YourId_TeammateId.tar*
(ex) project2_2014123456_2014987654.tar

IV. In the submitted tar file, every file name have to be in our ***File List***.

V. <CAUTION> Every file and directory name must be lower case.

Problem 1-(a)	5 points each x 6 = 30 points
Problem 1-(b)	20 points
Problem 2-(a)	5 points each x 4 = 20 points
Problem 2-(b)	30 points
TOTAL	**100 points**

File List

(Root Directory) project2_YourId_TeammateId

- (File) peer_review.txt
- (Directory) problem1
  - (File) basicmath_arm_atomic_stats.txt
  - (File) basicmath_arm_timing_stats.txt
  - (File) dijkstra_large_arm_atomic_stats.txt
  - (File) dijkstra_large_arm_timing_stats.txt
  - (File) fft_arm_atomic_stats.txt
  - (File) fft_arm_timing_stats.txt
  - (File) problem1-b.txt
- (Directory) problem2
  - (File) sha_o0_arm_stats.txt
  - (File) sha_o3_arm_stats.txt
  - (File) sha_unrolled_o0_arm_stats.txt
  - (File) sha_unrolled_o3_arm_stats.txt
  - (File) problem2-b.txt

Problem 1

Get the following benchmarks among *MiBench* benchmarks, and perform simulation with *gem5* for those benchmarks.

Configuration	config/example/se.py	
ISA	ARM	
CPU Type	[atomic / timing]	
Benchmarks	*Binary*	*Input Set*
	basicmath_large	No input needed
	dijkstra_large	input.dat
	fft	See **NOTE**

(a) Submit files which contain simulation statistics of each benchmark.

(b) Based on simulation statistics, explain major differences observed in the atomic and timing model.

NOTE

1. Put $\sum(last\ 3\ digits\ of\ ID)$, 64 as input to the binary *fft*. For example, if your team members have student ID 2015987001 and 2015321301, then you have to put 302, 64 as input to the binary *fft*.
2. Build all binaries with compiler optimization level 3. (-O3)
3. You need to explain major differences, not all differences.

Problem 2

Get the benchmark *sha* among *MiBench* benchmarks. Then, build the binaries described in below. You can easily unroll loops: Just add "#define UNROLL_LOOPS 1" at the first line of 'sha.c', then build. Perform simulation with *gem5* for benchmarks below.

Binary	*Unrolled*	*Compiler Optimization*	*ISA*
sha_o0	No	-O0	ARM
sha_o3	No	-O3	ARM
sha_unrolled_o0	Yes	-O0	ARM
sha_unrolled_o3	Yes	-O3	ARM

Configuration	config/example/se.py
ISA	ARM
CPU Type	atomic
Input Set	input_large.asc
Binary	sha_o0
	sha_o3
	sha_unrolled_o0
	sha_unrolled_o3

(a) Submit files which contain simulation statistics of each benchmark.

(b) Based on simulation statistics, explain effects of unrolling technique and compiler optimization on data hazards and control hazards.

Hint

Important Metircs in Simulation Statistics

sim_ticks	Number of ticks simulated
sim_insts	Number of instructions simulated
system.cpu.numCycles	Number of cpu cycles simulated
system.cpu.num_int_alu_accesses	Number of integer alu accesses
system.cpu.num_func_calls	Number of times a function call or return occurred
system.cpu.num_conditional_control_insts	Number of instructions that are conditional controls
system.cpu.num_int_insts	Number of integer instructions
system.cpu.num_int_register_reads	Number of times the integer registers were read
system.cpu.num_int_register_writes	Number of times the integer registers were written
system.cpu.num_mem_refs	Number of memory refs
system.cpu.num_load_insts	Number of load instructions
system.cpu.num_store_insts	Number of store instructions
system.cpu.num_idle_cycles	Number of idle cycles
system.cpu.num_busy_cycles	Number of busy cycles