ECE 462/562
Homework 1 Deliverables

Due: Wednesday, February 12, 11.59pm.

Group members (and netID) who participated:

Name	**NetID**
Mitchell Dzurick	**mitchdz@email.arizona.edu**
Lena Voytek	**dvoytek@email.arizona.edu**
Amir	**asdaghpour@email.arizona.edu**

PART 1

Fill in the following tables with the required information and answer the following questions. The *notes* below contain the stats that should be used for filling out the tables.

1. Overall profiling stats (*30 pts*)

Benchmark	Instruction count	#cycles simulated	IPC	% load	% store	% branches	% int	% fp
a2time01	10000003	5562305	1.7978	12.2534	7.2712	12.2534	98.7770	0
cacheb01	10000001	5327135	1.8771	25.3292	11.7464	9.5808	97.7695	0
bitmnp01	10000002	8325401	1.2011	20.7439	18.1393	5.9520	95.4615	0
mcf	10000001	7240997	1.3810	22.6185	17.6623	14.5372	89.3985	0
libquantum	10000000	12010380	0.8326	22.9182	16.6714	16.6721	100.000	0

Answer the following questions based on your simulations.

2. Using the number of ALU instructions (*system.cpu.commit.int_insts* + *system.cpu.commit.fp_insts*), determine the % compute intensity of each benchmark and list the benchmarks in order of compute intensity. (*5 pts*)

Benchmark	**% ALU instructions**
mcf	8939850
bitmnp01	9546151
cacheb01	9776948
a2time01	9877700
libquantum	10000016
GEOMETRIC MEAN	9620633.551

*Note that the geometric mean is *not* the arithmetic mean!

3. Using the number of executed memory references (*system.cpu.iew.exec_refs*), determine the % memory reference of each benchmark and list the benchmarks in ascending order of memory intensity (*5 pts*).

Benchmark	**% memory references**
a2time01	1952466
cacheb01	3707558
bitmnp01	3888313
libquantum	3958959
mcf	4028078
GEOMETRIC MEAN	3391738.064

4. Cache performance (*30 pts*)

Benchmark	iCache miss rate (%)	dCache miss rate (%)	iCache AMAT (cycles)	dCache AMAT (cycles)
a2time01	0.000196	0.000537	1.01568	1.04296
cacheb01	0.000185	0.004289	1.0148	1.34312
bitmnp01	0.000208	0.000433	1.01664	1.03464
mcf	0.000177	0.021864	1.01416	2.74912
libquantum	0.000127	0.249866	1.01016	20.98928
GEOMETRIC MEAN	0.000176 1382286	0.005587 930582	1.01428 5553	2.42366501 7

*To calculate the average memory access time (AMAT), assume an L1 iCache and dCache latency of 1 cycle and main memory access latency of 80 cycles.

Notes:

- Instruction count is *sim_insts*
- % load and % store are the percentage of load and store instructions executed, respectively. Calculate this using the total memory references executed (*system.cpu.iew.exec_refs*), stores executed (*system.cpu.iew.exec_stores*), and loads executed (*system.cpu.iew.iewExecLoadInsts*). These are events in the Issue/Execute/Writeback (iew) stages of the 7-stage O3CPU pipeline
- Total branches: *system.cpu.iew.exec_branches*.
- % int (*system.cpu.commit.int_insts*) and %fp (*system.cpu.commit.fp_insts*) are the percentage of integer and floating point instructions, respectively, with respect to the total number of instructions
- iCache and dCache are the instruction and data caches, respectively.
    - iCache miss rate: *system.cpu.icache.overall_miss_rate::total*
    - dCache miss rate: *system.cpu.dcache.overall_miss_rate::total*

PART 2

Fill in the following tables with the required information and answer the following questions. The *notes* below contain the stats that should be used for filling out the tables.

5. L2 cache performance impact *(30 pts)*

Benchmark	L2 cache miss rates	Simulation seconds without L2	Simulation seconds with L2	% time improvement with L2	IPC without L2	IPC with L2	% IPC improvement with L2
a2time01	0.894737	0.003476	0.003478	-0.05753739931	1.797816	1.797219	0.03320695778
cacheb01	0.06185	0.003329	0.003217	3.364373686	1.877182	1.942835	-3.497423265
bitmnp01	0.9159	0.005203	0.005207	-0.07687872381	1.201144	1.200307	0.06968356833
mcf	0.304824	0.004526	0.004139	8.550596553	1.381025	1.509983	-9.337846889
libquantum	0.073175	0.007506	0.00422	43.77831068	0.832613	1.481134	-77.88984798
GEOMETRIC MEAN	0.2574299129	0.004593609779	0.003994985605	1.409883432	1.360499183	1.564512847	1.425499451

*Simulation seconds is *sim_seconds;* L2 cache miss rates is *system.l2.overall_miss_rate::total*

Think about and discuss the results and your observations with your group.

Some benchmarks' L2 cache miss rates are much higher than the L1 cache miss rates. Why do you think that is the case?

Data that is being used a lot will be in the L1 cache, and therefore be hit a lot for benchmarks that reference the same data.

Some benchmarks for which the L1 cache seems to suffice (i.e., very low miss rates) have very high L2 cache miss rates. Why do you think that is the case?

Some benchmarks use the same data a lot, therefore the data will be in the L1 cache instead of L2.