

**CS 4504 – Computer Organization**  
**Prof. Kirk W. Cameron**  
**Project 1: Microbenchmarking using LMBench**

**Important Dates**

Assigned: Jan 24, 2006

Official due date: Feb. 7, 2006

Extended due date: Feb. 9, 2006 (-10 points for late submission)

**Summary**

In this project you are expected to compile and run codes from the LMBench benchmark suite to identify the memory characteristics of your architecture. You may use any architecture where you have the ability to compile C code. The TA can answer technical questions related to Unix systems, but running benchmarks on any other operating systems is at your discretion. You will deliver a short report (< 3 pages) illustrating your results and listing the characteristics of your system's configuration.

**Details**

Read the paper "Optimizing Application Performance: A Case Study Using LMBench" available at the course home page. Section 3 describes how to download and use the LMBench code to determine the sizes and latencies of a given architecture.

You are to perform the experiments described in the paper to allow you to empirically determine the cache sizes and average memory latencies of your target machine. To get accurate results, you should minimize other applications running when performing measurements and at times it may be helpful to reboot your machine prior to experimentation (if you are using your own PC). Measurements should consist of multiple runs to rule out statistical variations. You may use any platform you have access to, although you should expect to use the same platform on later projects.

Your results should primarily consist of 1 figure and 1 table. You should also spend about 1 page discussing your methodology, any problems, and interesting results. The figure should be similar to Figure 10 found in the paper illustrating the average latencies and sizes of the caches in the memory hierarchy. You should reference machine specifications for data unable to be found empirically (e.g. associativity) – though you must note this in your report. The table provides a summary of the architecture studied with entries (filled out for your target system) as in the following example:

Architecture Type	Single CPU/Symmetric Multiprocessor/Distributed Shared Memory/etc.
Processor	MIPS R10000/Intel PIII/etc.
Clock Speed	200 MHz
Processor Cycle time	5.0 ns
Word size	32-bit
Translation look-aside buffer	64-entries
TLB miss penalty (page in memory)	~2000 cycles (10 $\mu$ s) <sup>†</sup>
TLB miss penalty (page fault)	10 to 100x10 <sup>6</sup> cycles (50 to 500 ms)
# General Purpose Registers (int+fp)	32
L1 instr cache size (block size)	On-chip, 32 Kbytes (64 bytes) <sup>†</sup> 2-way set associative, LRU array of 512 64 byte cache lines
L1 data cache size (block size)	On-chip, 32 Kbytes (32 bytes) <sup>†</sup> 2-way set associative, LRU array of 1024 32 byte cache lines
Average L1 data access latency	2 cycles (10 ns) <sup>†</sup>
L2 instr cache size (block size)	Off-chip, Unified, 4 Mbytes (128 bytes) <sup>†</sup> multiplex 2-way set associative, LRU
L2 data cache size (block size)	
Average L2 data access latency	10 cycles (50 ns) <sup>†</sup>
L3 instr cache size (block size)	No L3 cache
L3 data cache size (block size)	
Main memory size (page size)	1GB (16 Kbytes) <sup>†</sup>
Average memory access latency	80 cycles (400 ns) <sup>†</sup>
System bus	400 MB/s <sup>†</sup>
Operating System	SGI IRIX 6.5.14
Compiler	MIPSpro Compiler Version 7.3.1.3m

<sup>†</sup>Entries determined through empirical measurement using LMBench.