McGill University
Department of Electrical and Computer Engineering
Computer Organization and Architecture
ECSE 425 – Winter 2007
Project

Due date: April 11, 2007, 5:00 PM

Deliverables

1. Submit a single .zip file on WebCT including:
   - A single PDF file of your report
   - your code and data (plain text only)

   - Other formats / multiple files will not be marked.

2. Submit a hardcopy version of your report (including your code) in the assignment box.

3. Your submission is not complete without both the hardcopy and electronic versions.

4. In your report, describe how to run your code. We should be able to reproduce your results using the instructions you give.

You can choose from one of the two following project topics:

A. Comparative study: The goal of this project is to analyze existing computers in order to understand key aspects of their architecture by means of small benchmarks that students design to reveal their respective strengths and weaknesses.

B. Architectural simulation: This project consists of writing a software behavioral simulator for an architectural component such as a cache or a branch predictor. The simulator will be used to derive performance statistics as a function of design parameters. For example, in the case of a cache, these parameters would consist of cache size, block size and cache organization. Similarly, for branch predictor static and dynamic predictors would be compared as a function of their complexity. It could also be a Tomasulo data path looking at the size of the reservation stations, or a reorder buffer (ROB).

GENERAL GUIDELINES

- You may make use of reasonable assumptions of your own for those data that might be missing in the following problem text, provided that they are explicitly and clearly stated, and not contrasting with the text itself.

- You may submit a partial project. The grading will be scaled balancing problem complexity and fraction completed.

- You may work on this project alone or in teams of **at most two students**. **Teams of three will NOT be accepted under any circumstances.** If you don't have a partner, see the course instructor. Team partners will be jointly responsible for the whole work, and will receive the same grade. Teams partners will submit only one copy of their project report.

- A project should consist of 8 to 14 pages **maximum** of text, plus an appropriate amount of graphical or tabular data. In addition to this text, include an appendix which gives instructions on how to run your code and reproduce your results. The appropriateness and the presentation of the supporting data will influence the mark greatly.

- Remember, your job is to present your work very clearly to those who read the report. It is hard to appreciate a poorly presented design, no matter how clever it is.

Project A: Comparative Study

GOAL

The goal of this project is to analyze existing computers and to understand their architecture by pushing them to their limits (it is often said that in science, understanding of a phenomenon is gained by finding the conditions under which known laws no longer apply, for a system, the game is to find operating conditions under which performance breaks down).

This project will also help debunk the fallacious concept of "peak performance" (quote from the textbook: "performance guaranteed never to be exceeded"!).

A primary requirement is to secure access to two computers of comparable performance but of different architectures. For example a computer using a SPARC CPU (like in the Solaris stations). It could also be a Pentium, or a PowerPC/Gx, or any other modern design, as long as you can secure access to it for experimentation. Making this choice, you must make sure that you can find detailed technical description about their architecture. This type of information is hard to get from manufacturers for obvious reasons. However, there are books written about almost every architecture, so a part of your research will be find what is available at the Library (at McGill or at other Universities in Montreal).

The problem is to design at least two small benchmarks that will cause one computer to perform at nearly full speed and the other badly *and vice-versa.*.

PRESENTATION OF THE REPORT

PART-I Analysis.

Perform a comparative study of the two machines you have selected, in terms of the architectural concepts we study in the course, in quantitative terms as much as possible (data path width, number of regs, etc...):

1. Pipeline(s) structure. Functional units, depth, etc...

2. Instruction scheduling and issue.
3. Data hazard detection and resolution.
4. Control hazard handling.
5. Memory hierarchy structure: cache type, levels, replacement algorithms, etc...
6. Virtual memory system, if supported.

Discuss the opportunities the architectural differences offer in terms of the goals of the project. For example, if one architecture supports speculation and the other not, etc...

PART-II Benchmark and Experiment Design.

Design of the two benchmarks designed to tease out the differences between the two machines. Discuss which particular architectural features (or lack thereof) from your estimates, will give you the largest spread in performance. Places to look for are any of the items in the list above (and perhaps others, like IO performance). The benchmarks could be integer (sorting and searching, string processing, etc...) or scientific calculations (matrices, sparse matrices, or smaller computational cores like inner products).

It is **very important** to design parametric benchmarks (that have tunable parameters like the size of vectors) which are meaningful in the sense that they perform useful functions.

PART-III Reporting and Discussion

In this part, the results must be concisely and **fairly** reported. This means that as much as possible, factors like clock speed or technology generation should be taken out using proper averaging and normalization methods.

Discuss the relative merits and disadvantages of the two machines in terms of prospective applications and conclude.

Project B. Architectural Simulation

GOAL

You are to write a software simulator for an architectural component of a CPU. Good candidates are cache subsystems or branch predictors. The software simulator is to be used to derive statistics relative to the performance as a function of design parameters. For example, in the case of a cache, these parameters would consist of cache size, block size and cache organization in terms of associativity. Similarly, for branch predictor static and dynamic predictors would be compared as a function of their complexity and storage requirements.

There are several approaches to generate test benches used to produce the statistics. One of them involves the use of SPIM (http://www.cs.wisc.edu/~larus/spim.html), a simulator for the R2000 CPU. One other consists of instrumenting actual code to collect execution traces. Yet another possibility is to download ready-made execution traces published on the web. Some traces will be provided to you on WebCT if you wish to use them.

PRESENTATION OF THE REPORT

PART-I Simulator Design.

The simulator takes as input a stream of addresses, and as output produces statistics as to the performance of the subsystem under study. In all cases, the output should be stored in a file in human readable form.

In the case of caches you may concentrate on the performance of a data cache. The most important statistics should be the hit rate for a particular sequence of effective addresses. In the case of a branch predictor, the input is the *trace* of a program, that is, the sequence of all addresses visited during the execution of a program. The addresses that correspond to a branch conditional are marked. This is all the information that's needed since the predictor's only source of information is the past execution of a program. The output is the correct prediction rate for a particular program.

The simulator should be coded in C or C++. As any program, its design consists of data structures and algorithms. The data structure must be designed to reflect the storage components of the subsystem, while the algorithm should reflect the logic that controls them. In any case, it is important to design a program which is highly structured so it is easy to debug, even if it is not efficient.

PART-II Test bench design

To make the statistics meaningful, the addresses should be derived from a real program, even if it is a simple one. Therefore, the collection of addresses should be automated.

PART-III Reporting and Discussion

One of the advantages of having a software simulator is that parameters can be changed easily and the results known immediately. Discuss and explain your results.

References Used by Previous Student Teams

This list is by no means comprehensive, or up-to-date.

Tabak, Daniel. 1996. *RISC Systems and Applications. Research Studies Press. Ltd.*

Tabak, Daniel. 1995. *Advanced Microprocessors (2nd edition). McGraw Hill.* Staken, P. H., 1996. *A practitioner's guide to RSC Microprocessor Architecture. Johm Wiley and Sons.*

Jacob, B., Mudge, T. 1998. *Virtual Memory in Contemporary Microprocessors.*

Anderson, D. Shanley, T. 1995. *T. Pentium processor system architecture,* Mindshare Inc.

Cockcrof, A., 1995. *Sun performance and tuning: SPARC and Solaris.* Prentice Hall.

Heuring, V. P. 1997. *Computer Systems and Architecture.* Addison Wesley.

Milutinovic, V., 1997. Surviving the design of a 200MHz RISC processor. *IEEE Computer Society Press.*

Kain, R. Y. 1996. *Advanced computer architecture, a system design approach.* Prentice Hall.

Handy, J. 1998. *The cache memory handbook.* Academic Press.

Paul, R. P. 1994. *SPARC architecture, assembly language programming and C,* Prentice Hall.

Intel Corp. (Mt. Prospect IL) 1997. *Pentium processor family developer's manual*

Intel Corp. (Mt. Prospect IL) 1997. *Pentium pro processor workstation performance*

Intel Corp. (Mt. Prospect IL) 1997. *Overview of processor architectures and pipelines*

Motorola 1994. *Power PC 604 RISC microprocessor technical summary.*

Shanley, T. 1995. *Power PC system architecture.* Mindshare Inc.

Shanley, T. 1997. *Pentium pro processor architecture* Addison-Wesley Developers Press.

Great resource page. http://ulita.ms.mff.cuni.cz/pub/techdoc/

PowerPC 604e. http://ebus.motorola.com

K7 Challenges Intel Microprocessor Report. Vol. 12, No. 14, 1998.

CPU Guide - The Athlon Processor. *Tom's hardware* http://www.tomshardware.com

AMD Athlon Technical Brief. *AMD Technical Documents*

http://www.amd.com/products/cpg/athlon/techdocs/index.html

http://www.amd.com/products/duron/index.html

Sun Microsystems. 1997. *UltraSPARC-IIi processor technology whitepaper.* http://www.sun.com/microelectronics/whitepapers/UltraSPARC-IIi/

UltraSparc-IIi User Manual. *http://www.sun.com/developer*

Sun Microsystems. 1997. *UltraSPARC-IIi user's manual.*

Sun Microsystems 1998. *UltraSPARC 5 and UltraSPARC 10 Workstation architecture.*

Yang. R. 1998. *Evaluation of a commercial microprocessor (SMLI TR 98-65).* University of California, Berkeley, CA.

Dowd, K. 1993. *High performance computing: RISC architectures, optimization and benchmarks.* O'Reilly and Associates, Inc.: Sebastopol CA.

http://developer.intel.com/vtune/cbts/pproopt/ppopdown.htm

http://developer.intel.com/design/pentiumii/manuals/243190.htm

ftp://download.intel.com/design/PentiumII/manual

www.sandpile.org

Case study. http://www.cs.umu.se/~tdv94jbd/dark/Lab3.html

Tower of Hanoi. http://cs.bsu.edu/homepages/peb/sc120/examples/hanoi.html

Branch Prediction in the Pentium family

http://x86.ddj.com/articles/branch/branchprediction.html

SPEC Benchmark test results: Standrd Performance Evaluation Corporation.

http://www.spec.org

Ready made Trace files http://cs.waikato.ac.nz/~jcleary/311/311/traces

Ready made Trace files

http://www.cs.utexas.edu/users/vin/CS352/Project/Cache.Project.html

http://www.systemlogic.net/articles/00/01/cache

http://www.linuxdic.org

http://www.linuxdoc.org/HOWTO/Benchmarking-HOWTO.html

http://www.cs.wisc.edu/~thomas/comp.benchmark.FAQ.html

http://www.gnu.org

http://www4.nscu.edu/~liang/processors.html

http://cag-www.lcs.mit.edu/~wklee/sparc.html

http://www.cs.wisc.edu/~arch/www/search.html