CS 350c, Spring 2017, Laboratory 0
Memory Mountain

Assigned: Thursday, January 19, 2017
Due: Thursday, February 2, 2016, by noon

1 Introduction

In this lab, you will learn about the memory throughput of an x86-compatible microprocessor. The access time to the memory varies with its use. We have provided a program that will help you visualize the differences in the memory performance; this program repeatedly accesses different amounts of memory so as to estimate the speed at which memory can be read. This program varies the amount of memory accessed and the access pattern.

The original version of this program was provided by Bryant and O'Hallaron for helping students understand material from their book "Computer Systems, A Programmer's Perspective." We have simplified the original version and inserted questions in the code that you are expected to answer as a part of this laboratory assignment.

The memory-performance program you will use is provided as a part of this laboratory assignment. Below, we describe how to compile and use the resulting binary to observe the effects of repeatedly reading various amounts of memory. For each memory size, this program also varies the *stride*; that is, it first reads memory locations sequentially, then it reads every second location, and then every third location, and so on, up to every 16th location.

When you have completed the lab, you will hopefully have a better appreciation for bandwidth between the main memory, the caches, and the microprocessor. Memory performance is the dominant factor in the performance of programs. It is important that you internalize how modern microprocessor memory systems effect the execution of your programs.

2 Logistics

You are expected to work on this lab alone. However, you may communicate with others concerning your understanding of C code and the various tools, e.g., the compiler, some spreadsheet program for creating a 3-D graph, and other systems issues. But, the answers that you submit in response to the questions posed (more later) must be answered by you alone.

Any clarifications and revisions to the laboratory will be posted on the course Web page.

There are many sources of information about x86 processors and the associated systems that use them. We will make use of material that can be downloaded from the Web. In the case of Agner Fog's information about the x86 family of processors, we have provided a local copy on the class website. In addition, we have provided a local copy of Intel's x86 documentation.

Your submission should include this information and an explanation of why the graph of the *memory mountain* looks as it does.

3 Handout Instructions

You will find the file `memory-mountain.tar` referenced on the homework page of the class Website. You will need to download this file so you can use its contents. There may be changes or updates, so please be sure to download the latest version – check the top-level class Webpage for any updates or corrections.

4 Evaluation

You are expected to write a report that includes a 3D graph of the results of running the memory mountain code. Running the program is straightforward. Embedded in comments of the actual C-language code for the memory-mountain program are questions that you are expected to answer. Please devote a paragraph or two to each answer.

The maximum score for this laboratory is 100 points. The value of the individual components is as follows.

- Raw data from the run of the memory-mountain program (10 points).
- A 3-D picture showing the memory performance (10 points).
- An explanation of the results that explains the raw data and your associated graph (30 points).
- Answers to the five questions that are embedded as comments in the C-language code that you compiled and then executed (10 points for each answer).

5 Running the Memory Mountain Code

Your task is to compile and run the memory mountain code. For the code provided, we recommend that you use a UTCS Department Linux system running on a computer with a X86 architecture; any of the CS Department machines are suitable. The memory-mountain source files are contained in the `memory-mountain.tar` file.

The program is contained in the files `clock.c`, `fcyc.c`, and `mountain.c`, along with the two associated header files `clock.h`, `fcyc.h`.

Below is a command sufficient to compile this program so you may run it; you only need to type:

```
gcc -Wall -O2 -o mountain *.c
```

The resulting compiled program can be run by just typing:

```
./mountain
```

The output of this program will be a matrix of values that presents the memory bandwidth of the computer for different sized working sets and different stride sizes.

To complete your laboratory, you need to graph your results. In addition, you need to write down the model and speed of the microprocessor that you use to get the bandwidth results. On a Linux system, you can find some of the information by typing the commands:

```
cat /proc/cpuinfo
cat /proc/meminfo
```

On an Apple System, use the "About This Mac" menu option under the Apple (upper-left-hand corner of the screen). On FreeBSD and TrueOS one can look at the boot-time messages with command **dmesg**.

With the information you gather, you can then lookup on Intel's Website the configuration of the internal caches. You may find it somewhat difficult to exactly identify the processor in the system you are using, but on this point, please feel free to ask anyone for help.

We have left five questions in the source code of the laboratory. You need to write explanations for the questions, and submit your answers as a part of your report. One or two paragraph answers are expected for each question posed – please don't write just to fill space, keep your answers short and clear.

Hand In Instructions

Please follow the instructions below for turning in your work.

- Make sure you have included your identifying information in your files.
- To handin your memory-mountain laboratory, prepare a report that you can submit to the CANVAS system as a PDF file. Name the file with your report `<YourUTID>.pdf`. The Canvas system identifies you in this manner.