ECE150, Computer Architecture – Assignment 1

Submit by Mar. 9, 8PM

tldr: Uncover the underlying reason for the performance discrepancy between two functionally equivalent C routines.

Problem Statement Consider an uninitialized two-dimensional array in C. We may traverse this array and initialize each value to some integer along the way. Does the order we initialize the elements the matter? For those of you with any DSA experience you will know that accessing an element of an array takes constant time in both the worse case and average case. This may lead you to believe that the order of array access does not matter — but when executing on a computer it does! (This is another example of why I suggest you divorce algorithms from their implementations.) I ask you to consider why. While we have hinted at the underlying reason in class, we have not explicitly discussed it.

For this assignment do not use a MIPS simulator. I ask you to compile the provided C programs with GCC for your local (presumably x86) processor. The observed phenomenon will be the same. That being said, I ask you to present your argument using your knowledge of MIPS and to present any instructions in the MIPS assembly language.

Please attach any code you write/modify and cite all third party sources you use to supplement your own work.

nb: A "correct" hypothesis is not sufficent; I am interested in your argument.

Some *suggestions* to guide your analysis:

- Write a proper benchmark for the programs (need not be in C)
  - Think about your performance metric. I will not grade you based on your choice, but rather your argument for it.
  - Be careful with *what* you are actually measuring.
  - Report the 5th, 50th, and 95th percentiles.
- *Sketch* out what you except the MIPS assembly might look like.
- Vary the size of the two-dimensional array from very small to very large. It may be helpful to plot the resulting benchmarks versus array size.
- Consider non-square arrays (you will need to modify the C to do this).
- Does repeating the routine in C change anything?

Some resources:

C Program `http://ee.cooper.edu/~curro/comparch/hw1/main.c`

Makefile Sketch `http://ee.cooper.edu/~curro/comparch/hw1/Makefile`

This Document `http://ee.cooper.edu/~curro/comparch/hw1/assign.pdf`