# CSE P548: Computer Architecture
# Assignment 3
# Due: Wednesday, May 11

This assignment is meant to test your understanding of cache designs and the interaction between certain cache configuration parameter values and the effect they have on cache performance.

For this assignment work in teams if you'd like. Try to work with someone you haven't worked with before.

## Part I: Cache simulation

Write application programs that enable you to discover the cache size, associativity, line (block) size and memory update policy of the **L1 data cache** being simulated in a special version of the SimpleScalar tool set. Design the programs so that each one determines a particular configuration parameter. The order in which you run your applications might be important, so think about this as you are designing the program algorithms.

The cache size, associativity and line size of the L1 data cache will all be powers of 2. The cache will be no larger than 64KB, the associativity will be no greater than 4-way and the line size will be a minimum of 4B and no larger than 64B.

You will use a modified version of *sim-cache*, which is a simpler and faster version of *sim-outorder* that only gives cache statistics. The modified version of *sim-cache* is called *secret*, and is located in the bin/ subdirectory of the course directory. *secret* has been altered so that the cache configuration parameters of interest have been set to particular values, but are not printed as part of the output. Use it as follows

*secret myfile arg0 arg1 ...*

Since you'll only have access to the executable version of *secret* (if we give you the source, you obviously will be able to see the cache configuration), you'll need to run it on the department computers. Your application development, testing, data analysis and report writing can all be done on your own machines, as usual.

To compile your application programs for the SimpleScalar tools, use the *ss-gcc* compiler that is located in the simplescalar/bin/ subdirectory of the course directory. *ss-gcc* works just like the usual *gcc*, except that it cross-compiles to the SimpleScalar architecture. You should use a distinct suffix such as *.ss* on your cross-compiled binaries to avoid confusion with Linux executable binaries.

If you are developing the applications on your own machines, you can download the source code for the ss-gcc compiler and binary tools from the homework web page. There are also instructions for building ss-gcc. Sim-cache is part of the SimpleScalar tools and is built when sim-outorder is built.

*Secret* has been protected from your prying eyes. There are undoubtedly techniques that would enable you to discover the cache configuration we have used, but it is probably not worth your time trying. Whether you do or not, the graphs and/or tables you construct from your simulation output will still have to show data that "proves" that you have discovered the correct configuration, and your application programs must be able to run with the protected configuration file. To test your programs, we may run them ourselves, to confirm your output.

The standard version of *sim-cache* has also been placed in the bin/ subdirectory, and takes parameters in the same way as *sim-outorder*. You may use this to verify the configuration parameter values you have chosen. However, it is not necessary to use this software; it is your applications that will discover the configurations and your results that will validate your choice.

You can also read about *sim-cache* in section 4.2 of the SimpleScalar Tool Set documentation, located at: simplescalar/doc/users_guide_v2.pdf in the course directory.

.

## Part II: The report

Your report should follow the guidelines for the last report. Only 2-3 pages of text should be enough to explain whatever you need to explain (your figures and tables can be extra). You won't need all sections of a normal research paper this time. Instead just include the following sections: (1) an introduction that simply provides in a nutshell the problem you are trying to solve, (2) a section that describes the algorithms in your programs, and (3) a results section that presents your graphs/tables, explains them and analyzes the data. No summary will be needed. And don't forget to clearly state what you think the configuration is!

Learning to write concisely is one of the goals of an assignment like this. We are looking for a concise description of the techniques you used to discover the cache configuration parameter values and your analysis of the data that led to this conclusion. If your report is much longer than the recommended length, we may not read the whole thing.

## Part III: Turning it in

Turn in your report via e-mail. Douglas will send out directions for turning in the code if he wants to see it.