# CSCI 202 Computer Organization II
# Instructor: Gedare Bloom
# Project 2 (50 points)

## *Due: April 8 at 11:59 P.M.*

This project will be completed in teams of 3–5. You and your team members should work jointly and collaboratively on ALL parts of this assignment. Team members will receive the same grade subject to participation. You do not need to work with the same team members as in previous assignments.

The following may be useful:

- COD Appendix A (online http://spimsimulator.sourceforge.net/HP_AppA.pdf).
- Essential C: http://cslibrary.stanford.edu/101/
- QtSPIM available from http://spimsimulator.sourceforge.net/.
- https://openhatch.org/missions/git
- http://progit.org/book/ - git basics and advanced features.
- http://gitready.com/ - an excellent resource from beginner to advanced.
- The circled ? at the top of the screen in Bitbucket.

You may choose your own team members. You and your team members must arrange to join the same Project3_Groups on Blackboard. Send the instructor an e-mail if you have trouble and need a manual adjustment of your group.

In addition to using pair programming as described in the syllabus, I require that you use git source version control. One member of your team should already have a **private fork** of https://bitbucket.org/gedare/simple-mips-simulator.git that is shared with me (user gedare). **If your team has changed, revoke access for any members who left, and give access to any new members.**

Note: you may share the code from the previous project with other groups, or use another group's implementation of the previous project as the starting point for this project. However, all new code written to satisfy the requirements of this assignment must be the product of your team members' contributions only.

## Part 1 (50 points)

You will extend your single-cycle implementation of a processor simulator from the previous project to simulate a processor with L1 instruction cache (L1-i$) and L1 data cache (L1-d$). This requires you to write a new fetch() that uses the L1-i$ and memory() that uses the L1-d$, and likely to modify your main(), which you should put in a new file you create called single-cycle-with-cache.c, which will be similar to single-cycle.c but with a key difference: it will stall on cache misses and therefore have a cycle count different from 1 CPI.

For correctness of your cache simulation, you need to detect cache misses and stall the processor until the cache request is satisfied. You should extend your processor statistics to also keep track of the number of cache accesses, cache hits, and cache misses for every cache. Thus, for this processor, the total number of instructions plus stalls due to cache misses should be equal to the number of cycles. You should print out the cache statistics at the end of the program, and also calculate the CPI.

You should implement your cache with an option that can be enabled/disabled within cpu.h header file, for example by providing a
#define ENABLE_L1_CACHES
to turn it on, and using
#if defined(ENABLE_L1_CACHES)
to check if the caches are enabled before using them.

Modify your Makefile to add another target that builds the cpu-cache simulator as a binary executable program named single-cycle-with-cache.

Your L1 caches should be configured as follows. Note that the address space is 32-bits.

- 2 KiB for each L1-i$ and L1-d$
- Block size of 4 words
- L1-i$ is direct-mapped
- L1-d$ is 4-way set associative with True LRU replacement policy using writeback policy for write hits and write-allocate for write misses.

### Requirements

1. Your program must compile without error using ‘make’.
2. You must submit a Makefile.
3. COMMENT YOUR CODE. If something doesn’t work, you may get partial credit if your comments show that you were on the right track.
4. You must submit a README file.
5. You may not use code downloaded from the Internet. Always watch the course website for updates on the assignments.

## Submission Instructions (read carefully)

You are required to submit the following files

1. A README document (in plain text, PDF, or MS Word format). **I suggest starting work on your assignment with this file.** The README must include

    a. Your team members' names, class, an explanation of how to run your program, a brief description of the pieces of your assignment, any notes from group discussions, and the workload distribution among the team (who did what).
    b. Test cases that you used to test your program. Include: A description of what is being tested; the input; the expected output; the actual output; and any known problems of your program, which will help you earn partial credit.

2. Your plain text source code files.
3. Binary input files for testing, identified in the README.
4. A set of **git-diff** files, one for each commit you made, that can be created using the **git-format-patch** command.

Make a **tar and gzip** file with all team members last names in the file name.
– tar -zcvf Lastname1Lastname2Lastname3Lastname4.tgz
Lastname1Lastname2Lastname3Lastname4/
Replacing LastnameX with each member's **last name**. All of your files should be in the Lastname1Lastname2Lastname3Lastname4 directory. Avoid any further subdirectories.

**Important: Make sure de-compressing your submission creates a single folder with your username in it's name and all your submission contents in it!**

Submit the compressed file under the Project3 assignment on BlackBoard. Only one submission is required from each team.