ECE 356
Lab 2: Cache memory
Due Date: Apr 15 (by midnight)

In this project you will write a cache simulator. You may work in groups of your choice, just like PA1.

Your grade for the project will be based on the code (60%), your documentation (20%), and your testing cases (20%).

You should turn in a report about your code, how it works, how it was tested, and optionally, what you learned. Your code should be included with the report and it should be well structured and documented code. You should include tests you used with your code. Please include all you are turning in for the project as a single tar or zip file that you upload to Canvas.

You will develop your cache simulator to operate as with the cache memory we discussed in class and in your book. The cache will be configurable with command line options as follows (you may include default values and make these optional parameters for your program):

- Block size (number of words. Note that we don't differentiate in this project for words and bytes. You can assume all addresses are referring to words, and no further byte-word conversion is needed).
- Number of blocks in the cache.
- Associativity (only direct mapped is needed).
- Hit time (in cycles)
- Miss time (in cycles)

Your program will be given a series of addresses (from an ASCII file that contains hex addresses with one address per line). Given this input and for the cache as parameterized at the command line, compute the hit/miss rate and the AMAT. You may choose reasonable values according to your needs.

Your cache only needs to support reads. We will test your code with the test cases you developed (in txt file format). In this programming assignment's grading phase, if there are questions, the TA may need to meet with each group to discuss your code, your testing, and other details. If you work in pairs, you should each be able to discuss any aspect of the project.

Extra credit: you need to implement an additional associativity policy first proposed here to obtain 15% extra credit:

Implement flexible associativity policies, such as two way associative, etc. Your code should support an additional parameter for this purpose.

You may choose whatever language you like. None of the code for this project is to be developed by others (no finding other resources or sharing code).