**ECE 353 Lab A Summary**

**Fall 2013**

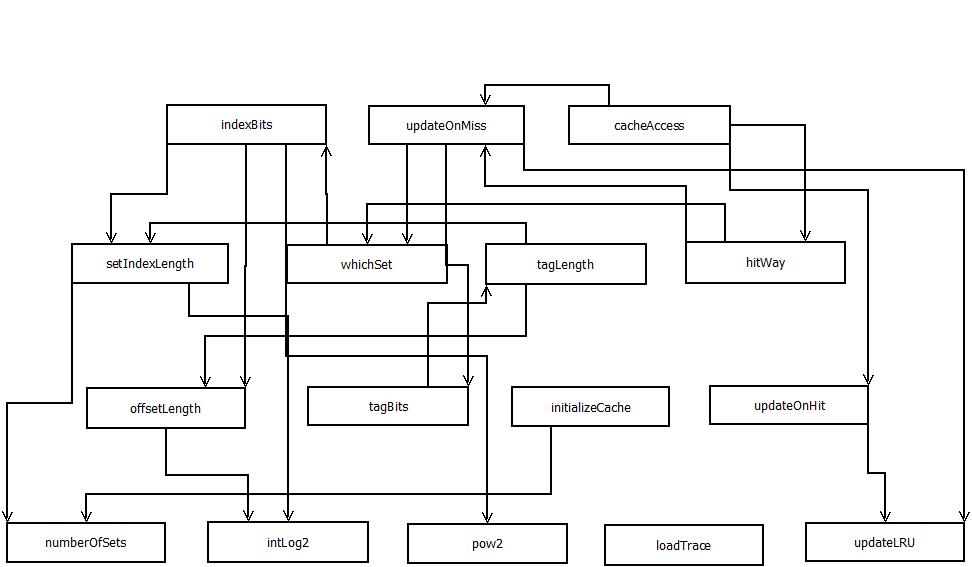
**Fill out this form and upload it to Moodle.**

**Student Names: Brandon Sprague, Zach Boynton, Jacquelyn Ingemi**

**1) Check off all the following functions which you believe work correctly in your code. These should each have passed some basic test cases. Each function should have one person principally responsible for writing the code and another who is responsible for checking its correctness and running test cases on it. Write their last names below. Authoring and testing responsibilities should be divided fairly evenly among group members. The principal author of a function should NOT be the tester for that function.**

* **whichSet**() Author: Sprague Tester: Boynton & Ingemi
* **offsetLength()**Author: Boynton Tester: Sprague
* **setIndexLength()**Author: Boynton Tester: Sprague
* **tagBits()**Author: Ingemi & Sprague Tester: Boynton
* **hitWay()**Author: Ingemi & Sprague Tester: Boynton
* **updateOnHit()**Author: Sprague Tester: Ingemi & Boynton
* **updateOnMiss()**Author: Sprague Tester: Ingemi & Boynton

**2) In the space below, draw a call graph of your code. This is a diagram which specifies which functions are called by which other functions. If A calls B, for example, there would be an arrow from node A to node B.**



**3) As a basic check of the correctness of your code, you should apply one or more test traces for which you know the miss rate by construction. Indicate briefly what kind of test trace(s) you used (coming up with a suitable, simple, test trace is considered part of the testing process). Did your program pass all your test cases?**

We created a test trace using the following Ruby code:

def rand32

str = ''

32.times do

str += ['0','1'].sample

end

str.to\_i(2)

end

File.open('trace6.txt','w') do |f|

50.times do

base = rand32

1000.times do

f.puts(base += (rand(100) - 50))

end

end

end

This code generated a trace of 50000 random 32-bit integers, with each set of 1000 deviating by no more than 50 from a base value. This should perform significantly better on caches where the block size is bigger, and the results showed this.

**4) In one of the lectures, we covered assertions as a way to identify errors in the program. Indicate which assertions you used in your code.**

Assertions were used in the main to check that our pow2() function was working properly by comparing it to the built in power function. They were also used in the main to make sure that the hit and miss counts were not erroneously negative. An assertion was also used in the pow2() function to make sure that the exponent that the function was given was non-negative because the function would give incorrect values if passed a negative value.