**Assignment 4**

**Due: 11:59 pm, November 4th, 2015**

**Purpose**: Become familiar with the implementation and operation of basic control flow divergence and reconvergence mechanisms in the GPU SIMT execution model.

**Target Machines:** Your programs must run on the PACE system using gcc 4.9. The ISA we will use is the HARP ISA.

**Assignment**: Augment your emulator by implementing control flow instructions and warp control instructions as presented in the HARP ISA and described in the Assignment 4 presentation in class. Your emulator must read the input program binaries and execute them according to the HARP ISA documentation. Emulator correctness will be determined by comparing the output against a golden output.

1. Write the emulator to conform to the HARP ISA specification. The emulator must now support control divergence using the PDOM algorithm described paper “*Dynamic Warp Formation: Efficient MIMD Control Flow on SIMD Graphics Hardware*,” by Fung. et. al. This includes support for the following HARP instructions: SIMD Control and Warp Control instructions (5.10 and 5.11 in the HARP ISA). The program binaries, inputs, and outputs are provided. Additionally, the program in assembly format is also provided for reference.
2. Implement SIMD Control instructions and test using binaries in bin-1warp/
3. Implement Warp Control instructions and test using binaries in bin-8warp/
4. The binaries assume an emulator configuration to be either 4w8/8/8/8 or 4w8/8/1/8. Consult the HARP ISA documentation for details
5. The launch arguments of your emulator should be in the form of:   
   ./harp\_emulator program output

All programs start execution at PC=0 and with a single thread. Your emulator should dynamically grow as the program spawns additional warps and threads.

1. The given programs come with inputs as part of the binary (make sure you unzip them before use). The given programs will write out to the IO device one character per instruction execution. Take the outputs from thread 0 and print them to a file whose name is defined by the output argument.
2. Grading will use a unique program binary and input/output to verify correctness.
3. You can safely ignore interrupts, trap, and privilege instructions for this assignment.

**Supporting Information.**

* Additional information is available in SIMDControlFlow.pptx
* Examples of split/join in SplitJoin.pdf

**Grading Guidelines**

For your information here are the grading guidelines

* Program compiles without errors (and appears to be correct): 25 points
* Program executes correctly: (additional) 50 points
* Documentation (comments): 25 points
  + Include a Makefile and a readme with comments, command line invocation example, and notes specific to your implementation of the assignment
  + Have a commented program header with your name, class, and assignment
  + Thoroughly comment your functions and variables

### Submission Guidelines:

Submissions should be electronic. Submissions must time stamped by midnight on the due date. Submissions will be via Tsquare.

**Note:** **No late assignments will be graded**. Remember, you are expected to make a passing grade on the assignments to pass the course!