CS246 Fall 2019 Final Project Guidelines

Expectations

- General Guidelines
  - Independent or groups of 2 for provided project ideas
  - Groups of 2 for a self-guided project ideas (please reach out to staff for any exceptions)
  - Your project should evaluate an architectural idea or technique, requiring both engineering effort and advanced understanding of computer architecture concepts to implement and analyze results.
  - Detailed information for the project ideas (see below) and additional questions will be answered during Friday's class (**Nov. 1st**).
  - If you choose your own project topic, you are encouraged to get early and frequent feedback from the staff (piazza + OH on Nov. 4th and 5th).
- Deliverables
  - Project Proposal (15%) due **November 5th**
    - Written description of which project idea your group will do including related work, methodology, and timeline of execution.
  - Progress Update (15%) due **November 22nd**
    - Written description and preliminary results including updated timeline until the final report is due.
  - Final Report (70%) due **December 9th**
    - Formal report and final results of your exploration (think of the structure of the conference papers we have read in class).
    - 10 pages maximum (including figures, not including references).

Project Topic Proposal Options

1. **TopDown Analysis**
   - For this project, you will use TopDown analysis to understand and optimize linear algebra operations
     i. Beginning with your own naive implementation of matrix multiplication, use the results of *perf* with TopDown analysis to understand computational bottlenecks (https://github.com/andikleen/pmu-tools)

ii. Make a software optimization (e.g., tiling or loop reordering) based on your observations, and observe how this changes the results of your TopDown analysis
iii. Next, compare performance characteristics and perform TopDown analysis on linear algebra operations as implemented in popular, highly optimized libraries:
    1. Eigen (http://eigen.tuxfamily.org/index.php?title=Main_Page)
    2. OpenBLAS (https://github.com/xianyi/OpenBLAS)
iv. In addition to matrix multiplication, examine how performance characteristics differ between two implementations of the same operation (e.g., matrix inversion, SVD, or others) and hypothesize why these differences occur based on TopDown results

2. **LLVM program analysis: creating a program dataflow graph to estimate maximum ILP detectable at static time. Also dynamic ILP should be used.**
    ○ Write an LLVM pass that prints the dependencies between all the operations. **http://llvm.org/docs/WritingAnLLVMPass.html**
        i. After you've tried creating a simple pass using the link above, for each instruction I you can see what are its dependencies by iterating from I.use_begin() to I.use_end().
        ii. At this point alias can be assumed between all load-store pairs.
        iii. The dependencies have to be per basic block.

    ○ Assuming that each instruction takes one cycle to execute, plot the maximum ILP per basic block from lower to higher and compute the average.
    ○ Compute the maximum dynamic ILP with Pin. Note that in pin you know all the accessed memory addresses.
    ○ Use polly or other LLVM-based alias analysis tool to diminish the gap between the static and the dynamic ILP.
3. **Value Prediction:** Pin instrumentation and evaluation of different methods for value prediction.
    ○ Implement a value predictor in Pin as described in Lipasti's paper "Exceeding the dataflow limit using Value Prediction" we've discussed in class.
    ○ The pin tool must be written in such a way that you can choose what instructions to benefit from the Value Predictor.
    ○ Create a breakdown with the predictability of data values for three spec applications in hw2 and hw3. Your have to report the percentage of predicted instructions of type.

        i. Floating point additions, substractions, multiplies and divides.
        ii. Integer multiples and divisions.
        iii. Loads.
    - Generate a separate graph for each of the 5 configurations used in Figure 9 in Lipasti's paper.
    - Generate Figure 6 as in Lipasti's paper, by aggregating your numbers for the operations you've considered and by taking the average across three benchmarks.

4. **Verilog Implementation of Branch Prediction**
    - Implement different branch prediction strategies in hardware, including those you evaluated in HW2 and one additional (equally or more advanced) strategy of your choosing
    - Port your verilog modules to an FPGA (please contact course staff, this will be the same FPGA set-up as is used for CS141)
    - Evaluate your implementations using instruction traces (i.e., list of branch instructions from a given program extracted using pin).
    - Analyze the accuracy, resource usage, and efficiency of each prediction approach for different input instruction traces.
    - Alternatively, if there is another concept from the course for which you would like to implement and evaluate a verilog implementation, please follow these guidelines to propose your idea.
5. **Choose your own adventure**
    - If your group has a research idea or project that is relevant to course material, please describe it to us in detail including
        i. Objectives (i.e., main research question you are trying to solve)
        ii. Brief survey of related work
        iii. Methodology (i.e., specific tools and a plan for how you will evaluate your project or idea)
        iv. Timeline (what you intend to accomplish by when)
    - This could be a new research idea or a re-implementation, evaluation, verification, and/or modification of an important architecture research paper that you find interesting (i.e., ISCA, MICRO, HPCA)