ECE571: Embedded Systems – Final Project
Updated 25 November 2014

Officially Due: Friday, 12 December 2014 (Last day of Classes)

Overview:

- Design a project that explores the power and/or energy performance of a modern microprocessor design. This is very open-ended, but some guidelines are below.

Guidelines:

- You may work either alone or in groups of two. If you work in a group your end project will have higher expectations.
- You may use any system you like for this project. It can be one of the systems used in class (Haswell or Trimslice), a personal system you have access to, or I can set up accounts on some of my other server or embedded boards.

Part 1: Topic Selection (due 7 November 2014)

Each group should send a brief e-mail describing your project topic and listing group members.

Part 2: Related work (counts as a Homework grade, due 4 December 2014 3:30pm)

A list of related work on your project. Do a literature search and find examples of other people who have done similar research. It is OK if you find a lot of related work, or even if someone has done the exact same thing before. This is a class project (not a PhD thesis) so reproducing something that has been done before is perfectly fine.

A quick way of finding related work is using Google, but don't limit yourself to papers turned up that way. Sometimes you might find an interesting paper via Google, but you will not be able to find a free copy online (just a site asking you to pay money). If it's a journal like the IEEE or ACM you can still get copies of the articles for free. If you're on an on-campus internet connection (ending in .maine.edu) you can go to the UMaine library homepage, search for IEEE or ACM there, and it will give you a link to click through to get to those sites (IEEE explore or similar) and you will then be able to find and download those papers for free.

I'd prefer if the references you find are books or academic papers, but if you find a few good blog or website references that's probably OK.

What I would like to receive:

- A file containing a few paragraphs about the related work. If you are working alone, I'd like to see at least 5. If you are working as a group, I'd like to see at least 10. (More is fine).
- A list of references with the work cited.
- You can submit the status update by e-mail. Only one submission is needed per group.

An short example of roughly what I expect:

Our research involves making an open-source low-powered video game that will run on an embedded Raspberry Pi board. Weaver[1] wrote a cross-platform assembly language game for the ARM platform but unlike us he did not characterize the power consumption while running. Mallow and Snap[2] look at optimizing a Ray-Tracing program on the Tegra2 ARM system. This is similar to our project, only they looked solely at ray-tracing applications and not video games.

[1] V. Weaver. "Tom Bombem: An ARM Implementation of the Classic DOS game." Proc. of the 4th Conference on Useless Video Games, p 10-18, May 1996.

[2] M. Mallow and G. Snap. "Optimizing Energy Consumption on the Tegra2." Journal of Embedded Programming, p11-19, Vol 1 Issue 15, June 2010.

Part 3: In-class Presentation, 11 December 2014

- You will have 10 minutes to present. Plan for 8 minutes describing your project and allowing 2 minutes for questions.
- Give a summary of what you did and why. Show any results you obtained. Describe any future work that needs to be done.
- You may present slides using the projector if you want, but that's not strictly necessary.

Part 4: Project Writeup, Officially Due 12 December 2014

This will be a short paper (6-8 pages) that will be in the style of an a short academic conference/journal/workshop paper and should have the following sections:

1. Introduction – describe your project and provide some background on what you are looking at
2. Related Work (what you submitted in Part 2 possibly extended a little to fit the flow of the paper)
3. Experimental Setup – list everything you did to set up your project. List benchmarks used, compiler options, hardware used, software versions, etc. It is best to provide too much than too little. Charts and diagrams are fine too.
4. Results / Analysis – describe what you found, feel free to include graphs and tables
5. Conclusion / Future Work
6. Bibliography / References Cited

Ideally any source code will also be submitted as a separate file, but I understand there might be various reasons why you cannot include this.

Also I would like to post final reports to the class web-site. If you'd rather not have your work posted in that way, let me know.

You can e-mail your final report to me. pdf or word document is fine, the code should be attached too.

Hardware Ideas:

- ARM Cortex A9 Pandaboard with performance counters
- ARM Cortex A8 Beagleboard with power measurement and performance counters
- ARM Cortex A15 Chromebook with performance counters
- ARM 1176 Raspberry Pi (perf counters complicated to access)
- Watts Up Pro logging power meter
- Haswell machine as used in homeworks
- Various recent Intel and AMD machines in both server and desktop form
- Measuring power/energy with simulators
- Various other machines (Pentium II, Pentium 4, SPARC, AVR)
- Feel free to use any other machines you have access to

Project Ideas:

- Power/Energy overhead of architectural features
  - Power/Energy overhead from prefetching (can turn off prefetching on some systems, like core2, in theory cortex a9, and many recent intel chips)
  - Power/Energy overhead from branch prediction (can turn off branch predictors on MIPS chips, not sure about ARM)
  - Power/Energy overhead from caches (find a system you can disable the cache? The ARM systems)
  - Power/Energy overhead from virtual memory
  - Power/Energy implications from frequency scaling
  - Power/Energy implications from GPUs
  - Power/Energy implications from network devices (ethernet, wireless, bluetooth, etc)
  - Power/Energy implications from vector instructions (SSE)
  - Power/Energy implications of multi-threading or multi-processing
- Hardware Performance Counters
  - Performance counter validation tests
  - Estimating power/energy using performance counters
  - Validating RAPL energy measurements on real hardware.
- Operating Systems
  - Power/Energy comparison of same task under various operating systems (Linux,OSX,Windows,FreeBSD,et

  - Make DVFS frequency scaling decisions based on hardware counter results
  - Make power/energy/performance characteristics of various Linux kernel versions

- Architectural Comparisons
  - Power/Energy vs performance on various ARM processors
  - Power/Energy vs performance on ARM vs x86

- Application Investigation
  - Pick a favorite application type and compare the power/ performance of various implementations
  - Investigate the power/performance of a benchmark when varying compiler options
  - Pick a poorly behaving benchmark (power or performance wise), find the cause of poor performance, and improve it.

- Simulation
  - Explore power/performance of an architectural feature using a simulator or DBI tool.

- Many other topics are open for investigation, feel free to suggest something.