Project Summary: Static Energy Analysis of Lower Level Program Representation

Patrick May

1 Abstract

Within software development, noticeable efforts are constantly being made towards *improving* existing code, to various ends – readability, safety, and speed are common goals of development manpower. Another axis upon which programs can be improved is that of energy efficiency. Detecting when and where a program has a weakness is a difficult problem that is done with various *profiling* and *analysis* tools.

Dynamic analysis involves interacting with a program during its runtime to glean further information about it. Unfortunately, this additional interaction creates overhead that, when profiling for energy consumption, can significantly effect power usage results. For my senior independent study, I want to investigate **static analysis** of programs for energy consumption. Static analysis is the process of understanding code *before* runtime.

Static energy profiling will involve interpreting lower level code, either **Assembly** or **LLVM Intermediate Representation** to understand the energy cost of different blocks of a program. The overall energy cost of a program can be modeled through primarily summing all energy costs of program subsections [6]. Since static analysis does not occur during runtime, another non-trivial theoretical challenge is that of solving **recurrence relations** of the underlying code. This is still an unsolved problem, although some specific recurrence relations have discrete solutions, and general form 'best effort' solvers exist as well [1].

The minimum goal of this work is to produce a tool that, given an ARM assembly program, can produce an estimate for total energy cost to execute the program, in the form of an equation dependent on underlying individual instruction energy usage. **ARM** is of interest because it is both a simpler assembly language than x86, and also allows the researcher to use a much cheaper testbench, such as a Raspberry Pi. A general *ARM & Computer Organization* textbook [5] will be used as a reference for assembly and hardware difficulties. This analysis tool will be completed no later than the end of the Fall 2023 semester. The next few weeks will be spent working on constructing an experimental dataset of instruction energy costs to allow for a simple {n, μ, m}J prediction of a program's energy cost, which then can be compared with dynamic readings obtained through simple *RAPL* means [3]. This would allow for statistical verification of static vs. actual energy usage accuracy. Other additional work considerations could involve multithreading concerns [4] or stepping backwards in the compilation chain towards LLVM representation [2].

This is a very ambitious project, with multiple different areas of exploration (and potential pitfalls). I believe it is unlikely that any issue will arise that makes the project forcibly incomplete, as other researchers have created similar static energy analysis tools before. Hardware benchmarking could become difficult depending on hardware availability and involved-ness of constructing a testing harness for a Raspberry Pi. Should this be insurmountable, a virtualized workaround may be utilized, at risk of result degredation.

References

[1] E. Albert, P. Arenas, S. Genaim, and G. Puebla. Cost relation systems: A language-independent target language for cost analysis. *Electronic Notes in Theoretical Computer Science*, 248:31–46, 2009. Proceedings of the Eighth Spanish Conference on Programming and Computer Languages (PROLE 2008).

[2] N. Grech, K. Georgiou, J. Pallister, S. Kerrison, J. Morse, and K. Eder. Static analysis of energy consumption for LLVM IR programs. In *Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems*. ACM, jun 2015.

[3] S. Kerrison. Monitoring the energy consumption of a raspberry pi with a mageec wand, 08 2016.

[4] S. Kerrison and K. Eder. Energy modeling of software for a hardware multithreaded embedded microprocessor. *ACM Trans. Embed. Comput. Syst.*, 14(3), apr 2015.

[5] R. G. Plantz. Introduction to Computer Organization: ARM Assembly Language Using Raspberry Pi. https://bob.cs.sonoma.edu/IntroCompOrg-RPi/intro-co-rpi.html. Accessed: 09-01-2023.

[6] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: First step towards software power minimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2:437 – 445, 01 1995.