# Computer Architecture
# EECE7352
# Prof. Kaeli

**Homework 1:**

**Part A:**

For the first part of this homework, compile and run the Dhrystone benchmark (***drystone.c***) on two different microarchitectural platforms. The benchmark source code is provided on this page (drystone.c). Start by using the ***gcc*** compiler. Note, you may get some warnings while compiling this benchmark, though it compiles without warnings on the COE X86-64 systems.

In the COE computer system we provide two different ISAs, the ARM/Ubuntu/V8 (the degrees system), and Intel/Linux/X86-64 (the head nodes Theta, Iota and Zeta) and the Intel/Linux/X86-32 (the systems alpha and beta). Note, if you login to ***gateway.coe.neu.edu***, the system will select one of the head nodes for you (you do not have the ability of selecting which of the X86-64 machines you run on). For the 32-bit x86 or ARM V8 systems, you will need to ssh to those systems on port 27 after logging into the head node (e.g., ***ssh beta.coe.neu.edu -p 27***). You can also use any system you have available (e.g., ARM, MIPS, etc.). If you have difficulty connecting to any system through the VLAB, try the VPN instead.

Note: you may have to adjust the number of LOOPS specified in the source code to get a reasonable Dhrystone number. You may also have to modify the libraries used to get the C code to compile (the compilers on the COE machines work fine, though do produce warnings). Try different numbers of iterations until you get a reasonable result. Also, make multiple runs of Dhrystone to guarantee you have factored out any sampling or cold-start effects. Generally, 10 runs are enough to obtain statistical significance. Make sure to discussion this issue in your paper and report on the error in your measurements.

Run Dhrystone compiled with and without optimization (find out how to use the compiler switches, the ***man gcc*** pages should help). Explain the results you are getting. How does optimization affect the results obtained on the different architectures and why? There are probably more than a million different switches you can try. Start by using the ***-OX*** switch, where X is (0, 1, 2, 3). This is an open-ended question, that should challenge everyone to find the switches that most impact this program. Make sure to explain why a particular switch is giving you good performance (this is the most important part of the assignment). Give examples that illustrate your reasoning (e.g., provide assembly listings snipits and explain what the compiler is doing. Please do not just print out the entire listing).

The Intel compiler is also available, so compare your resulting obtained above for X86-64 with using the Intel compiler, as run on the headnode. To run the Intel compiler, the binary is ***clang*** on the Intel-based systems.

For both the X86-64 and ARM, answer the following questions:

1) What are the three most frequently executed functions?
2) What percentage of the entire execution time do they each consume?
3) How does optimization change this percentage and why?

You should use ***gprof*** (on Linux on either x86 or ARM) to profile the execution of the program. To run ***gprof***, you will need to compile your code with debug information (figure out which compiler switch to use). ***Gprof*** will help you to find the "hot" portions of the code.

Besides answering the detailed questions above, write up a summary report on your experiments. If you are not familiar with these architectures, now is the time to learn them. The vendors provide details on their instruction sets online.

**Part B:**

Next we are going to look at the ***linpack*** benchmark provided on Canvas. Compile the program on X86/Linux and provide a detailed analysis how optimizations in the compiler can improve performance. Note, this is a floating-point benchmark, so you will need to look at using math libraries and understand floating point extensions and instructions. You only need to complete this problem on only one ISA, a X86-64.

**Part C:**

Find a third benchmark on the web (neither Dhrystone nor Linpack). Compile and run it on a system of your choice. Discuss what the benchmark is designed to measure and discuss the benchmarking results you obtain on that system. Can you suggest a system where it might run more efficiently? Make sure to justify your answer.

**Part D:**

Given the differences between the ARM V8 ISA and the X86 ISA, take a basic block from Dhrystone, and compare the instructions produced by the compiler targeting each ISA. Please explain each instruction.

**Part E:**

A benchmark suite is a set of applications used to characterize the performance of a processor or system. There are many different suites available. Select two suites, and answer the following questions:

1) What application domain or system architecture is the suite designed to evaluate?
2) How is performance evaluated with the benchmark? What is the metric used to evaluate performance?
3) Cite 2 papers each (a total of 4) where each suite has been used in a research study.
4) Provide your thoughts on how the suite could be improved.