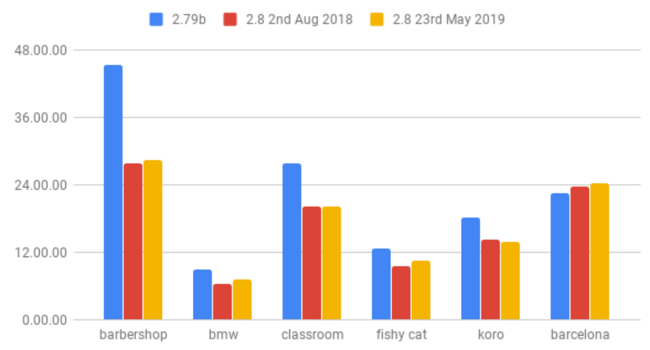
# Lab Activity #0:

# Benchmarks and Performance



In this lab, you will design a set of benchmarks and measure their performance on the OTTER (our 233 processor). The set of benchmarks (e.g. matrix multiplication and addition) are representative of many important applications that are used in systems today. This lab will give you more practice with assembly programming, the RISCV ISA, and the basic multi-cycle processor design. After implementing and testing your benchmarks, for the second part of the lab you will measure the performance of the multi-cycle processor (with a 1-cycle latency memory and a variable latency memory).

**Learning Objectives:**

- Understand basics of an instruction set architecture
- Understand a basic multicycle processor microarchitecture
- Abstraction levels, including register-transfer-level modeling
- Design principles including modularity, hierarchy, and encapsulation
- Design patterns including control/datapath split

**Introduction:** Performance, energy, power, reliability are key attributes when designing computers. Accurately measuring and comparing different computers is critical for designers. The performance of a computer is inversely proportional to is *execution time*, the total time required for a computer to complete a task (including disk accesses, memory accesses, I/O, OS overhead, CPU execution time, etc.). For many computers, designers may also be care about *throughput* or *bandwidth*, which is the total amount of work done in a given amount of time. Execution time (performance) is typically measured in seconds. The smaller the execution time, the faster the computer.

To evaluate two computer systems, a user would simply compare the execution time of a *workload* on the two computers (a workload is a set of representative programs that would typically run on a given computer). In most cases actual user workloads are not used, instead designers choose a set of *benchmarks* (i.e. programs specifically chosen to measure performance) – that will comprise the workload and hopefully accurately predict the performance of the actual workloads. Benchmarks play an important role in computer architecture, because in order to *make the common case fast* (we will discuss more on the principle later –also known as Amdahl's Law), we need to know which case is common.

**Assignment:**

1. Develop three benchmark assembly programs:

    - **matadd:** Matrix addition. Compute the addition of two MxN matrices.
    - **mul:** Multiplication. Multiply two numbers together.
    - **matmul:** Matrix multiplication. Computer the multiplication of two NxN matrices (assume the matrices have an equal number of columns and rows). *Note: I have posted C code that you can use for matmul on canvas. Of course, for more practice, you can also implement your own matmul in assembly or C.*

2. Setup the baseline OTTER (multi-cycle) and verify the processor with the test_all.s program on Canvas. You can use your OTTER design from 233 or the reference design posted on Canvas.

   a) Measure the performance of the test_all program (e.g. to cycle through all instruction test cases).

   a) Measure the performance of your benchmarks from part 1 on the OTTER, for matrices of size (3x3, 10x10, and 50x50).

   b) Measure the area and power (see vivado post-synthesis reports) of the processor on the FPGA. (e.g. LUT utilization).

3. Repeat step 2 experiments using the OTTER and a variable cycle memory (specifically a 10-cycle memory access latency). With larger main memories (e.g. off-chip DRAM), more cycles are required for the processor to communicate with the memory over an interconnect (bus). Larger memories enable larger and more complex programs (e.g. operating systems). *Note: you can either modify your 233 design or use the reference design on Canvas.*

**Deliverables:**

1. Short paragraph that provides an overview for all of your designs.

2. Answers to all questions, including bar graphs (and tables) showing your measurements.

3. HDL and other code for all designs