Exercise Set 1, EE3400, Jan-Apr 2024.

The exercises assume that you have access to a Linux or Unix operating system (e.g. Ubuntu or MacOS or Windows Subsystem for Linux on Microsoft Windows).

1. Gathering the statistics of instruction usage.
A. Select 3 programs on your computer ls, gcc, firefox/chrome (or any other scientific computing program). Disassemble them using the program called objdump, e.g.

```
$ objdump --no-leading-addr -d /bin/ls > ls.s
```

Write a python program to create a histogram of all the instructions in the generated assembly files. Plot them in a bar graph.
B. For the top 10 instructions in the list from (A) and from the table shown at https://www.agner.org/optimize/instruction_tables.pdf, find out the total time taken for the sequential execution of the programs. Can you use this information to say which program is faster or slower?

2. Assembly to opcode and vice versa. Use python for this exercise. For simplicity, implement the simpler instructions first and move on to more complex later, e.g. arithmetic (add, sub, mov, between registers and register or immediate numbers), ldr and str with registers and immediate offsets, conditional and unconditional branches.
A. Use the ARM ISA specification https://iitd-plos.github.io/col718/ref/arm-instructionset.pdf to generate opcodes corresponding to common ARM assembly instructions. Use an online ARM emulator to verify your program (e.g. https://cpulator.01xz.net/).
B. Write a dissassembler for ARM binaries. Again, verify the program by reversing the output generated by the assembler you have written.

3. This is from H&P, Comp. Arch & Design, RISC-V edition, 2020
4.10 When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from Figure 4.25, the latencies from Exercise 4.7, and the following costs:

I-mem: Reg File: Mux: ALU: Adder: D-Mem: Single Reg: Sign extend: Single Gate: Control
1000: 200: 10: 100: 30: 2000: 5: 100: 1: 500

Suppose doubling the number of general purpose registers from 32 to 64 would reduce the number of lw and sw instructions executed by 12%, but increase the latency of the register file from 150ps to 160ps and double the cost from 200 to 400. Use the following instruction mix:

R-type/I-type: lw : sw : beq

52% : 25% : 11% : 12%

4.10.1 [5] <§4.4> What is the speedup achieved by adding this improvement?

4.10.2 [10] <§4.4> Compare the change in performance to the change in cost.

4.10.3 [10] <§4.4> Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn't make sense to add more registers.

4. This is from H&P, Comp. Arch & Design, RISC-V edition, 2020
4.16 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:
IF: ID: EX: MEM: WB
250ps 350ps 150ps 300ps 200ps
Also, assume that instructions executed by the processor are broken down as follows:
ALU/LOGIC : Jump/Branch: Load: Store
45% 20% 20% 15%
4.16.1 [5] <§4.6> What is the clock cycle time in a pipelined and non-pipelined processor?
4.16.2 [10] <§4.6> What is the total latency of an lw instruction in a pipelined and non-pipelined processor?
4.16.3 [10] <§4.6> If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?
4.16.4 [10] <§4.6> Assuming there are no stalls or hazards, what is the utilization of the data memory?
4.16.5 [10] <§4.6> Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?