18-645: How to Write Fast Code

Assignment 2:- Machine-specific Kernels

General Instructions.

- 2 separate files need to be submitted via Canvas on the due date AoE. A PDF of the report and/or writeup, and a zipped files with all the code
- Both files should be named `asg_2_≪your andrew id≫`, and they should only differ in their extensions.
- The zipped file should contain the following
  1. Source code (appropriately named) for all implementations;
  2. A Makefile that compiles all implementations on one of the machine on the ECE cluster;
  3. Readme.txt that explains how to run your implementations.

1. **Computing Performance.** Consider the following floating-point vector-sum operation.

```
for (int i = 0; i != 100; ++i)
     sum = sum + a[i];
```

   (a) How many floating-point addition does it take to compute the vector-sum below (exclude the increment within the loop)?

   (b) Assume each scalar floating-point addition takes two cycles, and can be pipelined. In addition, there are 4 functional units that perform additions. What is the peak performance (number of floating-point additions per cycle) attainable on this machine?

   (c) Given the device in Part (b), how many cycles should it take to compute the vector-sum? Assume that the overhead of loading elements of `a`, and the loop are insignificant.

   (d) How many independent floating-point additions could theoretically be performed by the device in Part (b) in the time taken to compute the vector-sum? What fraction of peak performance is attained with the code shown above?

   (e) What is the minimum number of independent vector-sum operations required to utilize all the available scalar floating-point addition units?

2. **Mixed bag of instructions** A scalar `max` instruction on 2 input operands returns the larger of the 2 values. It can be performed in the following 2 steps:

```
mask = GEQ(input1, input2);        //return 1 if input 1 >= input 2; else 0
out = BLEND(input1, input2, mask); //return input1 if mask=1, else input2
```

   (a) Both `GEQ` and `BLEND` are executed on the same functional unit. If the functional unit is pipelined, only computes these 2 instructions, and each instruction has the same latency of 1 cycle, what is the number of `max` instructions that can be computed every cycle?

   (b) Assume that the latency of both instructions is now 2 cycles, how many `max` instructions can you compute every cycle? Explain how you get this answer.

   (c) Assume that there is only 1 functional unit that computes both instructions, and each instruction has a latency of 2 cycles. All other instructions (if needed) also take 2 cycles and are performed on other functional units. In addition, there are 16 registers. Design the kernel that computes the following routine:

```
function ReLU(int len, float* in, float* out)
   for (int i = 0; i < len; ++i)
      out[i] = max(0, in[i]);
```

and answer the following questions:

i. What is the size (number of outputs) of the kernel?

ii. Provide Pseudo-code for your kernel design.

iii. State any assumptions you may have made.

3. **Designing a fast matrix multiply kernel.** A matrix multiply kernel can be computed as a sum of outer-products, i.e.

$$C = \sum_{i=0}^{k-1} a_i b_i^T,$$

where a_i and b_i are column vectors, and C is a matrix of size $m \times n$. Note that this means that a_i is of length m and b_i is of length n. In addition, notice that for every pair of (a_i, b_i), every element of the C matrix can be computed independently. However, the accumulation of each outer-product is dependent on the result of the outer-products from previous iterations.

Answer the following:

(a) If a hypothetical architecture has a functional unit that computes a scalar fused multiply-and-add in 6 cycles, what is the minimum size of C, i.e. $m \times n$ in order to avoid bubbles in the pipeline?

(b) What if the hypothetical architecture has 4 functional units?

(c) Assume that the SIMD FMA instruction has a throughput of 2 instructions per cycle, and a latency of 5 cycles, what should the minimum size ($m \times n$) of C be if the kernel is implemented using SIMD instructions?

(d) Implement the SIMD matrix-multiply kernel based on your answer to the previous question (`kernel.c`). Assume that matrix A is stored in column-major order, B is stored in row-major-order. In addition, provide the following in your submission:

i. Specify what m and n should be.

ii. Provide a plot of floating point operations per clock cycle (y-axis) against the k dimension (x-axis) for k between 32 and 512 in steps of 32. Assum the size of C is $m \times n$ as computed previously.

4. **Miscellaneous.** How many hours did it take you to complete this assignment?