Stanford CS149: Parallel Computing
Written Assignment 1

Hardware Basics

Problem 1: (Graded for Correctness - 20 pts)

A. (5 pts) Consider a multi-core processor that has three cores. Each core runs at 1 GHz (1 billion operations per second). Each core is single-threaded (meaning it only maintains state for a single execution context) and can compete one single-precision floating point arithmetic operation per clock. What is the peak arithmetic throughput of the processor in terms of floating point operations per second?

B. (5 pts) Now imagine the cores from part A are upgraded so that they perform 8-wide SIMD instructions. Assuming these cores still complete one of these SIMD instructions per clock, what is the peak arithmetic throughput of the processor (in terms of floating point operations per second)?

C. (5 pts) Finally, imagine that each core from part B was a multi-threaded core that maintain execution contexts for up to four hardware threads each. What is the peak arithmetic throughput of the processor in terms of floating operations per second?

D. (5 pts) Imagine that each core from part C was further modified to support superscalar execution where the core can complete one scalar floating point operation and one 8-wide SIMD instruction per clock from the same thread (if those instructions are independent). What is the peak arithmetic throughput of the processor (in terms of floating point operations per second)?

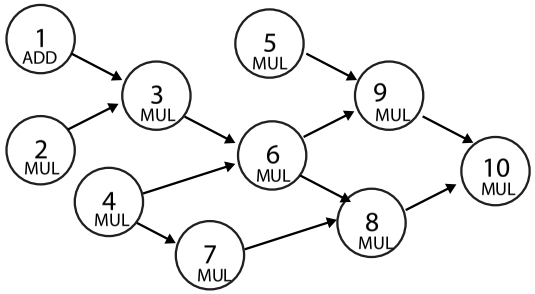
Identifying Dependencies + A Bit on Superscalar Execution

Problem 2: (Graded for Correctness - 20 pts)

A. (8 pts) Please draw the dependency graph for the following sequence of nine instructions. Label each node by the number of the instruction being performed.

```
1. ADD R2 <- R0, R1
2. MUL R3 <- R0, R1
3. MUL R3 <- R2, R3
4. SUB R4 <- R0, R1
5. MUL R2 <- R2, R3
6. MUL R3 <- R3, R4
7. MUL R2 <- R2, R3
8. MUL R4 <- R4, R1
9. MUL R0 <- R2, R4
```

B. (8 pts) **THIS PROBLEM IS INDEPENDENT FROM PART A.** Now consider the following dependency graph of instructions. Instructions are marked as ADDs and MULs.



Consider running the instruction stream on a processor that supports three-way superscalar processing. However the processor is constrained in three ways:

(a) Instructions that begin to execute in the same clock must all be MUL instructions.

(b) MUL instructions have a latency of two cycles to complete. In other words, if any instruction Y is dependent on MUL instruction X, and X executes in cycle c, then Y can only begin to execute in cycle $c + 2$. Other instructions that are independent of X can run in cycle $c + 1$.

(c) ADD instructions have a latency of one cycle to complete.

Please determine the minimum number of cycles that the instruction stream takes, and in the diagram below, please list which instructions begin in which cycle.

Clock 1	Clock 2	Clock 3	Clock 4	Clock 5	Clock 6	Clock 7	Clock 8	Clock 9	Clock 10

C. (4 pts) If your goal was to only run the instruction stream given by the DAG in part B above, is it a good idea to consider modifying the processor to support 5-way superscalar execution? Why or why not? (Hint: using the term "ILP" in your answer would be helpful.)

Problem 3: (Graded on Effort Only - 20 pts)

A. (10 pts) Consider a final exam grading pipeline where the 10 CS149 CAs work together to grade the exams. There are seven questions, and assume that each question takes 1 CA 1 minute to grade, except question 1, which takes a CA 3 minutes to grade. Imagine that the CAs arrange themselves in a pipeline in the following way:

- 2 CAs team up work on grading question 1 (CAs process different exams in parallel)
- 3 CAs work on grading question 2 (CAs process different exams in parallel)
- 1 CA (per question) works on grading each of the remaining 5 questions.

After one of the question 1 grader's completes grading question 1 for an exam, they put in the pile for the three question 2 graders. When question 2 for an exam is graded, it goes in the pile for the one question 3 grader. When that is finished, it goes in the pile for the one question four grader, etc. Given this configuration, assuming the CAs have to grade a very large number of exams, **what is the steady state throughput of the grading pipeline in terms of exams per minute?**

[ ] exams per minute

B. (10 pts) Given the same setup as part A, once a Q1 grader starts working on an exam, what's the latency of completing grading for all questions of the exam? (You can assume that if a grader is free they pick up and start grading a new exam the moment it arrives in their pile.)

[ ] minutes

Problem 4: (Graded on Effort Only - 20 pts)

Please assume that an application spawns two C++ threads that each execute the following C++ function. (The thread spawn is not shown.) Note how the function does different work based on the thread id.

```
// assume threadId is the id of the current thread
// assume numThreads evenly divides N
void my_function(float* in, float* out, int N, int numThreads, int threadId) {
   int start = threadId * N/numThreads;
   int end = start + N/numThreads;
   for (int i=0; i<1000; i++) {
      for (int j=start; j<end; j++) {
        float tmp = in[j]; // memory load
        for (int k=0; k<ITERS; k++)
          tmp *= tmp;      // 1 float op
        out[j] = tmp;      // memory store
      }
   }
```

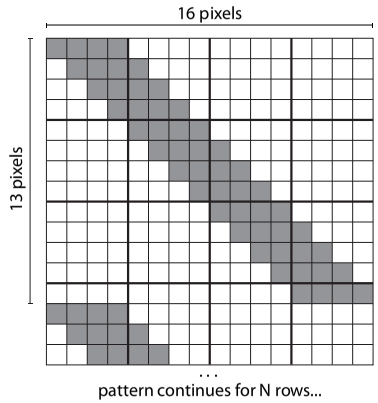
A. (10 pts) Assume you are running the application on a machine with: a single core processor that can perform one floating point operation per clock. The core has two hardware execution contexts. **The core is connected to a memory system with infinite bandwidth, a memory READ latency of 50 cycles, a memory write latency of 0, and with a cache that has four-byte cache lines and is 16 MB in size. Cache hits have zero latency.** When the application is executed on this processor with `N = 1 billion` and `ITERS=10`, what is the steady-state utilization of the core? Please put your answer in the box and then provide a brief justification. (Keep in mind the application has spawned two threads.)

[ ] % utilization

B. (10 pts) **Now consider the case where the value of `N` is reduced from 1 billion to 1 million.** Assuming all other aspects of the program, processor, and memory stay the same, you now have the option of either increasing the clock rate of the processor by 25%, or reducing the memory latency to 20 cycles. Which option will yield the highest performance. Why? **Recall that the cache size has four-byte cache lines, and is 16 MB in size**.

Problem 5: (Graded on Effort Only - 20 pts)

Consider the following image, which as a width of 16 pixels, and a height of N pixels. White pixels in the figure correspond to the value 1.0 and dark pixels correspond to the value 0.0. The first 16 rows are shown in the figure. You should assume that the pattern repeats vertically in chunks of 13 rows. (Yes, you can assume N is a multiple of 13 for simplicity.)



You write the following ISPC program to process the image above. Assume that `foo()` and `bar()` are helper functions that only perform arithmetic.

```
const int IMAGE_WIDTH = 16;
void myfunction(uniform float* input, uniform float* output, int imageHeight) {

  for (uniform int row=0; row<imageHeight; row++) {
    for (uniform int col=0; col<IMAGE_WIDTH; col+=programCount) {

      int idx = row*IMAGE_WIDTH + col + programIndex;
      float val = input[idx];   // load four bytes
      float result;

      if (val == 1.0) {
          result = foo(val);    // 1 cycle of arithmetic
      } else {
          result = bar(val);    // 10 cycles of arithmetic
      }
      output[idx] = result;     // store four bytes
    }
  }
```


A. (10 pts) Assume that you are running the code on the image above using a **ISPC gang size of 4 (`programCount=4`).** You run the code on a single core, 1 GHz processor, with 4-wide SIMD instructions, and support for eight hardware threads (execution contexts) per core. If we assume all loop indexing and memory operations are "free", how many total clocks does the code take? Your answer should be expressed in terms of N (which you can assume is divisible by 13). Please show your work and put your answer in the box.

clocks

B. (10 pts) Given your answer in part A, what is the overall average SIMD utilization of the processor over the duration of the computation? Your answer can just be an expression in terms of N or a final number between 0 and 100%. Please show your work and put your answer in the box. (Note that math does not work out to a clean integer number in this problem.)

utilization