First Trimester Examination

Name: _____

Student Id: _____

This is an open book examination. You are allowed to make use of the Internet, your class notes, etc. But you are not allowed to consult any human being, either in person or remotely. Attempt all questions.

Exam is graded out of 20, maximum grade points is 25.

Question 1 (2 points)

Congratulations, you have graduated, and you were hired as a manager at Cloudy Clouds, a company that specializes in cloud computing. You received for approval the following contract:

The company, Foggy Systems, is contracting a Web service with Cloudy Clouds with the parameters:

- Web requests arrive at random according to an exponential distribution, with average inter-arrival time of 50 msec.
- Each request is guaranteed not to exceed 5msec in processing time.
- A Service Level Agreement requires a request's response time cannot exceed 10 msec, with a very heavy penalty for violating this condition in addition of surrendering back any payment made by Foggy Systems.

Satoshi Nakamoto, an engineer in your working group, argues that this is a wonderful opportunity. Since requests arrive at such a low frequency, it is guaranteed that the response time requirements will be met. He provided you with a spreadsheet model showing his calculations and recommended approval. What is your decision? Explain.

Question 2 (3 points)

A processor chip consists of 16 cores. Each core runs at 2GHz and has two floating point units. The memory bandwidth of the entire chip is 4GB/sec. Answer the following questions:

1) What is the peak rate of floating-point operations in GF/sec? (GF: Gigaflop)?
2) Under what condition can the peak rate that you computed in (1) be realized? Be precise!
3) Considering your answer to (2), argue about the suitability or the lack thereof of the peak floating-point operations as a metric to characterize the floating-point performance of the processor.

Question 3 (5 points)

The Unnecessarily Complex Architecture (UCA) processor specifies a fully pipelined processor that executes out-of-order instructions and also allows speculative executions in a way to reduce the penalty of branches. The load-store (LS) unit of the UCA contains a queue of the memory writes that are waiting to be written to the cache (call it Q1). It also contains a separate queue of the memory writes belonging to instructions that have not committed yet (either because they ran out of their order or because they are speculative). Call the latter Q2. For either Q1 or Q2, each entry consists of an address and the data to be written.

1) Consider a memory address A. Is it possible for both Q1 and Q2 to contain an entry corresponding to A simultaneously?
2) Suppose a load instruction is issued to the LS unit requesting the data item at address A. There are three places to search (the cache, Q1 or Q2). Explain how the load instruction is to be handled, considering all possibilities.

Question 4 (5 points)

Consider the following code:

```
double a[N], b[N], c[N], d[N], e[N];// N is a constant
int i;

for(i = 0; i < N; i++) {
        e[i] = a[i] / b[i];
        e[i]+= c[i]/ d[i];
}
```

Rewrite this code so that it can be more pipeline-friendly.

Question 5 (10 points)

The computation Y = a * X + Y has long been used as a benchmark for processor performance (known as DAXPY). The typical assembly code for this function is:

```
start:  ld      f2, (r1)        # the machine has two register sets
        mul     f4, f2, f1      # a * X[i], f4 holds the results, a is in f1, and f2 holds X[i]
        ld      f6, (r2)        # Y[i],
        add     f6, f4, f6      # a*X[i] + Y[i]
        st      f6, (r2)        # Updating Y[i]
        add     r1, r1, 8       # increment X[i]'s index
        add     r2, r2, 8       # increment Y[i]'s index
        add     r3, -1          # the loop index
        bnz     r3, start
```

Assume the following components are available to you: an instruction fetch stage that can fetch up to two instructions in a cycle, an instruction decode that can issue up to two instructions in a cycle, an integer register file and a floating-point register file, each with four read ports and two write ports. A load-store unit can load a data item from the cache in two cycles and stores in one cycle. Assume that the integer ALU computes in one cycle, a floating-point multiplier that is pipelined over 3 stages, and a floating-point adder that is pipelined over 2 stages.

Design a pipeline showing how all the stages are connected. Draw a picture of the pipeline. Show how the code will execute on the pipeline. You may use out of order execution, register renaming, and branch prediction. Compute the IPC.