Here are some guidelines for your report that justifies your choice of application. It is much the same thing as I already wrote about the project in general, which you can find at **http://www.cs.washington.edu/education/courses/548/06wi/files/projectGuidelines.pdf**. Send email from the project, (i.e., one email from the two of you), with your evaluation to Andrew and me before class on Wednesday, February 1.

1. What does your application do (in a functional sense) and how does it do it (in an algorithmic sense)?

2. What is its hotspot code (you might have a few candidates here)? What percentage of the time is the hotspot executed? Is it small enough to code in WaveScalar assembly if necessary, i.e., how many lines of source or assembler is it? Would the second hottest spot be better to work with (not as hot, but smaller)? Give us some hard numbers from *gprof*.

3. From eyeballing the source and understanding what the algorithm does, can you formulate hypotheses about the hot spot's execution-time characteristics? The specific questions below are meant to guide your thinking. Some may apply to your application; some may not. You don't necessarily have to answer them explicitly. The issue behind them is: can you hypothesize now how well your hot spot can be changed to become WaveScalar-friendly code and what is the basis for your hypotheses, i.e., make arguments that show how good a candidate your application is.
    a. How many independent chains of operations lie within the same iteration? How long are these chains?
    b. How many data dependences are there across iterations? How tight are they?
    c. What is the pattern of memory references, within and across iterations? (Running sequentially through memory? Scatter/gather? Every other location? Something else?)
    d. How big (in terms of operations) are the iterations? Do they contain waves? How many?
    e. Are the same data memory locations used by multiple operations? Multiple iterations?
    f. How frequently do branches occur?
    g. Is it currently threaded? If so, at what granularity? Do you think this could be changed?
    h. Ask your own questions as well.

4. Compile the hotspot code using WaveScalar's binary translator. Is it necessary to compile the whole program, or can you construct a wrapper for the hotspot code and just compile that (like was done in `lcs`)? (Hint: Try to do the latter). If you have a large hotspot function, can you break it into smaller functions? Does this change the performance significantly?

5. Evaluate your benchmark's baseline performance on WaveScalar. You can answer most of these questions by looking at the output file from the *kahuna* simulator after running your benchmark. Some of the statistics will make sense, and some will not. Try to pick out the relevant statistics, and ask questions if you can't find the right ones. (Hint: memory system ≈ cache, performance ≈ IPC, AIPC, and total cycles to execute)
    a. Simulate the hotspot on the WaveScalar simulator. How fast is it?
    b. How much parallelism does it exhibit and what kind of parallelism is it? How are you going to determine this?
    c. What is the performance of the memory system? How are you evaluating that?
    d. Are there any other metrics you think are important to look at?
    e. Are there any other metrics you think are important to gather to evaluate your baseline? Are there any other metrics you think are important to gather to evaluate the WaveScalar-friendly design? Do they need to be added to the WaveScalar simulator?