Build off of Mircro C as much as possible

Need to write:
1. Lexer/Parser
2. Static semantics
3. Code generation (MIDI “Bytecode” compiler)
4. Assembler (MIDI compiler*)
*written in Java

Different people will own different files. Start with the microC code and write unit tests as you complete parts that should not be broken in our language. If you run into ambiguities or problems, e-mail the mailing list. Try to make golden input/output files so that unit testing your code is independent of other people breaking the code base. For the assembler, we’re going with that sample Java code we found that does CSV to MIDI. The compiler will output a CSV file that nearly matches the specifications there, but will be missing some information that’s always default, such as the instrument type. Person #4 should write up a specification for the CSV file before person 1 starts on the assembler.

Files:
Top-level (2)
Scanner (1)
Parser** (3)
AST (2)
Bytecode (4)
Compiler** (4)
Assembler (1)

** big jobs

Responsibilities

1 akiva/ye
2 akiva/ye
3 fred
4 ben

Deadlines (all Saturdays at 11:59 pm)

We have a little more than a month, but that time will pass quickly. We need to do this and save enough time for unanticipated errors.

11/20: Finish project plan and arch design sections of final report

Mostly complete scanner, top level, and AST
11/27: Each person should have finished at least one unit test that works for his component
Make major headway on parser and compiler
12/1: HW3*
12/4: Finish testing plan
12/11: Finish overall system (source should compile into MIDI files)
12/13: Final exam*
12/18: Finish up final sections of report

Final Report Sections (Due Dec 22)
1. Introduction: the proposal
2. Language Tutorial
3. Language Reference Manual
4. Project Plan
5. Architectural Design
6. Test Plan
7. Lessons Learned
8. Complete listing

Include the following sections:

1. Introduction
    - Include your language white paper.
2. Language Tutorial
    - A short explanation telling a novice how to use your language.
3. Language Manual
    - Include your language reference manual.
4. Project Plan - Deadline Saturday 11/20 11:59 pm, committed to repository (no late days - if you're late, you have to create a Gantt chart)
    - Identify process used for planning, specification, development and testing (ben)
    - Include a one-page programming style guide used by the team (akiva)
        i. copy/summarize this
    - Show your project timeline (done)
    - Identify roles and responsibilities of each team member (done)
    - Describe the software development environment used (tools and languages) (fred)
    - Include your project log (SVN commits)
5. Architectural Design - Deadline Saturday 11/20 11:59 pm, committed to repository
    - Give block diagram showing the major components of your translator (ben)
    - Describe the interfaces between the components (fred)
    - State who implemented each component (done)
6. Test Plan - Deadline Saturday 12/4 11:59 pm, committed to repository
    - Show two or three representative source language programs along with the target language program generated for each

    - Show the test suites used to test your translator
    - Explain why and how these test cases were chosen
    - What kind of automation was used in testing
    - State who did what
7. Lessons Learned
    - Each team member should explain his or her most important learning
    - Include any advice the team has for future teams
8. Appendix
    - Attach a complete code listing of your translator with each module signed by its author

Presentation (waiting for Edward’s reply for the date)