COMP 520 - Compilers

Lecture 18 (April 7, 2016)

PA4 submission details

- PA3 still not fully graded
  - but pa3_tests will be available on the web this weekend

- PA4 due next Wednesday (4/13)
  - Testing will emphasize code generation, not pa3 regression testing
The PA4 checkpoint

• Your pa4 directory should have
  – miniJava
    • Compiler.java
    • AbstractSyntaxTrees
    • SyntacticAnalysis
    • ContextualAnalysis
    • CodeGenerator (new subpackage)

  – mJAM (supplied)
    • Interpreter.java
    • Disassembler.java
    • Instruction.java
    • Machine.java
    • ObjectFile.java
    • Note: Test.java is not needed or used

• mJAM is recommended to check everything is working
  – pa4 testing will not copy your mJAM, it uses class files from mJAM as distributed
Compiling and running miniJava programs

• **Compiling test.java**
  - `java miniJava/Compiler test.java`
    • Use `ObjectFile` class to write `test.mJAM` (note spelling!), be sure that it is written in the same directory as `test.java`
    • do not execute the generated program or call the debugger as part of compilation!

• **Disassembling test.mJAM**
  - `java mJAM/Disassembler test.mJAM`
    • should write `test.asm` in same directory as `test.mJAM`

• **Executing test.mJAM**
  - `java mJAM/Interpreter test.mJAM`
    • `System.out.println` results from `test.java` will appear on stdout prefixed by “>>> “
    • Use HALT (4) in the generated code to display a snapshot of the data store at a given point (or use the debugger below).

• **Debugging test.mJAM**
  - `java mJAM/Interpreter test.mJAM test.asm`
    • Show machine data store and state, show code, set/remove breakpoints, single instruction execution
    • Type “?” for help
Check results

- To compare java semantics and miniJava semantics of program foo.java

1. Run as java program
   - `javac foo.java`
   - `java foo.class`

2. Run as miniJava program
   - `java miniJava/Compiler foo.java`
   - `java mJAM/Interpreter foo.mJAM`