Compiler Assignment 1

Group D: Herold, Jakobitsch, Lercher

Ant

The ant build file is located at `src/yapl/test/backend/sm/build-dist.xml`.

Targets

Ant target	Description	Parameters to overwrite	Notes
`init`	creates the directory		
`compile-java`	compile the sources		
`run-backend`	generate MJ bytecode	testname (which java file to run) outfile (where to write the bytecode)	
`run-mj`	run MJ code with mjvm	outfile (which bytecode file to run)	depends `run-backend`
`eval`	run a single test file (generate mj and run)	testname (which java file to run) runtimeoutput (where to write the output)	
`eval-all`	run all tests		depends `eval`
`clean`	remove generated files		
`decode`	decode generated MJ file	outfile (which bytecode file to decode)	prints output
`coverage`	run coverage for a single file	testname (which java file to run)	uses JaCoCo
`coverage-all`	run coverage for all tests		depends `coverage`
`cov-report`	create full coverage report		depends `coverage-all`

Notes:
`testname` just requires the name of the test class under `yapl.test.backend.sm`.
`testname` automatically updates `mainclass`, `outfile`, `runtimeoutput`, `truefile`, and `coveragefile`.

Project Files

With the exception of tests, all new code is in package `yapl.impl` running with Java SDK 13.

- `BackendMJ` implements BackendBinSM
- `ByteUtils` helper methods for byte manipulation
- `Instruction` enum of MJ opcodes with byte values
- `OperandType` enum of explicit MJ operand types (s8, s16, s32)
- `Procedure` represents a defined procedure. keeps track of allocated local variables

Tests are located in package `yapl.test.backend.sm`.

Coverage

Coverage was calculated with the Java Code Coverage (JaCoCo) plugin. The library is located at `lib/jacocoant.jar`.
The coverage report for `yapl.impl` is generated over all tests. See ant target `cov-report`.

Reported coverage for `yapl.impl`: 100.0%