Changes and Issues for uPascal VM: 01/31/2013

Patrick Berens(patrick.berens@cs.montana.edu)

I will be happy and able to fix any problems students experience so feel free to forward me their issues when they arise. I will get to them as soon as my schedule will allow.

Highlight of Features

See newMicromachineDefintion.pdf for all features, however here are a few to take note of:

- Write and Writeln now supported(see Issues below)
- PRTS and PRTR – May help students debug: prints stack and registers respectively.
- Automated Testing Scripts – For future development or optionally for students.
    - If provide scripts to students, or ask students to write themselves, it allows students to automatically run an unlimited number of test files. It compiles each with their compiler and fpc. Then runs theirs through the VM and executes fpc's. It compares the output, thereby ensuring their compiler/VM is producing the same output as FPC would.
    - I wrote these last Spring and it was invaluable when testing and checking for regressions in both my compiler and the Virtual Machine. They also could be used for quick grading.

Directory Structure:

- docs – Folder of documents describing VM and uPascal
    - uPascal – Folder of updated uPascal Language definitions
        - uPascal_CFG.(pdf/docx) – with string/float additions.
        - uPascal_EBNF.htm – with string/float additions.
        - uPascal_EBNFOriginal.htm
        - uPascal_Tokens.htm – Your list of uPascal tokens
    - newMicroMachineDefinitions.(pdf/docx)
    - original_uMachine_definition.pdf
    - source_Directions.txt – Original def of source with a few modifications.
    - uMachineQuickReference.txt – List of instructions(copy of table in definitions)
- micro-vm – Folder of code for VM. See source_Directions.txt.
    - compile – Script to compile and run a file of name "uMachine_code.il".
    - ….Source code
- release – Folder of executables, Compiler, and Tests
    - **execute - Virtual machine executable. Run as: ./execute some_vm_file.il**
    - Test Suite - Folder of tests and test scripts(See "How to Use Test Suites" below)
        - Automatic - Folder of automatic tests
        - Manual - Folder of tests which require human interaction(reads) or will fail(error tests)
        - batchAutomaticTests - Script to run all tests in automatic folder
        - batchManualTests - Script to run all tests in manual folder
        - fullTest - Script to run one test, called by both batch test scripts.
        - microPascalCompiler.jar – Patrick/Dain/Callum's Compiler.
        - resetAutomatic - Script to reset automatic folder back to original state.
        - resetManual – Script to reset manual folder back to original state.

Documentation(Please check and correct these before distribution!!!!!)

I have attempted to update documents to reflect the addition of floats and strings. This was done in both the CFG and EBNF. However in reality, I only added "float" and "string" wherever I saw "integer"(like in "type"). I also added float_literal and string_literal wherever integer_literal was used. This seemed to be enough, but I may have made invalid assumptions about its simplicity.

Issues

- Write and Writeln: Before, write was defined as writeln. This made the virtual machine incompatible with the free pascal compiler(fpc). This has been corrected and writeln is now also available. **This makes the VM no longer backwards-compatible with the old virtual machine since write instructions all print on one line. (See Testing)**
- Read instructions no longer print "? " as a prompt: This makes the output easier to parse for students who wish to write their own automated test scripts. However, can use "write" now to prompt properly(write 'Please enter a float between 0 and 10: ').
- FPC mod bug: The FPC program(version 2.4.0 on Esus) contains a bug where using a negative constant produces invalid output. For example, -10 mod 3, would not be correct. This took me forever to find. Specifically, test "B_1booleanTest.txt" has "if ((20+4) div (3+3) > -10000 mod (63+14)) then" on line 96. This produces incorrect results for FPC and therefore, this is the only test which doesn't match the VM output. The VM currently uses C's % operator for mod. Pascal defines mod as: I mod J = I – (I div J) * J, but again if you use a negative constant, FPC doesn't follow this definition. **Solution: Don't use negative constants with mod in tests.**

Testing Performed

- TestSuite: Has both an automated tests and manual tests(tests which require "read" input) which run through scripts. This may help future developers or students if you want to release the scripts. **Note: It also contains the jar of my compiler, so remove this if you are distributing it to students. They would have to replace it with their own.**
- I never updated my compiler to support Strings or Floats. This means testing for Floats and Strings was done by writing VM instructions by hand. I tried to be as thorough as possible, but eventually I stopped before I went insane…tweaking or minor issues may be experienced.

How to use TestSuite:

- These work for me, but may require some tweaking given a different set of assumptions.
- As described previously, the batch test scripts will run all tests within a folder through fpc and through the students compiler. It will then execute the fpc produced executable and run the IL file through the VM. It then compares the output and places passed tests in a new folder. This provides quick validation and regression testing.
- TestSuite Tests – **All your provided tests have been converted so "write" instructions are now "writeln"**:
  - Automated: Includes all your provided test files that didn't have a read instruction, almost all the read instruction tests with dummy values inserted(some couldn't be converted), and a bunch of random test files I made at some point. "B_1booleanTest.txt" doesn't pass(See Issues above).
  - Manual: All your original read tests and your one error test. There is a file within called "readInputList.list" which contains an ordered list of values which need to be entered whenever prompted during test run.

Instructions to Run Tests:

- Required: Linux, FPC, student compiler named “microPascalCompiler.jar”.
- Compiler must output IL file named: “uMachine_code.il”
- Must be in same directory structure as in repository/zip file.

1. From within TestSuite Folder, run ./batchAutomaticTests
2. Script will blurt out tons of crap, once script is complete cd into Automatic folder
3. Any scripts within Automatic that wasn’t copied into finishedTests folder failed. Those that passed will be within finishedTests, both in their original form, and with the IL code produced.
4. From within TestSuite folder, run ./resetAutomaticTests
5. Now Automatic Folder is reset and batchAutomaticTests can be run again.

1. Open readInputList.list within Manual folder
2. From within TestSuite Folder, run ./batchManualTests
3. When prompted for input, enter first entry in readInputList.list.
4. Continue this process until list ends and script finishes.
5. Manual folder will again contain all tests which have failed.
6. ./resetmanulaTests

VM Hard Limits(previously undocumented)

- RAM, Code, and Label arrays are all set to 1024. That means 1024 instructions max.
- Strings only allow 1024 characters. This was to reduce the need for maintenance (potential memory management problems leading to crash/mem dump).