MicroStackMachine

Course: Advanced Programming, Block 1 2010

Deadline: 12:00, September 24, 2010

1 Machine Description

The virtual machine MicroStackMachine (MSM) consists of:

- a program counter (PC),
- two integer registers (A and B), and
- and a stack of integers.

A program for the MSM is a zero-indexed sequence of instructions. The execution of a program in the MSM might halt with an error.

Instructions

MSM has the following instructions:

- `PUSH` n pushes the integer constant n on top of the stack.
- `POP` removes the top element of the stack.
- `DUP` duplicate the top element of the stack.
- `SWAP` swaps the two top elements of the stack.
- `LOAD_A` and `LOAD_B` pushes the content of register A and B respectively on the stack.
- `STORE_A` and `STORE_B` remove the top element of the stack and store it in register A and B respectively.
- `NEG` negate the top element of the stack.
- `ADD` removes the two top elements of the stack, adds them, and pushes the result to the top of the stack.
- `JMP` sets PC to the value of the top element of the stack and removes the top element of the stack.
- `CJMP` i removes the top element of the stack, if it is less than zero the PC is set to i; otherwise the PC is incremented by one.
- `HALT` halt the machine without an error.

The PC is incremented by one after each instruction that is not `JMP`, `CJMP`, or `HALT`.

Errors

Depending on the state of the MSM, the execution of an instruction can fail. As a result the MSM halts with an error. In the following situations the MSM halts with an error:

- `POP`, `DUP`, `STORE_A`, `STORE_B`, `NEG`, `JMP`, or `CJMP` is executed on an empty stack.
- `SWAP` or `ADD` is executed on a stack containing less than two elements.
- The PC changes to a value that no longer points to an instruction. That is, it becomes greater or equal to the size of the program or negative.

2 Tasks

(a) Declare types for modelling a MSM machine state.

(b) Use a monad to write an interpreter for MSM programs. The interpreter function must be named `interp`.

(c) Make some test examples that show your interpreter works.

(d) Discuss the advantages and/or disadvantages of this approach.

3 Optional Tasks

(e) Add extra arithmetic instructions, such as multiplication for instance.

(f) Add an instruction, `FORK`, for concurrency. `FORK` should create a copy of the stack, push `0` (zero) on the stack of the parent thread, and `1` (one) on the stack of the child thread. A concurrent MSM program stops when all threads have stopped.

(g) Add `READ` s and `WRITE` s that can read and write integers from the console. `READ` will push the integer it read on the stack while `WRITE` will remove the top element. The argument s is prompt string printed to console for user assistance before the integer is read/written (the prompt string can be empty).

4 Hand-in Instructions

- Your hand-in must be submitted via Absalon. Use the *Submit answer* feature on the page where you downloaded the assignment (`Assignments/MicroStackMachine`).

- The names (as written in Absalon) of all group members must be included in a comment in the beginning of the file you upload (this is also the case for singleton groups).
- You should comment your code properly, especially if you doubt the correctness or if you think there is an easier way. A good comment should explain your ideas and provide insight.