MicroStackMachine

Course: Advanced Programming, Block 1 2011

Deadline: 12:00, September 23, 2011

1 Machine Description

The virtual machine MicroStackMachine (MSM) consists of:

- a program counter (PC),
- a dynamically allocated number of integer registers, 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.
- `NEWREG` n allocates a new register n.
- `LOAD` removes the top element, n, of the stack, and pushes the content of register n on the stack.
- `STORE` removes the two top elements, m and n, of the stack, and stores the value m in register n. That is, store the top of the stack in the register denoted by the second topmost element of the stack, and remove both elements from the stack.
- `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`, `LOAD`, `NEG`, `JMP`, or `CJMP` is executed on an empty stack.
- `STORE`, `SWAP`, or `ADD` is executed on a stack containing less than two elements.
- `LOAD` or `STORE` is used on a register that has not been allocated with `NEWREG`.
- `NEWREG` is called with an already allocated register.
- 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) In case of an error, your interpreter should stop and return a proper error value. Make sure that you document which error protocol you are using.

(d) Make some systematic test examples that shows that your interpreter works.

(e) Discuss the advantages and/or disadvantages of the data structures code organisation you have used.

3 Hand-in Instructions

- Your hand-in must be submitted via Absalon.
- The names (as written in Absalon) of all group members must be included in a comment in the beginning of all files you upload (also for singleton groups).
- You should comment your code properly, especially if you doubt the correctness or if you think there is a more elegant way to write a certain piece of code. A good comment should explain your ideas and provide insight.

4 Optional Tasks

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

(g) 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.

(h) 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).