

CS232F: Programming Languages
II semester 2001-02

Programming Assignment 3: The VMC Machine.
Latest update: April 1, 2002

VMC: INTRODUCTION

Consider a simple two-stacks-with-memory machine.

V – A stack of strings used for decision making and evaluation.

M – A random access memory of non-negative integer values to store and retrieve values of variables.

C – The control stack to store the operations to be performed on the values in V.

This machine could be implemented in hardware too. We may use such a machine to execute WHILE programs. For simplicity assume

- that all operators (including commands) are represented by a single character code.
- Besides the usual operators, like “+”, “-”, “*”, “=” and “<” we have the following character codes for the various operations:

code	operation	sample	stands for
a, . . . , z	variable names		
+	addition	ab+	a + b
-	subtraction		
*	multiplication		
/	div	ab/	a div b
%	mod		
=	equality of numbers	ab=	a = b
<	less-than	ab<	a < b
~	not	ab=~	not (a=b)
	or	ab<ab=	a ≤ b
@	skip		
:	assignment	xe:	x := e
;	composition	cd;	c; d
?	if-then-else	cdb?	if b then c else d
!	while-do	cb!	while b do c

- that each program is a *single* command written in *postfix* form.

INSTRUCTIONS AND NOTES

1. The operational semantics of the WHILE language specifically for this machine is given separately as a collection of transitions.
2. the operational semantics may however transform a program into a sequence of more basic commands written in *left-to-right* order as *strings*.
3. Expressions in commands will further be transformed into a string in postfix notation, which then have to be evaluated on the stacks and the results stored in memory if necessary.
4. The constraint on the number of variables indicates that you may use a ML record of 26 fields to simulate a RAM.
5. The value stack V contains only strings. The values “0” and “1” respectively denote “false” and “true”.
6. The control stack C is a stack of strings. Hence while moving between C and M or M and V it may be necessary to transform strings of digits to integers and vice-versa.

7. Define a single signature and structure parametrized by a type variable for the type `Stack` with the operations

```

type 'a Stack
exception EmptyStack
create: -> 'a Stack
push : 'a * 'a Stack -> 'a Stack
pop  : 'a Stack -> 'a Stack
top  : 'a Stack -> 'a
empty: 'a Stack -> bool

```

8. Use the same signature and structure to create the stacks `V` and `C` where

```

V : string Stack
C : string Stack

```

Semantics of WHILE on the VMC-machine

The configuration of the machine is defined by the contents of the two stacks and the memory. Assume that all variables are non-negative integers.

The following is the semantics of the WHILE language expressed in terms of the transitions of the VMC-machine. Notational conventions used in the specification of the semantics are as follows:

- “ V ”, “ M ”, and “ C ” denote respectively the current contents of the Value stack, the Memory and the Control stack respectively.
- The letter “ m ” denotes an integer constant value. Since integer constants can be more than one character long, assume that every integer is terminated by the “backquote” symbol “ $`$ ”.
- The letter “ x ” denotes an arbitrary variable and $\#x$ denotes the value of the variable in memory.
- “ $\circ$ ” denotes any of the *five* binary operations on natural numbers.
- “ $M\{x \leftarrow m\}$ ” denotes the storing of the value “ m ” in the memory location of the variable “ x ”.
- “ t ”, “ u ” and “ v ” denote truth-values (“0” or “1”). For any truth value t , “ $\bar{t}$ ” denotes its logical complement (“1” or “0”).
- “ b ” denotes a boolean expression.
- “ c ” and “ d ” denote commands.
- “ ε ” denotes the value of an empty stack.
- For any stack value S , $a.S$ denotes the result of pushing a on to the stack. In other words, any, stack value of the form “ $m.C$ ” denotes that the value “ m ” is on the top of the control stack. Similarly “ $0.c_0.c_1.V$ ” denotes that “0”, “ c_0 ” and “ c_1 ” are respectively the top three elements on the Value stack.

Configurations The set Γ of *configurations* of the VMC-machine consists of triples of the form $\langle V, M, C \rangle$. The *initial* configurations are of the form $\langle \varepsilon, M_0, C \rangle$ where M_0 denotes that all variables are initialized to 0, and C is a command (equivalently a program). The *terminal* (or *final*) configurations are of the form $\langle \varepsilon, M, \varepsilon \rangle$ with both the control stack and the value stack being empty.

Semantics of value expressions

$$\begin{array}{lll}
E.m & \langle V, M, m.C \rangle & \rightarrow \langle m.V, M, C \rangle \\
E.x & \langle V, M, x.C \rangle & \rightarrow \langle \#x.V, M, C \rangle \\
E.\circ 00 & \langle V, M, mn \circ .C \rangle & \rightarrow \langle n.m.V, M, \circ.C \rangle \\
E.\circ 01 & \langle V, M, mx \circ .C \rangle & \rightarrow \langle m.V, M, x.\circ.C \rangle \\
E.\circ 10 & \langle V, M, xm \circ .C \rangle & \rightarrow \langle V, M, x.m.\circ.C \rangle \\
E.\circ 10 & \langle V, M, xy \circ .C \rangle & \rightarrow \langle V, M, x.y.\circ.C \rangle \\
E.\circ & \langle m.n.V, M, \circ.C \rangle & \rightarrow \langle p.V, M, C \rangle \quad p = n \circ m
\end{array}$$

Semantics of boolean expressions There would be 4 transitions each named

- $B. = 00$ and $B. < 00$
- $B. = 01$ and $B. < 01$
- $B. = 10$ and $B. < 10$
- $B. = 11$ and $B. < 11$

similar to those for $E.\circ$ which yield truth values instead as results. In addition we require the following transitions.

$$\begin{array}{lll}
B.t & \langle V, M, t.C \rangle & \rightarrow \langle t.V, M, C \rangle \\
B.= & \langle m.n.V, M, =.C \rangle & \rightarrow \langle t.V, M, C \rangle \\
B.< & \langle m.n.V, M, <.C \rangle & \rightarrow \langle t.V, M, C \rangle \\
B.\sim & \langle t.V, M, \sim.C \rangle & \rightarrow \langle u.V, M, C \rangle \quad u = \bar{t} \\
B.| & \langle u.t.V, M, |.C \rangle & \rightarrow \langle v.V, M, C \rangle \quad v = t \vee u
\end{array}$$

Semantics of Commands

$$\begin{array}{lll}
C.@ & \langle V, M, @.C \rangle & \rightarrow \langle V, M, C \rangle \\
C.:1 & \langle V, M, x.e.:.C \rangle & \rightarrow \langle x.V, M, e.:.C \rangle \\
C.:2 & \langle m.x.V, M, :.C \rangle & \rightarrow \langle V, M\{x \leftarrow m\}, C \rangle \\
C.; & \langle V, M, cd;.C \rangle & \rightarrow \langle V, M, c.d.C \rangle \\
C.? & \langle V, M, cdb?.C \rangle & \rightarrow \langle d.c.V, M, b?C \rangle \\
C.?0 & \langle 0.d.c.V, M, C \rangle & \rightarrow \langle V, M, d.C \rangle \\
C.?1 & \langle 1.d.c.V, M, C \rangle & \rightarrow \langle V, M, c.C \rangle \\
C.! & \langle V, M, cb!.C \rangle & \rightarrow \langle c.b.V, M, b!.C \rangle \\
C.!0 & \langle 0.c.b.V, M, !.C \rangle & \rightarrow \langle V, M, C \rangle \\
C.!1 & \langle 1.c.b.V, M, !.C \rangle & \rightarrow \langle V, M, c.cb!.C \rangle
\end{array}$$

Programming

1. Implement the VMC machine, and
2. Implement an EXECUTOR which executes the WHILE programs as specified above on the VMC.

Exercise (to be done during power-cuts or when the INTEL Lab is down!)

Prove that this machine implements exactly the operational semantics defined for WHILE programs in the lectures.