JOURNAL OF FORMALIZED MATHEMATICS
Volume 11, Released 1999, Published 2003
Inst. of Computer Science, Univ. of Białystok

The SCMPDS Computer and the Basic Semantics of its Instructions[1]

Jing-Chao Chen
Shanghai Jiaotong University

Summary. The article defines the SCMPDS computer and its instructions. The SCMPDS computer consists of such instructions as conventional arithmetic, "goto", "return" and "save instruction-counter" ("saveIC" for short). The address used in the "goto" instruction is an offset value rather than a pointer in the standard sense. Thus, we don't define halting instruction directly but define it by "goto 0" instruction. The "saveIC" and "return" equal almost call and return statements in the usual high programming language. Theoretically, the SCMPDS computer can implement all algorithms described by the usual high programming language including recursive routine. In addition, we describe the execution semantics and halting properties of each instruction.

MML Identifier: `SCMPDS_2`.

WWW: `http://mizar.org/JFM/Vol11/scmpds_2.html`

The articles [16], [15], [22], [2], [18], [5], [6], [20], [1], [17], [7], [3], [13], [23], [8], [10], [4], [11], [12], [9], [19], [21], and [14] provide the notation and terminology for this paper.

1. The SCMPDS Computer

In this paper x denotes a set and i, k denote natural numbers.

The strict AMI SCMPDS over $\{\mathbb{Z}\}$ is defined by:

(Def. 1) $\mathrm{SCMPDS} = \langle \mathbb{N}, 0, \text{Instr-Loc}_{\mathrm{SCM}}, \mathbb{Z}_{14}, \text{SCMPDS-Instr}, \text{SCMPDS-OK}, \text{SCMPDS-Exec} \rangle$.

Let us observe that SCMPDS is non empty and non void.

We now state three propositions:

(1) There exists k such that $x = 2 \cdot k + 2$ iff $x \in \text{Instr-Loc}_{\mathrm{SCM}}$.

(2) SCMPDS is data-oriented.

(3) SCMPDS is definite.

One can check that SCMPDS is IC-Ins-separated, data-oriented, and definite.

We now state two propositions:

(4)(i) The instruction locations of SCMPDS $\neq \mathbb{Z}$,

(ii) the instructions of SCMPDS $\neq \mathbb{Z}$, and

(iii) the instruction locations of SCMPDS $\neq$ the instructions of SCMPDS.

[1]This work was done while the author visited Shinshu University March–April 1999.

(5) $\mathbb{N} = \{0\} \cup \text{Data-Loc}_{\text{SCM}} \cup \text{Instr-Loc}_{\text{SCM}}$.

In the sequel s denotes a state of SCMPDS.

One can prove the following two propositions:

(6) $\mathbf{IC}_{\text{SCMPDS}} = 0$.

(7) For every SCMPDS-State S such that $S = s$ holds $\mathbf{IC}_s = \mathbf{IC}_S$.

2. The Memory Structure

An object of SCMPDS is called an Int position if:

(Def. 2) It $\in \text{Data-Loc}_{\text{SCM}}$.

The following propositions are true:

(9)[1] If $x \in \text{Data-Loc}_{\text{SCM}}$, then x is an Int position.

(10) $\text{Data-Loc}_{\text{SCM}}$ misses the instruction locations of SCMPDS.

(11) The instruction locations of SCMPDS are infinite.

(12) Every Int position is a data-location.

(13) For every Int position l holds $\text{ObjectKind}(l) = \mathbb{Z}$.

(14) For every set x such that $x \in \text{Instr-Loc}_{\text{SCM}}$ holds x is an instruction-location of SCMPDS.

3. The Instruction Structure

We use the following convention: d_1, d_2, d_3, d_4, d_5 are elements of $\text{Data-Loc}_{\text{SCM}}$ and k_1, k_2, k_3, k_4, k_5, k_6 are integers.

Let I be an instruction of SCMPDS. Observe that $\text{InsCode}(I)$ is natural.

In the sequel I is an instruction of SCMPDS.

The following proposition is true

(15) For every instruction I of SCMPDS holds $\text{InsCode}(I) \leq 13$.

Let s be a state of SCMPDS and let d be an Int position. Then $s(d)$ is an integer.

Let m, n be integers. The functor $\text{DataLoc}(m,n)$ yielding an Int position is defined by:

(Def. 4)[2] $\text{DataLoc}(m,n) = 2 \cdot |m+n| + 1$.

We now state several propositions:

(16) $\langle 0, \langle k_1 \rangle \rangle \in \text{SCMPDS-Instr}$.

(17) $\langle 1, \langle d_1 \rangle \rangle \in \text{SCMPDS-Instr}$.

(18) If $x \in \{2,3\}$, then $\langle x, \langle d_2, k_2 \rangle \rangle \in \text{SCMPDS-Instr}$.

(19) If $x \in \{4,5,6,7,8\}$, then $\langle x, \langle d_3, k_3, k_4 \rangle \rangle \in \text{SCMPDS-Instr}$.

(20) If $x \in \{9,10,11,12,13\}$, then $\langle x, \langle d_4, d_5, k_5, k_6 \rangle \rangle \in \text{SCMPDS-Instr}$.

In the sequel a, b, c are Int positions.

Let us consider k_1. The functor goto k_1 yields an instruction of SCMPDS and is defined by:

(Def. 5) goto $k_1 = \langle 0, \langle k_1 \rangle \rangle$.

[1] The proposition (8) has been removed.

[2] The definition (Def. 3) has been removed.

Let us consider a. The functor return a yields an instruction of SCMPDS and is defined by:

(Def. 6) return $a = \langle 1, \langle a \rangle \rangle$.

Let us consider a, k_1. The functor $a{:=}k_1$ yields an instruction of SCMPDS and is defined as follows:

(Def. 7) $a{:=}k_1 = \langle 2, \langle a, k_1 \rangle \rangle$.

The functor saveIC(a, k_1) yields an instruction of SCMPDS and is defined as follows:

(Def. 8) saveIC$(a, k_1) = \langle 3, \langle a, k_1 \rangle \rangle$.

Let us consider a, k_1, k_2. The functor $(a, k_1) <> 0$_goto k_2 yields an instruction of SCMPDS and is defined as follows:

(Def. 9) $(a, k_1) <> 0$_goto $k_2 = \langle 4, \langle a, k_1, k_2 \rangle \rangle$.

The functor $(a, k_1) <= 0$_goto k_2 yields an instruction of SCMPDS and is defined as follows:

(Def. 10) $(a, k_1) <= 0$_goto $k_2 = \langle 5, \langle a, k_1, k_2 \rangle \rangle$.

The functor $(a, k_1) >= 0$_goto k_2 yielding an instruction of SCMPDS is defined as follows:

(Def. 11) $(a, k_1) >= 0$_goto $k_2 = \langle 6, \langle a, k_1, k_2 \rangle \rangle$.

The functor $a_{k_1}{:=}k_2$ yields an instruction of SCMPDS and is defined by:

(Def. 12) $a_{k_1}{:=}k_2 = \langle 7, \langle a, k_1, k_2 \rangle \rangle$.

The functor AddTo(a, k_1, k_2) yields an instruction of SCMPDS and is defined by:

(Def. 13) AddTo$(a, k_1, k_2) = \langle 8, \langle a, k_1, k_2 \rangle \rangle$.

Let us consider a, b, k_1, k_2. The functor AddTo(a, k_1, b, k_2) yielding an instruction of SCMPDS is defined as follows:

(Def. 14) AddTo$(a, k_1, b, k_2) = \langle 9, \langle a, b, k_1, k_2 \rangle \rangle$.

The functor SubFrom(a, k_1, b, k_2) yielding an instruction of SCMPDS is defined by:

(Def. 15) SubFrom$(a, k_1, b, k_2) = \langle 10, \langle a, b, k_1, k_2 \rangle \rangle$.

The functor MultBy(a, k_1, b, k_2) yielding an instruction of SCMPDS is defined by:

(Def. 16) MultBy$(a, k_1, b, k_2) = \langle 11, \langle a, b, k_1, k_2 \rangle \rangle$.

The functor Divide(a, k_1, b, k_2) yielding an instruction of SCMPDS is defined by:

(Def. 17) Divide$(a, k_1, b, k_2) = \langle 12, \langle a, b, k_1, k_2 \rangle \rangle$.

The functor $(a, k_1) := (b, k_2)$ yields an instruction of SCMPDS and is defined as follows:

(Def. 18) $(a, k_1) := (b, k_2) = \langle 13, \langle a, b, k_1, k_2 \rangle \rangle$.

Next we state a number of propositions:

(21) InsCode(goto k_1) $= 0$.

(22) InsCode(return a) $= 1$.

(23) InsCode$(a{:=}k_1) = 2$.

(24) InsCode(saveIC(a, k_1)) $= 3$.

(25) InsCode($(a, k_1) <> 0$_goto k_2) $= 4$.

(26) InsCode($(a, k_1) <= 0$_goto k_2) $= 5$.

(27) $\mathrm{InsCode}((a,k_1) >= 0\_\mathrm{goto}\, k_2) = 6$.

(28) $\mathrm{InsCode}(a_{k_1}{:=}k_2) = 7$.

(29) $\mathrm{InsCode}(\mathrm{AddTo}(a,k_1,k_2)) = 8$.

(30) $\mathrm{InsCode}(\mathrm{AddTo}(a,k_1,b,k_2)) = 9$.

(31) $\mathrm{InsCode}(\mathrm{SubFrom}(a,k_1,b,k_2)) = 10$.

(32) $\mathrm{InsCode}(\mathrm{MultBy}(a,k_1,b,k_2)) = 11$.

(33) $\mathrm{InsCode}(\mathrm{Divide}(a,k_1,b,k_2)) = 12$.

(34) $\mathrm{InsCode}((a,k_1) := (b,k_2)) = 13$.

(35) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 0$ there exists k_1 such that $i_1 = \mathrm{goto}\, k_1$.

(36) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 1$ there exists a such that $i_1 = \mathrm{return}\, a$.

(37) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 2$ there exist a, k_1 such that $i_1 = a{:=}k_1$.

(38) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 3$ there exist a, k_1 such that $i_1 = \mathrm{saveIC}(a,k_1)$.

(39) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 4$ there exist a, k_1, k_2 such that $i_1 = (a,k_1) <> 0\_\mathrm{goto}\, k_2$.

(40) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 5$ there exist a, k_1, k_2 such that $i_1 = (a,k_1) <= 0\_\mathrm{goto}\, k_2$.

(41) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 6$ there exist a, k_1, k_2 such that $i_1 = (a,k_1) >= 0\_\mathrm{goto}\, k_2$.

(42) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 7$ there exist a, k_1, k_2 such that $i_1 = a_{k_1}{:=}k_2$.

(43) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 8$ there exist a, k_1, k_2 such that $i_1 = \mathrm{AddTo}(a,k_1,k_2)$.

(44) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 9$ there exist a, b, k_1, k_2 such that $i_1 = \mathrm{AddTo}(a,k_1,b,k_2)$.

(45) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 10$ there exist a, b, k_1, k_2 such that $i_1 = \mathrm{SubFrom}(a,k_1,b,k_2)$.

(46) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 11$ there exist a, b, k_1, k_2 such that $i_1 = \mathrm{MultBy}(a,k_1,b,k_2)$.

(47) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 12$ there exist a, b, k_1, k_2 such that $i_1 = \mathrm{Divide}(a,k_1,b,k_2)$.

(48) For every instruction i_1 of SCMPDS such that $\mathrm{InsCode}(i_1) = 13$ there exist a, b, k_1, k_2 such that $i_1 = (a,k_1) := (b,k_2)$.

(49) For every state s of SCMPDS and for every Int position d holds $d \in \mathrm{dom}\, s$.

(50) For every state s of SCMPDS holds $\text{Data-Loc}_{\mathrm{SCM}} \subseteq \mathrm{dom}\, s$.

(51) For every state s of SCMPDS holds $\mathrm{dom}(s \restriction \text{Data-Loc}_{\mathrm{SCM}}) = \text{Data-Loc}_{\mathrm{SCM}}$.

(52) For every Int position d_6 holds $d_6 \neq \mathbf{IC}_{\mathrm{SCMPDS}}$.

(53) For every instruction-location i_2 of SCMPDS and for every Int position d_6 holds $i_2 \neq d_6$.

(54) Let s_1, s_2 be states of SCMPDS. Suppose $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_2)}$ and for every Int position a holds $s_1(a) = s_2(a)$ and for every instruction-location i of SCMPDS holds $s_1(i) = s_2(i)$. Then $s_1 = s_2$.

Let l_1 be an instruction-location of SCMPDS. The functor $\mathrm{Next}(l_1)$ yields an instruction-location of SCMPDS and is defined as follows:

(Def. 19) There exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ such that $m_1 = l_1$ and $\mathrm{Next}(l_1) = \mathrm{Next}(m_1)$.

Next we state two propositions:

(55) For every instruction-location l_1 of SCMPDS and for every element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ such that $m_1 = l_1$ holds $\mathrm{Next}(m_1) = \mathrm{Next}(l_1)$.

(56) For every element i of SCMPDS-Instr such that $i = I$ and for every SCMPDS-State S such that $S = s$ holds $\mathrm{Exec}(I, s) = \text{Exec-Res}_{\text{SCM}}(i, S)$.

4. Execution Semantics of the SCMPDS instructions

The following propositions are true:

(57) $(\mathrm{Exec}(a{:=}k_1, s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(a{:=}k_1, s))(a) = k_1$ and for every b such that $b \neq a$ holds $(\mathrm{Exec}(a{:=}k_1, s))(b) = s(b)$.

(58) $(\mathrm{Exec}(a_{k_1}{:=}k_2, s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(a_{k_1}{:=}k_2, s))(\mathrm{DataLoc}(s(a), k_1)) = k_2$ and for every b such that $b \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(a_{k_1}{:=}k_2, s))(b) = s(b)$.

(59) $(\mathrm{Exec}((a, k_1) := (b, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}((a, k_1) := (b, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(b), k_2))$ and for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}((a, k_1) := (b, k_2), s))(c) = s(c)$.

(60) $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) + k_2$ and for every b such that $b \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, k_2), s))(b) = s(b)$.

(61) $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, b, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, b, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) + s(\mathrm{DataLoc}(s(b), k_2))$ and for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(\mathrm{AddTo}(a, k_1, b, k_2), s))(c) = s(c)$.

(62) $(\mathrm{Exec}(\mathrm{SubFrom}(a, k_1, b, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{SubFrom}(a, k_1, b, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) - s(\mathrm{DataLoc}(s(b), k_2))$ and for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(\mathrm{SubFrom}(a, k_1, b, k_2), s))(c) = s(c)$.

(63) $(\mathrm{Exec}(\mathrm{MultBy}(a, k_1, b, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{MultBy}(a, k_1, b, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) \cdot s(\mathrm{DataLoc}(s(b), k_2))$ and for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(\mathrm{MultBy}(a, k_1, b, k_2), s))(c) = s(c)$.

(64)(i) $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, b, k_2), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$,

(ii) if $\mathrm{DataLoc}(s(a), k_1) \neq \mathrm{DataLoc}(s(b), k_2)$, then $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, b, k_2), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) \div s(\mathrm{DataLoc}(s(b), k_2))$,

(iii) $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, b, k_2), s))(\mathrm{DataLoc}(s(b), k_2)) = s(\mathrm{DataLoc}(s(a), k_1)) \bmod s(\mathrm{DataLoc}(s(b), k_2))$, and

(iv) for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ and $c \neq \mathrm{DataLoc}(s(b), k_2)$ holds $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, b, k_2), s))(c) = s(c)$.

(65) $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, a, k_1), s))(\mathbf{IC}_{\text{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, a, k_1), s))(\mathrm{DataLoc}(s(a), k_1)) = s(\mathrm{DataLoc}(s(a), k_1)) \bmod s(\mathrm{DataLoc}(s(a), k_1))$ and for every c such that $c \neq \mathrm{DataLoc}(s(a), k_1)$ holds $(\mathrm{Exec}(\mathrm{Divide}(a, k_1, a, k_1), s))(c) = s(c)$.

Let s be a state of SCMPDS and let c be an integer. The functor ICplusConst(s,c) yielding an instruction-location of SCMPDS is defined as follows:

(Def. 20) There exists a natural number m such that $m = \mathbf{IC}_s$ and ICplusConst$(s,c) = |(m-2)+2\cdot c|+2$.

One can prove the following propositions:

(66) $(\mathrm{Exec}(\mathrm{goto}\ k_1,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{ICplusConst}(s,k_1)$ and for every a holds $(\mathrm{Exec}(\mathrm{goto}\ k_1,s))(a) = s(a)$.

(67) If $s(\mathrm{DataLoc}(s(a),k_1)) \neq 0$, then $(\mathrm{Exec}((a,k_1) <> 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{ICplusConst}(s,k_2)$ and if $s(\mathrm{DataLoc}(s(a),k_1)) = 0$, then $(\mathrm{Exec}((a,k_1) <> 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}((a,k_1) <> 0\_\mathrm{goto}\, k_2,s))(b) = s(b)$.

(68) If $s(\mathrm{DataLoc}(s(a),k_1)) \leq 0$, then $(\mathrm{Exec}((a,k_1) <= 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{ICplusConst}(s,k_2)$ and if $s(\mathrm{DataLoc}(s(a),k_1)) > 0$, then $(\mathrm{Exec}((a,k_1) <= 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}((a,k_1) <= 0\_\mathrm{goto}\, k_2,s))(b) = s(b)$.

(69) If $s(\mathrm{DataLoc}(s(a),k_1)) \geq 0$, then $(\mathrm{Exec}((a,k_1) >= 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{ICplusConst}(s,k_2)$ and if $s(\mathrm{DataLoc}(s(a),k_1)) < 0$, then $(\mathrm{Exec}((a,k_1) >= 0\_\mathrm{goto}\, k_2,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}((a,k_1) >= 0\_\mathrm{goto}\, k_2,s))(b) = s(b)$.

(70) $(\mathrm{Exec}(\mathrm{return}\, a,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = 2\cdot(|s(\mathrm{DataLoc}(s(a),\mathrm{RetIC}))| \div 2)+4$ and $(\mathrm{Exec}(\mathrm{return}\, a,s))(a) = s(\mathrm{DataLoc}(s(a),\mathrm{RetSP}))$ and for every b such that $a \neq b$ holds $(\mathrm{Exec}(\mathrm{return}\, a,s))(b) = s(b)$.

(71) $(\mathrm{Exec}(\mathrm{saveIC}(a,k_1),s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{saveIC}(a,k_1),s))(\mathrm{DataLoc}(s(a),k_1)) = \mathbf{IC}_s$ and for every b such that $\mathrm{DataLoc}(s(a),k_1) \neq b$ holds $(\mathrm{Exec}(\mathrm{saveIC}(a,k_1),s))(b) = s(b)$.

(72) For every integer k there exists a function f from Data-Loc$_{\mathrm{SCM}}$ into $\mathbb{Z}$ such that for every element x of Data-Loc$_{\mathrm{SCM}}$ holds $f(x) = k$.

(73) For every integer k there exists a state s of SCMPDS such that for every Int position d holds $s(d) = k$.

(74) Let k be an integer and l_1 be an instruction-location of SCMPDS. Then there exists a state s of SCMPDS such that $s(0) = l_1$ and for every Int position d holds $s(d) = k$.

(75) goto 0 is halting.

(76) For every instruction I of SCMPDS such that there exists s such that $(\mathrm{Exec}(I,s))(\mathbf{IC}_{\mathrm{SCMPDS}}) = \mathrm{Next}(\mathbf{IC}_s)$ holds I is non halting.

(77) a:=k_1 is non halting.

(78) a_{k_1}:=k_2 is non halting.

(79) $(a,k_1) := (b,k_2)$ is non halting.

(80) AddTo(a,k_1,k_2) is non halting.

(81) AddTo(a,k_1,b,k_2) is non halting.

(82) SubFrom(a,k_1,b,k_2) is non halting.

(83) MultBy(a,k_1,b,k_2) is non halting.

(84) Divide(a,k_1,b,k_2) is non halting.

(85) If $k_1 \neq 0$, then goto k_1 is non halting.

(86) $(a,k_1) <> 0\_\mathrm{goto}\, k_2$ is non halting.

(87) $(a,k_1) <= 0\_\mathrm{goto}\, k_2$ is non halting.

(88) $(a,k_1) >= 0\_\mathrm{goto}\,k_2$ is non halting.

(89) return a is non halting.

(90) $\mathrm{saveIC}(a,k_1)$ is non halting.

(91) Let I be a set. Then I is an instruction of SCMPDS if and only if one of the following conditions is satisfied:

there exists k_1 such that $I = \mathrm{goto}\ k_1$ or there exists a such that $I = \mathrm{return}\,a$ or there exist a, k_1 such that $I = \mathrm{saveIC}(a,k_1)$ or there exist a, k_1 such that $I = a{:=}k_1$ or there exist a, k_1, k_2 such that $I = a_{k_1}{:=}k_2$ or there exist a, k_1, k_2 such that $I = (a,k_1) <> 0\_\mathrm{goto}\,k_2$ or there exist a, k_1, k_2 such that $I = (a,k_1) <= 0\_\mathrm{goto}\,k_2$ or there exist a, k_1, k_2 such that $I = (a,k_1) >= 0\_\mathrm{goto}\,k_2$ or there exist a, b, k_1, k_2 such that $I = \mathrm{AddTo}(a,k_1,k_2)$ or there exist a, b, k_1, k_2 such that $I = \mathrm{AddTo}(a,k_1,b,k_2)$ or there exist a, b, k_1, k_2 such that $I = \mathrm{SubFrom}(a,k_1,b,k_2)$ or there exist a, b, k_1, k_2 such that $I = \mathrm{MultBy}(a,k_1,b,k_2)$ or there exist a, b, k_1, k_2 such that $I = \mathrm{Divide}(a,k_1,b,k_2)$ or there exist a, b, k_1, k_2 such that $I = (a,k_1) := (b,k_2)$.

Let us note that SCMPDS is halting.
The following propositions are true:

(92) For every instruction I of SCMPDS such that I is halting holds $I = \mathbf{halt}_{\mathrm{SCMPDS}}$.

(93) $\mathbf{halt}_{\mathrm{SCMPDS}} = \mathrm{goto}\ 0$.

(96)[3] For every state s of SCMPDS and for every instruction i of SCMPDS and for every instruction-location l of SCMPDS holds $(\mathrm{Exec}(i,s))(l) = s(l)$.

(97) SCMPDS is realistic.

One can verify that SCMPDS is steady-programmed and realistic.
The following propositions are true:

(98) $\mathbf{IC}_{\mathrm{SCMPDS}} \neq \mathbf{d}_i$ and $\mathbf{IC}_{\mathrm{SCMPDS}} \neq \mathbf{i}_i$.

(99) For every instruction I of SCMPDS such that $I = \mathrm{goto}\ 0$ holds I is halting.

ACKNOWLEDGMENTS

We wish to thank Prof. Y. Nakamura for many helpful suggestions.

REFERENCES

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/nat_1.html`.

[2] Grzegorz Bancerek. Sequences of ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/ordinal2.html`.

[3] Grzegorz Bancerek. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/card_3.html`.

[4] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/finseq_1.html`.

[5] Czesław Byliński. Functions and their basic properties. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funct_1.html`.

[6] Czesław Byliński. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funct_2.html`.

[7] Czesław Byliński. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/cqc_lang.html`.

[8] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/funct_4.html`.

[3] The propositions (94) and (95) have been removed.

[9] Jing-Chao Chen. A small computer model with push-down stack. *Journal of Formalized Mathematics*, 11, 1999. `http://mizar.org/JFM/Vol11/scmpds_1.html`.

[10] Agata Darmochwał. Finite sets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/finset_1.html`.

[11] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Journal of Formalized Mathematics*, 4, 1992. `http://mizar.org/JFM/Vol4/ami_1.html`.

[12] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Journal of Formalized Mathematics*, 4, 1992. `http://mizar.org/JFM/Vol4/ami_2.html`.

[13] Dariusz Surowik. Cyclic groups and some of their properties — part I. *Journal of Formalized Mathematics*, 3, 1991. `http://mizar.org/JFM/Vol3/gr_cy_1.html`.

[14] Yasushi Tanaka. On the decomposition of the states of SCM. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/ami_5.html`.

[15] Andrzej Trybulec. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/enumset1.html`.

[16] Andrzej Trybulec. Tarski Grothendieck set theory. *Journal of Formalized Mathematics*, Axiomatics, 1989. `http://mizar.org/JFM/Axiomatics/tarski.html`.

[17] Andrzej Trybulec. Tuples, projections and Cartesian products. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/mcart_1.html`.

[18] Andrzej Trybulec. Subsets of real numbers. *Journal of Formalized Mathematics*, Addenda, 2003. `http://mizar.org/JFM/Addenda/numbers.html`.

[19] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/ami_3.html`.

[20] Michał J. Trybulec. Integers. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/int_1.html`.

[21] Wojciech A. Trybulec. Groups. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/group_1.html`.

[22] Zinaida Trybulec. Properties of subsets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/subset_1.html`.

[23] Edmund Woronowicz. Relations and their basic properties. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/relat_1.html`.

Received June 15, 1999

Published January 2, 2004