Compiler Construction

Lecture 20: Code Generation III (Translation to Intermediate Code)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc10/

Winter semester 2010/11

Outline

Syntax of EPL

Definition (Syntax of EPL)

The syntax of EPL is defined as follows:

$$
\begin{array}{lrcl}
\mathbb{Z}: & z & & \text{(* } z \text{ is an integer *)} \\
\mathit{Ide}: & I & & \text{(* } I \text{ is an identifier *)} \\
\mathit{AExp}: & A & ::= & z \mid I \mid A_1 \; \texttt{+} \; A_2 \mid \ldots \\
\mathit{BExp}: & B & ::= & A_1 \; \texttt{<} \; A_2 \mid \texttt{not} \; B \mid B_1 \; \texttt{and} \; B_2 \mid B_1 \; \texttt{or} \; B_2 \\
\mathit{Cmd}: & C & ::= & I \; \texttt{:=} \; A \mid C_1 \texttt{;} C_2 \mid \texttt{if} \; B \; \texttt{then} \; C_1 \; \texttt{else} \; C_2 \mid \\
 & & & \texttt{while} \; B \; \texttt{do} \; C \mid I\texttt{()} \\
\mathit{Dcl}: & D & ::= & D_C \; D_V \; D_P \\
 & D_C & ::= & \varepsilon \mid \texttt{const} \; I_1 \; \texttt{:=} \; z_1 \texttt{,} \ldots \texttt{,} I_n \; \texttt{:=} \; z_n \texttt{;} \\
 & D_V & ::= & \varepsilon \mid \texttt{var} \; I_1 \texttt{,} \ldots \texttt{,} I_n \texttt{;} \\
 & D_P & ::= & \varepsilon \mid \texttt{proc} \; I_1 \texttt{;} K_1 \texttt{;} \ldots \texttt{;} I_n \texttt{;} K_n \texttt{;} \\
\mathit{Blk}: & K & ::= & D \; C \\
\mathit{Pgm}: & P & ::= & \texttt{in/out} \; I_1 \texttt{,} \ldots \texttt{,} I_n \texttt{;} K \texttt{.}
\end{array}
$$

The Abstract Machine AM

Definition (Abstract machine for EPL)

The abstract machine for EPL (AM) is defined by the state space

$$S := PC \times DS \times PS$$

with

- the program counter $PC := \mathbb{N}$,
- the data stack $DS := \mathbb{Z}^*$ (top of stack to the right), and
- the procedure stack (or: runtime stack) $PS := \mathbb{Z}^*$ (top of stack to the left).

Thus a state $s = (l, d, p) \in S$ is given by

- a program label $l \in PC$,
- a data stack $d = d.r : \ldots : d.1 \in DS$, and
- a procedure stack $p = p.1 : \ldots : p.t \in PS$.

AM Instructions

Definition (AM instructions)

The set of AM instructions is divided into

arithmetic instructions: `ADD`, `MULT`, ...

Boolean instructions: `NOT`, `AND`, `OR`, `LT`, ...

jumping instructions: `JMP(`ca`)`, `JFALSE(`ca`)` ($ca \in PC$)

procedure instructions: `CALL(`ca, dif, loc`)` ($ca \in PC$, $dif, loc \in \mathbb{N}$), `RET`

transfer instructions: `LOAD(`dif, off`)`, `STORE(`dif, off`)` ($dif, off \in \mathbb{N}$),
`LIT(`z`)` ($z \in \mathbb{Z}$)

Structure of Procedure Stack I

The semantics of procedure and transfer instructions requires a particular structure of the procedure stack $p \in PS$: is must be composed of frames (or: activation records) of the form

$$sl : dl : ra : v_1 : \ldots : v_k$$

where

static link sl: points to frame of surrounding declaration environment
$\Longrightarrow$ used to access non-local variables

dynamic link dl: points to previous frame (i.e., of calling procedure)
$\Longrightarrow$ used to remove topmost frame after termination of procedure call

return address ra: program label after termination of procedure call
$\Longrightarrow$ used to continue program execution after termination of procedure call

local variables v_i: values of locally declared variables

Structure of Procedure Stack II

Observation:

- The usage of a variable in a procedure body refers to its innermost declaration.
- If the level difference between the usage and the declaration is *dif*, then a chain of *dif* static links has to be followed to access the corresponding frame.

Structure of Procedure Stack II

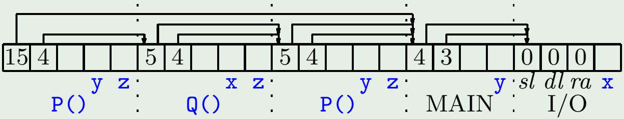
Observation:

- The usage of a variable in a procedure body refers to its innermost declaration.
- If the level difference between the usage and the declaration is *dif*, then a chain of *dif* static links has to be followed to access the corresponding frame.

Example (cf. Example 19.5)

```
in/out x;
  const c = 10;
  var y;
  proc P;
    var y, z;
    proc Q;
      var x, z;
      [... P() ...]
    [... x ... y ... Q() ...]
  proc R;
    [... P() ...]
  [... P() ...].
```

Procedure stack after second call of P:



Structure of Procedure Stack II

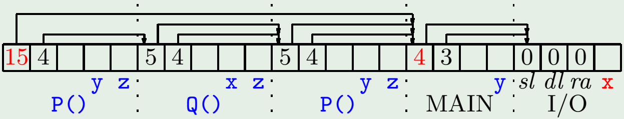
Observation:

- The usage of a variable in a procedure body refers to its innermost declaration.
- If the level difference between the usage and the declaration is *dif*, then a chain of *dif* static links has to be followed to access the corresponding frame.

Example (cf. Example 19.5)

```
in/out x;
  const c = 10;
  var y;
  proc P;
    var y, z;
    proc Q;
      var x, z;
      [... P() ...]
    [... x ... y ... Q() ...]
  proc R;
    [... P() ...]
  [... P() ...].
```

Procedure stack after second call of P:



P uses x $\Longrightarrow$ $dif = 2$

Structure of Procedure Stack II

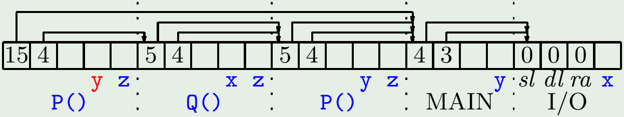
Observation:

- The usage of a variable in a procedure body refers to its innermost declaration.
- If the level difference between the usage and the declaration is *dif*, then a chain of *dif* static links has to be followed to access the corresponding frame.

Example (cf. Example 19.5)

```
in/out x;
  const c = 10;
  var y;
  proc P;
    var y, z;
    proc Q;
      var x, z;
      [... P() ...]
    [... x ... y ... Q() ...]
  proc R;
    [... P() ...]
  [... P() ...].
```

Procedure stack after second call of P:



P uses y $\Longrightarrow$ $\mathit{dif} = 0$

Definition (Semantics of AM programs)

An AM program is a sequence of $k \geq 1$ labeled AM instructions:

$$P = 1 : O_1; \ldots ; k : O_k$$

The set of all AM programs is denoted by AM.

The semantics of AM programs is determined by

$$[\![.]\!] : AM \times S \dashrightarrow S$$

with

$$[\![P]\!](l, d, p) := \begin{cases} [\![P]\!]([\![O_l]\!](l, d, p)) & \text{if } l \in [k] \\ (l, d, p) & \text{otherwise} \end{cases}$$

Outline

Translation of EPL into AM Programs

Goal: define translation mapping

$$\text{trans} : Pgm \dashrightarrow AM$$

Translation of EPL into AM Programs

Goal: define translation mapping

$$\text{trans} : Pgm \dashrightarrow AM$$

The translation employs a symbol table:

$$\begin{aligned} Tab := \{\text{st} \mid \text{st} : Ide \dashrightarrow & (\{\texttt{const}\} \times \mathbb{Z}) \\ & \cup (\{\texttt{var}\} \times Lev \times Off) \\ & \cup (\{\texttt{proc}\} \times PC \times Lev \times Size)\} \end{aligned}$$

whose entries are created by declarations:

- constant declarations: $(\texttt{const}, z)$
  - value $z \in \mathbb{Z}$
- variable declarations: $(\texttt{var}, lev, off)$
  - declaration level $lev \in Lev := \mathbb{N}$ ($0 \cong$ I/O, $1 \cong$ MAIN, ...)
  - offset $off \in Off := \mathbb{N}$
  - offset and difference between usage and declaration level determine procedure stack entry
- procedure declarations: $(\texttt{proc}, ca, lev, loc)$
  - code address $ca \in PC$
  - declaration level $lev \in Lev$
  - number of local variables $loc \in Size := \mathbb{N}$

Maintaining the Symbol Table

The symbol table is maintained by the function update(D, st, l) which specifies the update of symbol table st according to declaration D (with respect to current level l):

Maintaining the Symbol Table

The symbol table is maintained by the function $\mathrm{update}(D, \mathrm{st}, l)$ which specifies the update of symbol table st according to declaration D (with respect to current level l):

Definition 20.1 (update function)

$$\mathrm{update} : Dcl \times Tab \times Lev \dashrightarrow Tab$$

is defined by

$$\begin{aligned}
&\mathrm{update}(D_C\ D_V\ D_P, \mathrm{st}, l)\\
&\quad := \quad \mathrm{update}(D_P, \mathrm{update}(D_V, \mathrm{update}(D_C, \mathrm{st}, l), l), l)\\
&\qquad\quad\ \ \text{if all identifiers in } D_C\ D_V\ D_P \text{ different}\\
&\mathrm{update}(\varepsilon, \mathrm{st}, l)\\
&\quad := \quad \mathrm{st}\\
&\mathrm{update}(\texttt{const}\ I_1\ \texttt{:=}\ z_1\texttt{,}\ \ldots\texttt{,}I_n\ \texttt{:=}\ z_n\texttt{;}, \mathrm{st}, l)\\
&\quad := \quad \mathrm{st}[I_1 \mapsto (\texttt{const}, z_1), \ldots, I_n \mapsto (\texttt{const}, z_n)]\\
&\mathrm{update}(\texttt{var}\ I_1\texttt{,}\ \ldots\texttt{,}I_n\texttt{;}, \mathrm{st}, l)\\
&\quad := \quad \mathrm{st}[I_1 \mapsto (\texttt{var}, l, 1), \ldots, I_n \mapsto (\texttt{var}, l, n)]\\
&\mathrm{update}(\texttt{proc}\ I_1\texttt{;}K_1\texttt{;}\ \ldots\texttt{;}I_n\texttt{;}K_n\texttt{;}, \mathrm{st}, l)\\
&\quad := \quad \mathrm{st}[I_1 \mapsto (\texttt{proc}, a_1, l, \mathrm{size}(K_1)), \ldots, I_n \mapsto (\texttt{proc}, a_n, l, \mathrm{size}(K_n))]\\
&\qquad\quad\ \ \text{with “fresh” addresses } a_1, \ldots, a_n\\
&\qquad\quad\ \ \text{where } \mathrm{size}(D_C\ \texttt{var}\ I_1\texttt{,}\ \ldots\texttt{,}I_n\texttt{;}\ D_P C) := n
\end{aligned}$$

The Initial Symbol Table

An EPL program $P = \texttt{in/out}\ I_1\texttt{,}\ \ldots\ \texttt{,}I_n\texttt{;}K\texttt{.} \in Pgm$ has a semantics of type $\mathbb{Z}^n \dashrightarrow \mathbb{Z}^n$.

The Initial Symbol Table

An EPL program $P = \texttt{in/out}\ I_1\texttt{,}\ldots\texttt{,}I_n\texttt{;}K\texttt{.} \in Pgm$ has a semantics of type $\mathbb{Z}^n \dashrightarrow \mathbb{Z}^n$.

Given $(z_1, \ldots, z_n) \in \mathbb{Z}^n$, we choose the initial state

$$s := (1, \varepsilon, \underbrace{0 : 0 : 0 : z_1 : \ldots : z_n}_{\text{I/O frame}}) \in S = PC \times DS \times PS$$

The Initial Symbol Table

An EPL program $P = \texttt{in/out}\ I_1\texttt{,}\ \dots\ \texttt{,}I_n\texttt{;}K\texttt{.} \in Pgm$ has a semantics of type $\mathbb{Z}^n \dashrightarrow \mathbb{Z}^n$.

Given $(z_1, \dots, z_n) \in \mathbb{Z}^n$, we choose the initial state

$$s := (1, \varepsilon, \underbrace{0 : 0 : 0 : z_1 : \dots : z_n}_{\text{I/O frame}}) \in S = PC \times DS \times PS$$

Thus the corresponding initial symbol table has n entries:

$$\mathrm{st}_{I/O}(I_j) := (\texttt{var}, 0, j) \qquad \text{for every } j \in [n]$$

Outline

Translation of Programs

Translation of `in/out` I_1`,` $\ldots$ `,`I_n`;`D C`.`:

1. Create MAIN frame for executing C
2. Stop program execution after return

Translation of Programs

Translation of $\texttt{in/out}\ I_1\texttt{,}\ldots\texttt{,}I_n\texttt{;}D\ C\texttt{.}$:

1. Create MAIN frame for executing C
2. Stop program execution after return

Definition 20.2 (Translation of programs)

The mapping

$$\mathrm{trans} : \mathit{Pgm} \dashrightarrow \mathit{AM}$$

is defined by

$$\begin{aligned}\mathrm{trans}(\texttt{in/out}\ I_1\texttt{,}\ldots\texttt{,}I_n\texttt{;}K\ \texttt{.}) := {} & 1 : \texttt{CALL(}a\texttt{,0,}\mathrm{size}(K)\texttt{);} \\ & 2 : \texttt{JMP(0);} \\ & \mathrm{kt}(K, \mathrm{st}_{I/O}, a, 1)\end{aligned}$$

Outline

Translation of Blocks

Translation of $D\ C$:

1. Update symbol table according to D
2. Create code for procedures declared in D
   (using the updated symbol table – recursion!)
3. Create code for C (using the updated symbol table)

Translation of Blocks

Translation of $D\ C$:

1. Update symbol table according to D
2. Create code for procedures declared in D (using the updated symbol table – recursion!)
3. Create code for C (using the updated symbol table)

Definition 20.3 (Translation of blocks)

The mapping

$$\mathrm{kt} : \mathit{Blk} \times \mathit{Tab} \times \mathit{PC} \times \mathit{Lev} \dashrightarrow \mathit{AM}$$

("block translation") is defined by

$$\begin{aligned}\mathrm{kt}(D\ C, \mathrm{st}, a, l) := {} & \mathrm{dt}(D, \mathrm{update}(D, \mathrm{st}, l), l)\\ & \mathrm{ct}(C, \mathrm{update}(D, \mathrm{st}, l), a, l)\\ & a' : \texttt{RET;}\end{aligned}$$

Outline

Translation of Declarations

Translation of D:

- Generate code for the procedures declared in D

Translation of Declarations

Translation of D:

- Generate code for the procedures declared in D

Definition 20.4 (Translation of declarations)

The mapping

$$\mathrm{dt} : \mathit{Dcl} \times \mathit{Tab} \times \mathit{Lev} \dashrightarrow \mathit{AM}$$

("declaration translation") is defined by

$$\begin{aligned}
&\mathrm{dt}(D_C\ D_V\ D_P, \mathrm{st}, l)\\
&\quad := \quad \mathrm{dt}(D_P, \mathrm{st}, l)\\
&\mathrm{dt}(\varepsilon, \mathrm{st}, l)\\
&\quad := \quad \varepsilon\\
&\mathrm{dt}(\texttt{proc}\ I_1\,\texttt{;}K_1\,\texttt{;}\ \ldots\ \texttt{;}I_n\,\texttt{;}K_n\,\texttt{;}, \mathrm{st}, l)\\
&\quad := \quad \mathrm{kt}(K_1, \mathrm{st}, a_1, l+1)\\
&\qquad\qquad\quad \vdots\\
&\qquad\qquad \mathrm{kt}(K_n, \mathrm{st}, a_n, l+1)\\
&\qquad\qquad \text{where } \mathrm{st}(I_j) = (\texttt{proc}, a_j, \ldots, \ldots) \text{ for every } j \in [n]
\end{aligned}$$

Outline

Translation of Commands

Definition 20.5 (Translation of commands)

The mapping

$$\textrm{ct} : \mathit{Cmd} \times \mathit{Tab} \times \mathit{PC} \times \mathit{Lev} \dashrightarrow \mathit{AM}$$

("command translation") is defined by

$$\begin{aligned}
\textrm{ct}(I\ \texttt{:=}\ A, \textrm{st}, a, l) &:= \textrm{at}(A, \textrm{st}, a, l) \\
&\quad\ a' : \texttt{STORE(}l - \mathit{lev}\texttt{,}\mathit{off}\texttt{);} \\
&\quad\ \text{if } \textrm{st}(I) = (\texttt{var}, \mathit{lev}, \mathit{off}) \\
\textrm{ct}(I\texttt{()}, \textrm{st}, a, l) &:= a : \texttt{CALL(}\mathit{ca}\texttt{,}l - \mathit{lev}\texttt{,}\mathit{loc}\texttt{);} \\
&\quad\ \text{if } \textrm{st}(I) = (\texttt{proc}, \mathit{ca}, \mathit{lev}, \mathit{loc}) \\
\textrm{ct}(C_1\texttt{;}C_2, \textrm{st}, a, l) &:= \textrm{ct}(C_1, \textrm{st}, a, l) \\
&\quad\ \textrm{ct}(C_2, \textrm{st}, a', l) \\
\textrm{ct}(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, \textrm{st}, a, l) &:= \textrm{bt}(B, \textrm{st}, a, l) \\
&\quad\ a' : \texttt{JFALSE(}a''\texttt{);} \\
&\quad\ \textrm{ct}(C_1, \textrm{st}, a'+1, l) \\
&\quad\ a''-1 : \texttt{JMP(}a'''\texttt{);} \\
&\quad\ \textrm{ct}(C_2, \textrm{st}, a'', l) \\
&\quad\ a''' : \\
\textrm{ct}(\texttt{while}\ B\ \texttt{do}\ C, \textrm{st}, a, l) &:= \textrm{bt}(B, \textrm{st}, a, l) \\
&\quad\ a' : \texttt{JFALSE(}a''+1\texttt{);} \\
&\quad\ \textrm{ct}(C, \textrm{st}, a'+1, l) \\
&\quad\ a'' : \texttt{JMP(}a\texttt{);}
\end{aligned}$$

Outline

Definition 20.6 (Translation of Boolean expressions)

The mapping

$$\mathrm{bt} : \mathit{BExp} \times \mathit{Tab} \times \mathit{PC} \times \mathit{Lev} \dashrightarrow \mathit{AM}$$

("Boolean expression translation") is defined by

$$\begin{array}{rcl}
\mathrm{bt}(A_1 \texttt{<} A_2, \mathrm{st}, a, l) & := & \mathrm{at}(A_1, \mathrm{st}, a, l) \\
& & \mathrm{at}(A_2, \mathrm{st}, a', l) \\
& & a'' : \texttt{LT;} \\
\mathrm{bt}(\texttt{not}\ B, \mathrm{st}, a, l) & := & \mathrm{bt}(B, \mathrm{st}, a, l) \\
& & a' : \texttt{NOT;} \\
\mathrm{bt}(B_1\ \texttt{and}\ B_2, \mathrm{st}, a, l) & := & \mathrm{bt}(B_1, \mathrm{st}, a, l) \\
& & \mathrm{bt}(B_2, \mathrm{st}, a', l) \\
& & a'' : \texttt{AND;} \\
\mathrm{bt}(B_1\ \texttt{or}\ B_2, \mathrm{st}, a, l) & := & \mathrm{bt}(B_1, \mathrm{st}, a, l) \\
& & \mathrm{bt}(B_2, \mathrm{st}, a', l) \\
& & a'' : \texttt{OR;}
\end{array}$$

Definition 20.7 (Translation of arithmetic expressions)

The mapping

$$\mathrm{at} : \mathit{AExp} \times \mathit{Tab} \times \mathit{PC} \times \mathit{Lev} \dashrightarrow \mathit{AM}$$

("arithmetic expression translation") is defined by

$$\begin{aligned}
\mathrm{at}(z, \mathrm{st}, a, l) &:= a : \texttt{LIT(}z\texttt{);} \\
\mathrm{at}(I, \mathrm{st}, a, l) &:= \begin{cases} a : \texttt{LIT(}z\texttt{);} & \text{if } \mathrm{st}(I) = (\texttt{const}, z) \\ a : \texttt{LOAD(}l - \mathit{lev}\texttt{,}\mathit{off}\texttt{);} & \text{if } \mathrm{st}(I) = (\texttt{var}, \mathit{lev}, \mathit{off}) \end{cases} \\
\mathrm{at}(A_1 \texttt{+} A_2, \mathrm{st}, a, l) &:= \mathrm{at}(A_1, \mathrm{st}, a, l) \\
& \mathrm{at}(A_2, \mathrm{st}, a', l) \\
& a'' : \texttt{ADD;}
\end{aligned}$$

Outline

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

Intermediate code:

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{trans}(\texttt{in/out } I_1\texttt{,}\ldots\texttt{,}I_n\texttt{;}K\texttt{.}) :=$

$1 : \texttt{CALL(}a\texttt{,0,}\mathrm{size}(K)\texttt{);}$

$2 : \texttt{JMP(0);}$

$\mathrm{kt}(K, \mathrm{st}_{I/O}, a, 1)$

$\mathrm{st}_{I/O} = [\texttt{x} \mapsto (\texttt{var}, 0, 1)]$

Intermediate code:

$\mathrm{trans}(\texttt{in/out x;}K\texttt{.})$

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{kt}(D\ C, \mathrm{st}, a, l) :=$

$\quad \mathrm{dt}(D, \mathrm{update}(D, \mathrm{st}, l), l)$

$\quad \mathrm{ct}(C, \mathrm{update}(D, \mathrm{st}, l), a, l)$

$\quad a' :$ `RET;`

$\mathrm{st}_{I/O} = [\mathtt{x} \mapsto (\mathtt{var}, 0, 1)]$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
$\quad \mathrm{kt}(K, \mathrm{st}_{I/O}, a_0, 1)$

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\text{update}(\texttt{var } I_1, \ldots, I_n;, \text{st}, l) :=$
$\quad \text{st}[I_1 \mapsto (\texttt{var}, l, 1), \ldots, I_n \mapsto (\texttt{var}, l, n)]$
$\text{update}(\texttt{proc } I_1; K_1; \ldots; I_n; K_n;, \text{st}, l) :=$
$\quad \text{st}[I_1 \mapsto (\texttt{proc}, a_1, l, \text{size}(K_1)), \ldots, I_n \mapsto (\texttt{proc}, a_n, l, \text{size}(K_n))]$

$$\text{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1), \texttt{y} \mapsto (\texttt{var}, 1, 1), \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$$

Intermediate code:

$1:$ `CALL(`a_0`,0,1);`
$2:$ `JMP(0);`
$\quad \text{dt}(D, \text{update}(D, \text{st}_{I/O}, 1), 1)$
$\quad \text{ct}(C, \text{update}(D, \text{st}_{I/O}, 1), a_0, 1)$
$a_2:$ `RET;`

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

Intermediate code:

$1:$ `CALL(`a_0`,0,1);`
$2:$ `JMP(0);`
$\mathrm{dt}(D, \mathrm{st}', 1)$
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
$a_2:$ `RET;`

$\mathrm{dt}(\texttt{proc}\ I_1; K_1; \ldots; I_n; K_n;, \mathrm{st}, l) :=$

$$\mathrm{kt}(K_1, \mathrm{st}, a_1, l+1)$$
$$\vdots$$
$$\mathrm{kt}(K_n, \mathrm{st}, a_n, l+1)$$

where $\mathrm{st}(I_j) = (\texttt{proc}, a_j, \ldots, \ldots)$ for every $j \in [n]$

$$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1), \; \texttt{y} \mapsto (\texttt{var}, 1, 1), \; \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$$

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{kt}(D\ C, \mathrm{st}, a, l) :=$

$\mathrm{dt}(D, \mathrm{update}(D, \mathrm{st}, l), l)$
$\mathrm{ct}(C, \mathrm{update}(D, \mathrm{st}, l), a, l)$
$a' :$ `RET;`

$\mathrm{st}' = [\mathtt{x} \mapsto (\mathtt{var}, 0, 1),$
$\mathtt{y} \mapsto (\mathtt{var}, 1, 1),$
$\mathtt{F} \mapsto (\mathtt{proc}, a_1, 1, 0)]$

Intermediate code:

$1 :$ `CALL(`a_0`,0,1);`
$2 :$ `JMP(0);`
$\mathrm{kt}(K_{\mathtt{F}}, \mathrm{st}', a_1, 2)$
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
$a_2 :$ `RET;`

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{ct}(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, \mathrm{st}, a, l) :=$

$\quad \mathrm{bt}(B, \mathrm{st}, a, l)$
$\quad a' : \texttt{JFALSE(}a''\texttt{);}$
$\quad \mathrm{ct}(C_1, \mathrm{st}, a'+1, l)$
$\quad a''-1 : \texttt{JMP(}a'''\texttt{);}$
$\quad \mathrm{ct}(C_2, \mathrm{st}, a'', l)$
$\quad a''' :$

$$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1), \texttt{y} \mapsto (\texttt{var}, 1, 1), \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$$

Intermediate code:

$1 : \texttt{CALL(}a_0\texttt{,0,1);}$
$2 : \texttt{JMP(0);}$
$\quad \mathrm{ct}(C_{\texttt{F}}, \mathrm{st}', a_1, 2)$
$a_3 : \texttt{RET;}$
$\quad \mathrm{ct}(C, \mathrm{st}', a_0, 1)$
$a_2 : \texttt{RET;}$

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$$\mathrm{bt}(A_1 \texttt{ > } A_2, \mathrm{st}, a, l) := \begin{array}{l} \mathrm{at}(A_1, \mathrm{st}, a, l) \\ \mathrm{at}(A_2, \mathrm{st}, a', l) \\ a'' : \texttt{GT;} \end{array}$$

$$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1), \; \texttt{y} \mapsto (\texttt{var}, 1, 1), \; \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$$

Intermediate code:

$$\begin{array}{rl} 1: & \texttt{CALL(}a_0\texttt{,0,1);} \\ 2: & \texttt{JMP(0);} \\ & \mathrm{bt}(\texttt{x > 1}, \mathrm{st}', a_1, 2) \\ a_4: & \texttt{JFALSE(}a_3\texttt{);} \\ & \mathrm{ct}(C_1, \mathrm{st}', a_4 + 1, 2) \\ a_3: & \texttt{RET;} \\ & \mathrm{ct}(C, \mathrm{st}', a_0, 1) \\ a_2: & \texttt{RET;} \end{array}$$

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\text{at}(I, \text{st}, a, l) :=$

$$\begin{cases} a : \texttt{LIT(}z\texttt{);} & \text{if } \text{st}(I) = (\texttt{const}, z) \\ a : \texttt{LOAD(}l - \mathit{lev}\texttt{,}\mathit{off}\texttt{);} & \text{if } \text{st}(I) = (\texttt{var}, \mathit{lev}, \mathit{off}) \end{cases}$$

$$\begin{aligned} \text{st}' = [&\texttt{x} \mapsto (\texttt{var}, 0, 1), \\ &\texttt{y} \mapsto (\texttt{var}, 1, 1), \\ &\texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)] \end{aligned}$$

Intermediate code:

$$\begin{aligned} 1 &: \texttt{CALL(}a_0\texttt{,0,1);} \\ 2 &: \texttt{JMP(0);} \\ &\quad \text{at}(\texttt{x}, \text{st}', a_1, 2) \\ &\quad \text{at}(\texttt{1}, \text{st}', a', 2) \\ &\quad \texttt{GT;} \\ a_4 &: \texttt{JFALSE(}a_3\texttt{);} \\ &\quad \text{ct}(C_1, \text{st}', a_4 + 1, 2) \\ a_3 &: \texttt{RET;} \\ &\quad \text{ct}(C, \text{st}', a_0, 1) \\ a_2 &: \texttt{RET;} \end{aligned}$$

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{at}(z, \mathrm{st}, a, l) := a : \texttt{LIT(}z\texttt{);}$

$$\begin{aligned}\mathrm{st}' = [&\texttt{x} \mapsto (\texttt{var}, 0, 1),\\ &\texttt{y} \mapsto (\texttt{var}, 1, 1),\\ &\texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]\end{aligned}$$

Intermediate code:

$$\begin{aligned}1 &: \texttt{CALL(}a_0\texttt{,0,1);}\\ 2 &: \texttt{JMP(0);}\\ a_1 &: \texttt{LOAD(2,1);}\\ & \ \mathrm{at}(1, \mathrm{st}', a', 2)\\ & \ \texttt{GT;}\\ a_4 &: \texttt{JFALSE(}a_3\texttt{);}\\ & \ \mathrm{ct}(C_1, \mathrm{st}', a_4 + 1, 2)\\ a_3 &: \texttt{RET;}\\ & \ \mathrm{ct}(C, \mathrm{st}', a_0, 1)\\ a_2 &: \texttt{RET;}\end{aligned}$$

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{ct}(I \texttt{ := } A, \mathrm{st}, a, l) :=$
$\quad \mathrm{at}(A, \mathrm{st}, a, l)$
$\quad a' : \texttt{STORE(}l - \mathit{lev}\texttt{,}\mathit{off}\texttt{);}$
if $\mathrm{st}(I) = (\texttt{var}, \mathit{lev}, \mathit{off})$

$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1),$
$\quad \texttt{y} \mapsto (\texttt{var}, 1, 1),$
$\quad \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
$\mathrm{ct}(C_1, \mathrm{st}', a_4 + 1, 2)$
a_3 : `RET;`
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
a_2 : `RET;`

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$$\mathrm{at}(A_1 \texttt{ + } A_2, \mathrm{st}, a, l) := \begin{array}{l} \mathrm{at}(A_1, \mathrm{st}, a, l) \\ \mathrm{at}(A_2, \mathrm{st}, a', l) \\ a'' : \texttt{ADD;} \end{array}$$

$$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1), \\ \texttt{y} \mapsto (\texttt{var}, 1, 1), \\ \texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
$\mathrm{at}(\texttt{y * x}, \mathrm{st}', a_4 + 1, 2)$
`STORE(1,1);`
$\mathrm{at}(\texttt{x - 1}, \mathrm{st}', a', 2)$
`STORE(2,1);`
$\mathrm{ct}(\texttt{F()}, \mathrm{st}', a'', 2)$
a_3 : `RET;`
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
a_2 : `RET;`

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\mathrm{ct}(I(), \mathrm{st}, a, l) :=$

$\quad a : \texttt{CALL(}ca\texttt{,}l - lev\texttt{,}loc\texttt{);}$

if $\mathrm{st}(I) = (\texttt{proc}, ca, lev, loc)$

$\mathrm{st}' = [\texttt{x} \mapsto (\texttt{var}, 0, 1),$
$\texttt{y} \mapsto (\texttt{var}, 1, 1),$
$\texttt{F} \mapsto (\texttt{proc}, a_1, 1, 0)]$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
`LOAD(1,1);`
`LOAD(2,1);`
`MULT;`
`STORE(1,1);`
`LOAD(2,1);`
`LIT(1);`
`SUB;`
`STORE(2,1);`
$\mathrm{ct}(\texttt{F()}, \mathrm{st}', a'', 2)$
a_3 : `RET;`
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
a_2 : `RET;`

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$$\begin{aligned}\mathrm{st}' = [&\mathtt{x} \mapsto (\mathtt{var}, 0, 1),\\ &\mathtt{y} \mapsto (\mathtt{var}, 1, 1),\\ &\mathtt{F} \mapsto (\mathtt{proc}, a_1, 1, 0)]\end{aligned}$$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
`LOAD(1,1);`
`LOAD(2,1);`
`MULT;`
`STORE(1,1);`
`LOAD(2,1);`
`LIT(1);`
`SUB;`
`STORE(2,1);`
`CALL(`a_1`,1,0);`
a_3 : `RET;`
$\mathrm{ct}(C, \mathrm{st}', a_0, 1)$
a_2 : `RET;`

Example: Factorial Function I

Example 20.8 (Factorial function (cf. Ex. 19.1))

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$$\mathrm{st}' = [\mathtt{x} \mapsto (\mathtt{var}, 0, 1), \mathtt{y} \mapsto (\mathtt{var}, 1, 1), \mathtt{F} \mapsto (\mathtt{proc}, a_1, 1, 0)]$$

Intermediate code:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
`LOAD(1,1);`
`LOAD(2,1);`
`MULT;`
`STORE(1,1);`
`LOAD(2,1);`
`LIT(1);`
`SUB;`
`STORE(2,1);`
`CALL(`a_1`,1,0);`
a_3 : `RET;`
a_0 : `LIT(1);`
`STORE(0,1);`
`CALL(`a_1`,0,0);`
`LOAD(0,1);`
`STORE(1,1);`
a_2 : `RET;`

Example: Factorial Function II

Example 20.8 (Factorial function; continued)

Code with symbolic addresses:

1 : `CALL(`a_0`,0,1);`
2 : `JMP(0);`
a_1 : `LOAD(2,1);`
`LIT(1);`
`GT;`
a_4 : `JFALSE(`a_3`);`
`LOAD(1,1);`
`LOAD(2,1);`
`MULT;`
`STORE(1,1);`
`LOAD(2,1);`
`LIT(1);`
`SUB;`
`STORE(2,1);`
`CALL(`a_1`,1,0);`
a_3 : `RET;`
a_0 : `LIT(1);`
`STORE(0,1);`
`CALL(`a_1`,0,0);`
`LOAD(0,1);`
`STORE(1,1);`
a_2 : `RET;`

Linearized
($a_0 = 17, a_1 = 3, a_2 = 22, a_3 = 16, a_4 = 6$):

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
1 : CALL(17,0,1);
2 : JMP(0);
3 : LOAD(2,1);
4 : LIT(1);
5 : GT;
6 : JFALSE(16);
7 : LOAD(1,1);
8 : LOAD(2,1);
9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$\underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\texttt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$
18	1	$\underline{4:3:2:0}:\underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
1 : CALL(17,0,1);
2 : JMP(0);
3 : LOAD(2,1);
4 : LIT(1);
5 : GT;
6 : JFALSE(16);
7 : LOAD(1,1);
8 : LOAD(2,1);
9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` = 2:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\texttt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$
18	1	$\underline{4:3:2:0}:\underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1}:\underline{0:0:0:2}$
3	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
4	2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` = 2:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$
18	1	$\underline{4:3:2:0}:\underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1}:\underline{0:0:0:2}$
3	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
4	2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
6	1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
7	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for `x` = 2:

```
1 : CALL(17,0,1);
2 : JMP(0);
3 : LOAD(2,1);
4 : LIT(1);
5 : GT;
6 : JFALSE(16);
7 : LOAD(1,1);
8 : LOAD(2,1);
9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$
18	1	$\underline{4:3:2:0}:\underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1}:\underline{0:0:0:2}$
3	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
4	2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
5	2 : 1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
6	1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
7	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
8	1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
9	1 : 2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
1 : CALL(17,0,1);
2 : JMP(0);
3 : LOAD(2,1);
4 : LIT(1);
5 : GT;
6 : JFALSE(16);
7 : LOAD(1,1);
8 : LOAD(2,1);
9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for x = 2:

```
1 : CALL(17,0,1);
2 : JMP(0);
3 : LOAD(2,1);
4 : LIT(1);
5 : GT;
6 : JFALSE(16);
7 : LOAD(1,1);
8 : LOAD(2,1);
9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0}:\underline{0:0:0:2}$
18	1	$\underline{4:3:2:0}:\underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1}:\underline{0:0:0:2}$
3	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
4	2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
6	1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
7	ε	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
8	1	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
10	2	$\underline{3:2:20}:\underline{4:3:2:1}:\underline{0:0:0:2}$
11	ε	$\underline{3:2:20}:\underline{4:3:2:2}:\underline{0:0:0:2}$
12	2	$\underline{3:2:20}:\underline{4:3:2:2}:\underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20}:\underline{4:3:2:2}:\underline{0:0:0:2}$
14	1	$\underline{3:2:20}:\underline{4:3:2:2}:\underline{0:0:0:2}$

Example 20.8 (Factorial function; continued)

Computation for x = 2:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$0:0:0:2$
17	ε	$4:3:2:0 : 0:0:0:2$
18	1	$4:3:2:0 : 0:0:0:2$
19	ε	$4:3:2:1 : 0:0:0:2$
3	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
4	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
5	$2:1$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
6	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
7	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
8	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
9	$1:2$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
10	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
11	ε	$3:2:20 : 4:3:2:2 : 0:0:0:2$
12	2	$3:2:20 : 4:3:2:2 : 0:0:0:2$
13	$2:1$	$3:2:20 : 4:3:2:2 : 0:0:0:2$
14	1	$3:2:20 : 4:3:2:2 : 0:0:0:2$
15	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$

Example 20.8 (Factorial function; continued)

Computation for `x` = 2:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$0:0:0:2$
17	ε	$4:3:2:0 : 0:0:0:2$
18	1	$4:3:2:0 : 0:0:0:2$
19	ε	$4:3:2:1 : 0:0:0:2$
3	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
4	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
5	$2:1$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
6	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
7	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
8	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
9	$1:2$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
10	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
11	ε	$3:2:20 : 4:3:2:2 : 0:0:0:2$
12	2	$3:2:20 : 4:3:2:2 : 0:0:0:2$
13	$2:1$	$3:2:20 : 4:3:2:2 : 0:0:0:2$
14	1	$3:2:20 : 4:3:2:2 : 0:0:0:2$
15	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$
3	ε	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
4	1	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for `x` $= 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
20	ε	$\underline{4:3:2:2} : \underline{0:0:0:1}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\texttt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
20	ε	$\underline{4:3:2:2} : \underline{0:0:0:1}$
21	2	$\underline{4:3:2:2} : \underline{0:0:0:1}$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\texttt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	DS	PS
1	ε	$0:0:0:2$
17	ε	$4:3:2:0 : 0:0:0:2$
18	1	$4:3:2:0 : 0:0:0:2$
19	ε	$4:3:2:1 : 0:0:0:2$
3	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
4	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
5	$2:1$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
6	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
7	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
8	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
9	$1:2$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
10	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
11	ε	$3:2:20 : 4:3:2:2 : 0:0:0:2$
12	2	$3:2:20 : 4:3:2:2 : 0:0:0:2$
13	$2:1$	$3:2:20 : 4:3:2:2 : 0:0:0:2$
14	1	$3:2:20 : 4:3:2:2 : 0:0:0:2$
15	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$
3	ε	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
4	1	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
5	$1:1$	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
6	0	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
16	ε	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
16	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$
20	ε	$4:3:2:2 : 0:0:0:1$
21	2	$4:3:2:2 : 0:0:0:1$
22	ε	$4:3:2:2 : 0:0:0:2$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$0:0:0:2$
17	ε	$4:3:2:0 : 0:0:0:2$
18	1	$4:3:2:0 : 0:0:0:2$
19	ε	$4:3:2:1 : 0:0:0:2$
3	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
4	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
5	$2:1$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
6	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
7	ε	$3:2:20 : 4:3:2:1 : 0:0:0:2$
8	1	$3:2:20 : 4:3:2:1 : 0:0:0:2$
9	$1:2$	$3:2:20 : 4:3:2:1 : 0:0:0:2$
10	2	$3:2:20 : 4:3:2:1 : 0:0:0:2$
11	ε	$3:2:20 : 4:3:2:2 : 0:0:0:2$
12	2	$3:2:20 : 4:3:2:2 : 0:0:0:2$
13	$2:1$	$3:2:20 : 4:3:2:2 : 0:0:0:2$
14	1	$3:2:20 : 4:3:2:2 : 0:0:0:2$
15	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$
3	ε	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
4	1	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
5	$1:1$	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
6	0	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
16	ε	$6:2:16 : 3:2:20 : 4:3:2:2 : 0:0:0:1$
16	ε	$3:2:20 : 4:3:2:2 : 0:0:0:1$
20	ε	$4:3:2:2 : 0:0:0:1$
21	2	$4:3:2:2 : 0:0:0:1$
22	ε	$4:3:2:2 : 0:0:0:2$
2	ε	$0:0:0:2$

Example: Factorial Function III

Example 20.8 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	$\underline{0:0:0:2}$
17	ε	$\underline{4:3:2:0} : \underline{0:0:0:2}$
18	1	$\underline{4:3:2:0} : \underline{0:0:0:2}$
19	ε	$\underline{4:3:2:1} : \underline{0:0:0:2}$
3	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
4	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
5	$2:1$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
6	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
7	ε	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
8	1	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
9	$1:2$	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
10	2	$\underline{3:2:20} : \underline{4:3:2:1} : \underline{0:0:0:2}$
11	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
12	2	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
13	$2:1$	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
14	1	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:2}$
15	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
3	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
4	1	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
5	$1:1$	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
6	0	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{6:2:16} : \underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
16	ε	$\underline{3:2:20} : \underline{4:3:2:2} : \underline{0:0:0:1}$
20	ε	$\underline{4:3:2:2} : \underline{0:0:0:1}$
21	2	$\underline{4:3:2:2} : \underline{0:0:0:1}$
22	ε	$\underline{4:3:2:2} : \underline{0:0:0:2}$
2	ε	$\underline{0:0:0:2}$
0	ε	$\underline{0:0:0:2}$

Outline

Correctness of the Translation

Theorem 20.9 (Correctness of translation)

For every $P \in Pgm$, $n \in \mathbb{N}$, and $(z_1, \ldots, z_n), (z'_1, \ldots, z'_n) \in \mathbb{Z}^n$:

$$\begin{array}{rl} & \mathfrak{M}[\![P]\!](z_1, \ldots, z_n) = (z'_1, \ldots, z'_n) \\ \iff & [\![\mathrm{trans}(P)]\!](1, \varepsilon, 0 : 0 : 0 : z_1 : \ldots : z_n) = (0, \varepsilon, 0 : 0 : 0 : z'_1 : \ldots : z'_n) \end{array}$$

Theorem 20.9 (Correctness of translation)

For every $P \in Pgm$, $n \in \mathbb{N}$, and $(z_1, \ldots, z_n), (z'_1, \ldots, z'_n) \in \mathbb{Z}^n$:

$$\begin{array}{ll} & \mathfrak{M}[\![P]\!](z_1, \ldots, z_n) = (z'_1, \ldots, z'_n) \\ \iff & [\![\mathrm{trans}(P)]\!](1, \varepsilon, 0 : 0 : 0 : z_1 : \ldots : z_n) = (0, \varepsilon, 0 : 0 : 0 : z'_1 : \ldots : z'_n) \end{array}$$

Proof.

see M. Mohnen: *A Compiler Correctness Proof for the Static Link Technique by means of Evolving Algebras*, Fundamenta Informaticae 29(3), 1997, pp. 257–303 □

Outline

Lectures after Chistmas Break

1. Error handling in (top-down) parsing
2. Strong noncircularity of attribute grammars
3. More about code generation
   - Static & dynamic data structures
   - Compiler backend (register allocation, ...)
4. Code analysis & optimization