Universal register machine, U

High-level specification

Universal RM U carries out the following computation, starting with $\mathtt{R}_0 = 0$, $\mathtt{R}_1 = e$ (code of a program), $\mathtt{R}_2 = a$ (code of a list of arguments) and all other registers zeroed:

- decode e as a RM program P
- decode a as a list of register values $a_1, \ldots, a_n$
- carry out the computation of the RM program P starting with $\mathtt{R}_0 = 0, \mathtt{R}_1 = a_1, \ldots, \mathtt{R}_n = a_n$ (and any other registers occurring in P set to 0).

Mnemonics for the registers of U and the role they play in its program:

$R_1 \equiv P$ code of the RM to be simulated

$R_2 \equiv A$ code of current register contents of simulated RM

$R_3 \equiv PC$ program counter—number of the current instruction (counting from 0)

$R_4 \equiv N$ code of the current instruction body

$R_5 \equiv C$ type of the current instruction body

$R_6 \equiv R$ current value of the register to be incremented or decremented by current instruction (if not HALT)

$R_7 \equiv S$, $R_8 \equiv T$ and $R_9 \equiv Z$ are auxiliary registers.

Overall structure of U's program

$\boxed{1}$ copy $\mathtt{PC}$th item of list in $\mathtt{P}$ to $\mathtt{N}$; goto $\boxed{2}$

$\boxed{2}$ if $\mathtt{N} = 0$ then copy 0th item of list in $\mathtt{A}$ to $\mathtt{R}_0$ and halt, else (decode $\mathtt{N}$ as $\langle\!\langle y, z \rangle\!\rangle$; $\mathtt{C} ::= y$; $\mathtt{N} ::= z$; goto $\boxed{3}$)

{at this point either $\mathtt{C} = 2i$ is even and current instruction is $\mathtt{R}_i^+ \to \mathtt{L}_z$,
or $\mathtt{C} = 2i + 1$ is odd and current instruction is $\mathtt{R}_i^- \to \mathtt{L}_j, \mathtt{L}_k$ where $z = \langle j, k \rangle$}

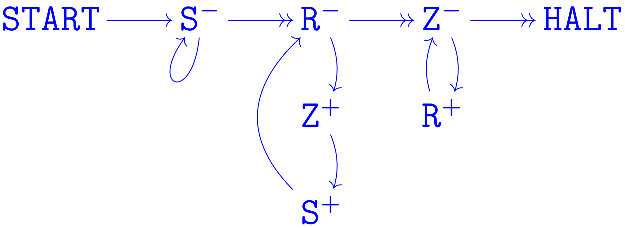
$\boxed{3}$ copy ith item of list in $\mathtt{A}$ to $\mathtt{R}$; goto $\boxed{4}$

$\boxed{4}$ execute current instruction on $\mathtt{R}$; update $\mathtt{PC}$ to next label; restore register values to $\mathtt{A}$; goto $\boxed{1}$

To implement this, we need RMs for manipulating (codes of) lists of numbers. . .

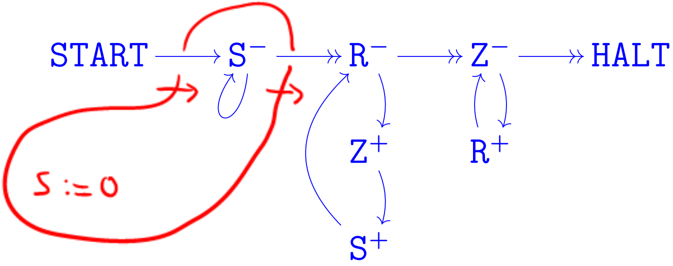
The program $\mathtt{START} \longrightarrow \boxed{\mathtt{S} ::= \mathtt{R}} \longrightarrow \mathtt{HALT}$

to copy the contents of $\mathtt{R}$ to $\mathtt{S}$ can be implemented by



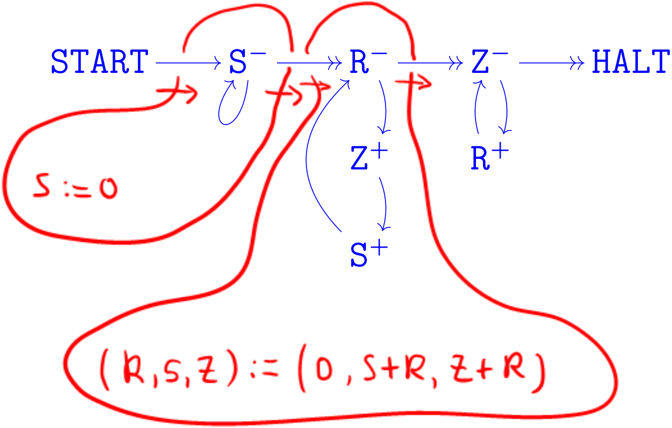
The program $\texttt{START} \longrightarrow \boxed{\texttt{S} ::= \texttt{R}} \longrightarrow \texttt{HALT}$

to copy the contents of $\texttt{R}$ to $\texttt{S}$ can be implemented by



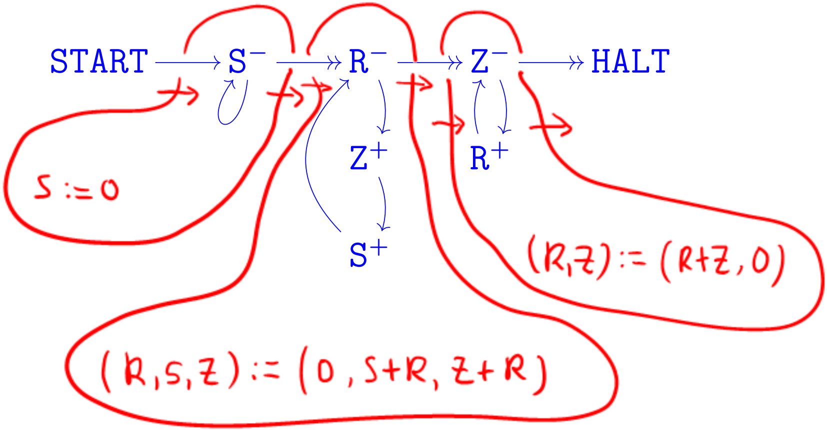
The program $\texttt{START} \rightarrow \boxed{\texttt{S ::= R}} \rightarrow \texttt{HALT}$

to copy the contents of $\texttt{R}$ to $\texttt{S}$ can be implemented by



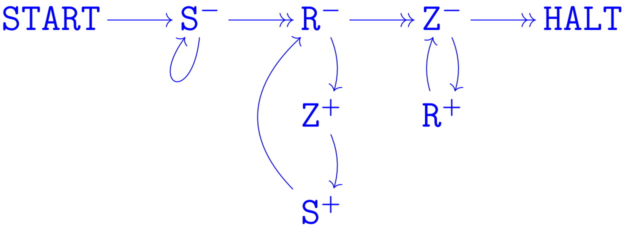
The program $\texttt{START} \to \boxed{\texttt{S ::= R}} \to \texttt{HALT}$

to copy the contents of $\texttt{R}$ to $\texttt{S}$ can be implemented by



The program $\mathtt{START} \longrightarrow \boxed{\mathtt{S} ::= \mathtt{R}} \longrightarrow \mathtt{HALT}$

to copy the contents of $\mathtt{R}$ to $\mathtt{S}$ can be implemented by

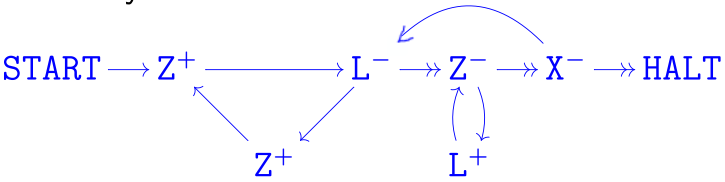


precondition:
$\mathtt{R} = x$
$\mathtt{S} = y$
$\mathtt{Z} = 0$

postcondition:
$\mathtt{R} = x$
$\mathtt{S} = x$
$\mathtt{Z} = 0$

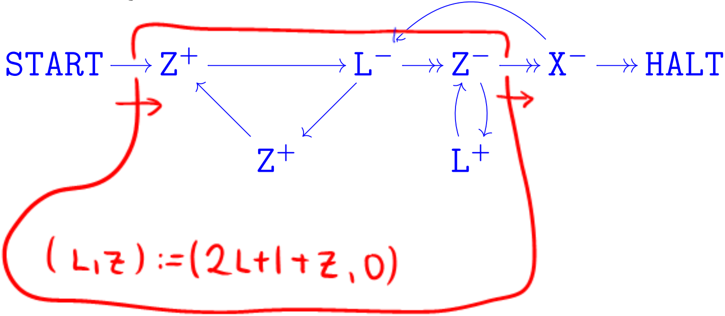
The program $\texttt{START} \rightarrow \boxed{\begin{matrix}\textit{push } \texttt{X} \\ \textit{to } \texttt{L}\end{matrix}} \rightarrow \texttt{HALT}$ $\quad \boxed{2^{\texttt{X}}(2\texttt{L}+1)}$

to carry out the assignment $(\texttt{X},\texttt{L}) ::= (0, \texttt{X} :: \texttt{L})$ can be implemented by



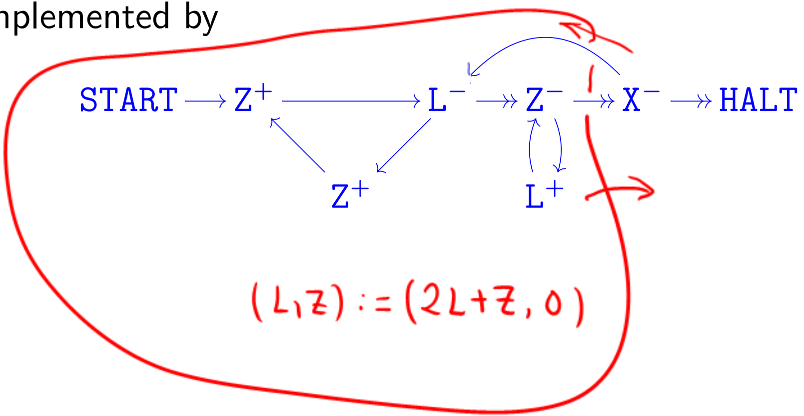
The program $\texttt{START} \longrightarrow \boxed{\textit{push } \texttt{X} \textit{ to } \texttt{L}} \longrightarrow \texttt{HALT}$

to carry out the assignment $(\texttt{X}, \texttt{L}) ::= (\mathbf{0}, \texttt{X} :: \texttt{L})$ can be implemented by



The program $\mathtt{START} \longrightarrow \boxed{\textit{push } \mathtt{X} \textit{ to } \mathtt{L}} \longrightarrow \mathtt{HALT}$

to carry out the assignment $(\mathtt{X}, \mathtt{L}) ::= (\mathbf{0}, \mathtt{X} :: \mathtt{L})$ can be implemented by



The program $\mathtt{START} \longrightarrow \boxed{\begin{array}{c}\textit{push } \mathtt{X} \\ \textit{to } \mathtt{L}\end{array}} \longrightarrow \mathtt{HALT}$

to carry out the assignment $(\mathtt{X}, \mathtt{L}) ::= (0, \mathtt{X} :: \mathtt{L})$ can be implemented by

$$\begin{tikzcd}
\mathtt{START} \arrow[r] & \mathtt{Z}^+ \arrow[rr] & & \mathtt{L}^- \arrow[r, two heads] \arrow[dl] & \mathtt{Z}^- \arrow[r, two heads] \arrow[d, bend left] & \mathtt{X}^- \arrow[r, two heads] \arrow[ll, bend right] & \mathtt{HALT} \\
& & \mathtt{Z}^+ \arrow[ul] & & \mathtt{L}^+ \arrow[u, bend left] & &
\end{tikzcd}$$

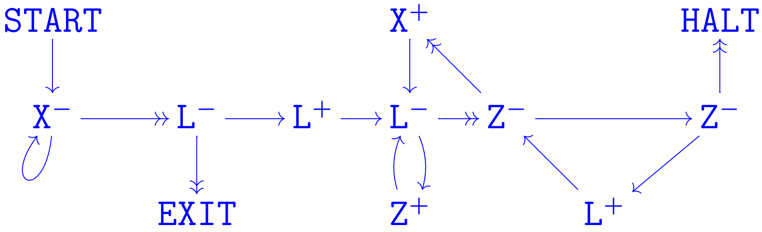
precondition:
$\mathtt{X} = x$
$\mathtt{L} = \ell$
$\mathtt{Z} = 0$

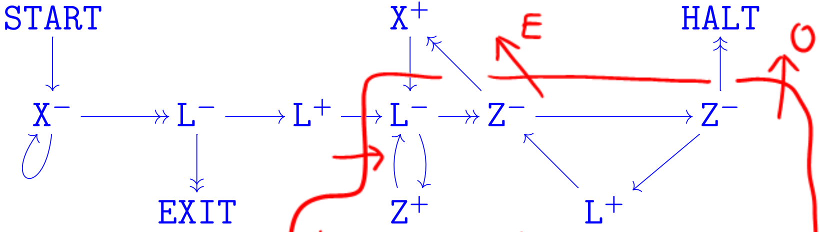
postcondition:
$\mathtt{X} = 0$
$\mathtt{L} = \langle\!\langle x, \ell \rangle\!\rangle = 2^x(2\ell + 1)$
$\mathtt{Z} = 0$

The program $\mathtt{START}\longrightarrow\boxed{\begin{matrix}\textit{pop}\ \mathtt{L}\\ \textit{to}\ \mathtt{X}\end{matrix}}\begin{matrix}\longrightarrow\mathtt{HALT}\\ \twoheadrightarrow\mathtt{EXIT}\end{matrix}$ specified by

"$\textit{if}\ \mathtt{L} = \mathbf{0}\ \textit{then}\ (\mathtt{X} ::= \mathbf{0};\ \textit{goto}\ \mathtt{EXIT})\ \textit{else}$
$\textit{let}\ \mathtt{L} = \langle\!\langle x, \ell \rangle\!\rangle\ \textit{in}\ (\mathtt{X} ::= x;\ \mathtt{L} ::= \ell;\ \textit{goto}\ \mathtt{HALT})$"

can be implemented by





if $Z+L$ even then

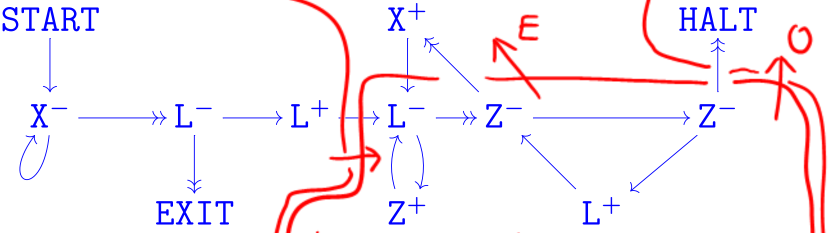
$(Z,L) := (0, \frac{1}{2}(Z+L))$ & goto E

else

$(Z,L) := (0, \frac{1}{2}(Z+L-1))$ & goto O

$\{$assuming $Z=0$ & $L>0\}$

(While L even do $L := \frac{1}{2}L$; $X := X+1$);
$L := \frac{1}{2}(L-1)$



if $Z+L$ even then
$(Z,L) := (0, \frac{1}{2}(Z+L))$ & goto E
else
$(Z,L) := (0, \frac{1}{2}(Z+L-1))$ & goto O

The program $\mathtt{START} \longrightarrow \boxed{\begin{array}{c} \textit{\textbf{pop}}\ \mathtt{L} \\ \textit{\textbf{to}}\ \mathtt{X} \end{array}} \begin{array}{l} \longrightarrow \mathtt{HALT} \\ \twoheadrightarrow \mathtt{EXIT} \end{array}$ specified by

"$\textit{\textbf{if}}\ \mathtt{L} = \mathbf{0}\ \textit{\textbf{then}}\ (\mathtt{X} ::= \mathbf{0};\ \textit{\textbf{goto}}\ \mathtt{EXIT})\ \textit{\textbf{else}}$
$\textit{\textbf{let}}\ \mathtt{L} = \langle\!\langle x, \ell \rangle\!\rangle\ \textit{\textbf{in}}\ (\mathtt{X} ::= x;\ \mathtt{L} ::= \ell;\ \textit{\textbf{goto}}\ \mathtt{HALT})$"

can be implemented by

$$\begin{tikzcd}
\mathtt{START} \arrow[d] & & & \mathtt{X}^+ \arrow[d] & & & \mathtt{HALT} \\
\mathtt{X}^- \arrow[loop below] \arrow[r, two heads] & \mathtt{L}^- \arrow[r] \arrow[d, two heads] & \mathtt{L}^+ \arrow[r] & \mathtt{L}^- \arrow[r, two heads] \arrow[d, bend left] & \mathtt{Z}^- \arrow[rr] \arrow[ul] & & \mathtt{Z}^- \arrow[u, two heads] \arrow[dl] \\
& \mathtt{EXIT} & & \mathtt{Z}^+ \arrow[u, bend left] & & \mathtt{L}^+ \arrow[ul] &
\end{tikzcd}$$

Overall structure of U's program

$\boxed{1}$ copy $\mathtt{PC}$th item of list in $\mathtt{P}$ to $\mathtt{N}$; goto $\boxed{2}$

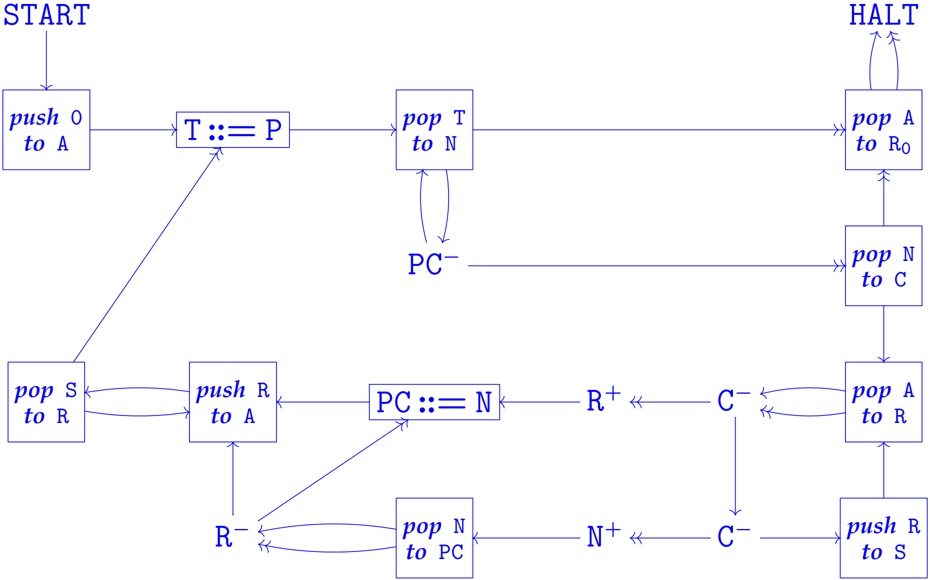
$\boxed{2}$ if $\mathtt{N} = 0$ then copy 0th item of list in $\mathtt{A}$ to $\mathtt{R}_0$ and halt, else (decode $\mathtt{N}$ as $\langle\!\langle y, z \rangle\!\rangle$; $\mathtt{C} ::= y$; $\mathtt{N} ::= z$; goto $\boxed{3}$)

{at this point either $\mathtt{C} = 2i$ is even and current instruction is $\mathtt{R}_i^+ \to \mathtt{L}_z$, or $\mathtt{C} = 2i + 1$ is odd and current instruction is $\mathtt{R}_i^- \to \mathtt{L}_j, \mathtt{L}_k$ where $z = \langle j, k \rangle$}

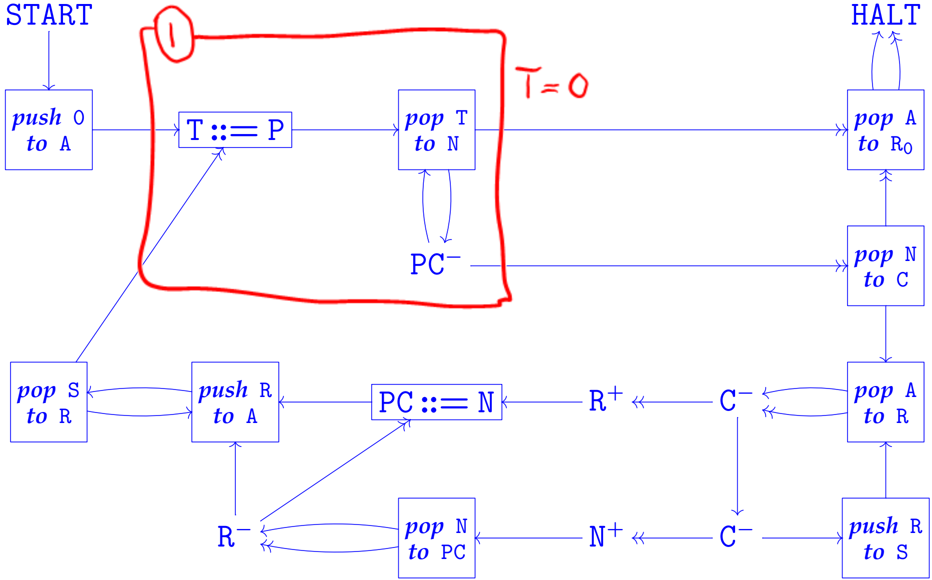
$\boxed{3}$ copy ith item of list in $\mathtt{A}$ to $\mathtt{R}$; goto $\boxed{4}$

$\boxed{4}$ execute current instruction on $\mathtt{R}$; update $\mathtt{PC}$ to next label; restore register values to $\mathtt{A}$; goto $\boxed{1}$

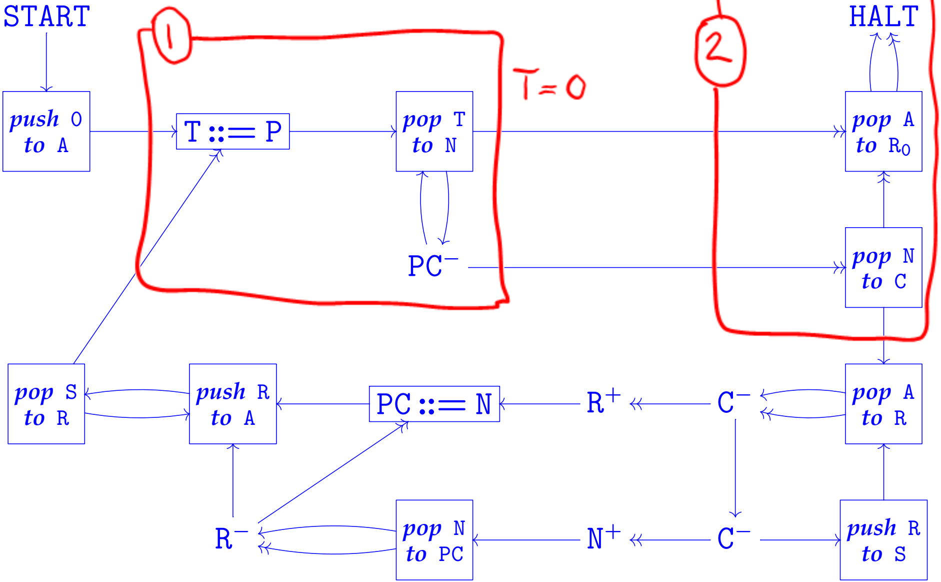
The program for U



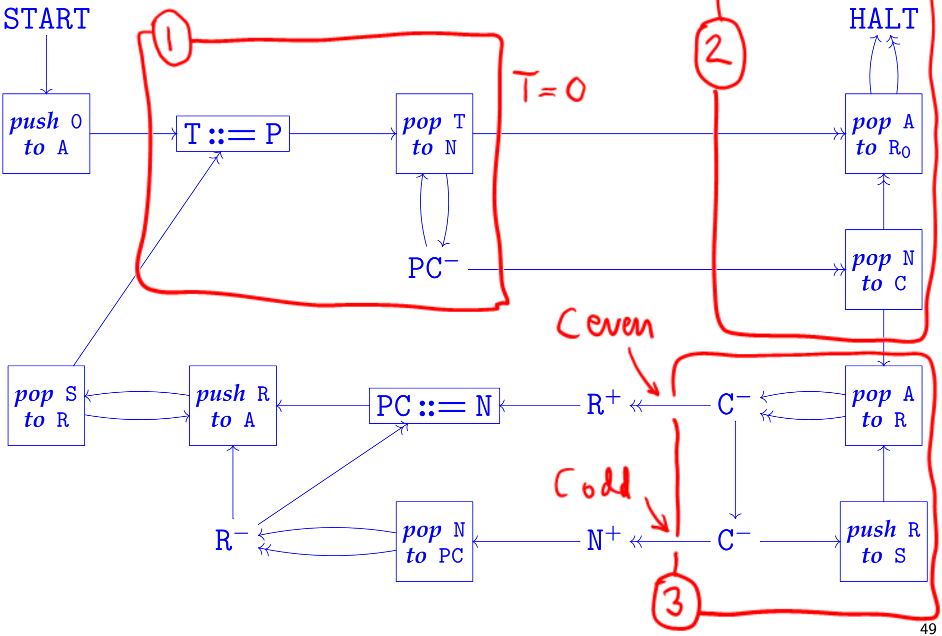
The program for U



The program for U



The program for U



The program for U