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

[1] copy $\mathtt{PC}$th item of list in $\mathtt{P}$ to $\mathtt{N}$ (halting if $\mathtt{PC} >$ length of list); goto [2]

[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 [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 $z = \langle j, k \rangle$}

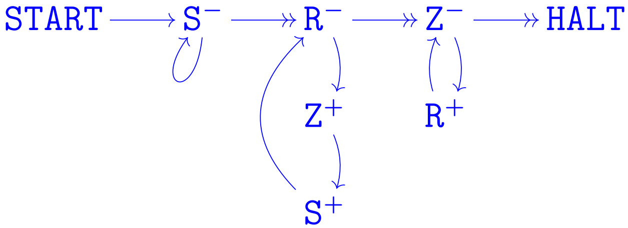
[3] copy ith item of list in $\mathtt{A}$ to $\mathtt{R}$; goto [4]

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

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

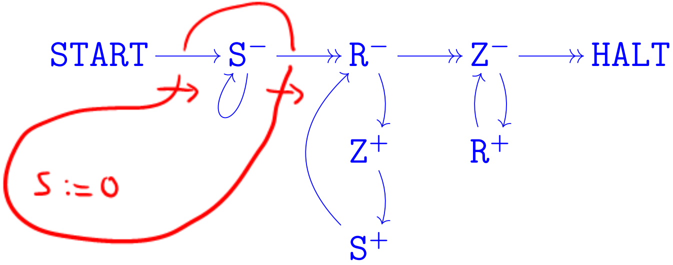
The program START⟶ $\boxed{\texttt{S} ::= \texttt{R}}$ ⟶HALT

to copy the contents of R to S can be implemented by



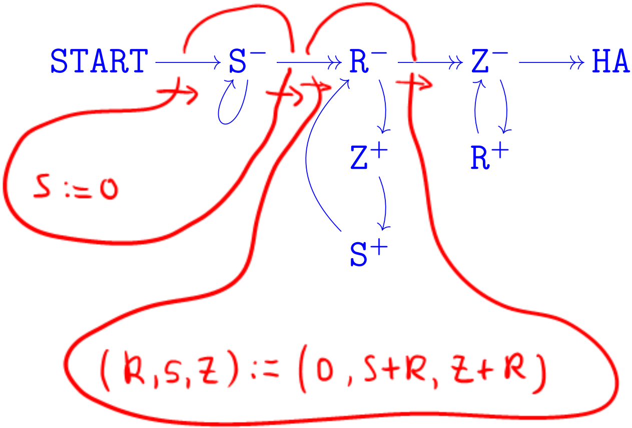
The program START$\longrightarrow$ $\boxed{\texttt{S ::= R}}$ $\longrightarrow$HALT

to copy the contents of R to S can be implemented by



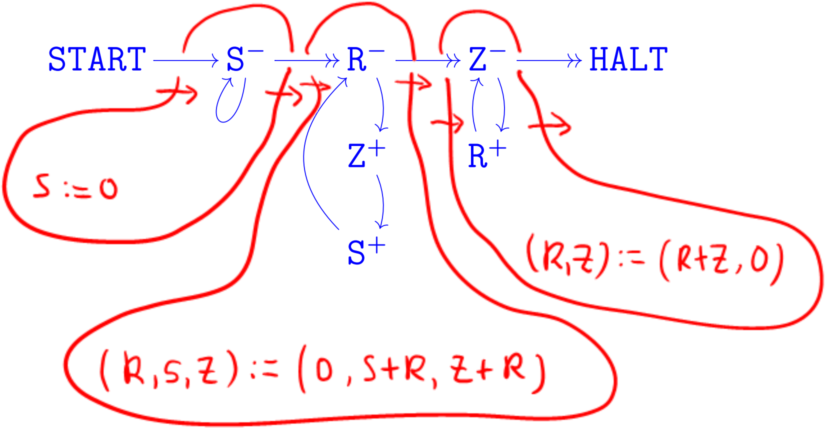
The program $\mathtt{START} \rightarrow \boxed{\mathtt{S ::= R}} \rightarrow \mathtt{HALT}$

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



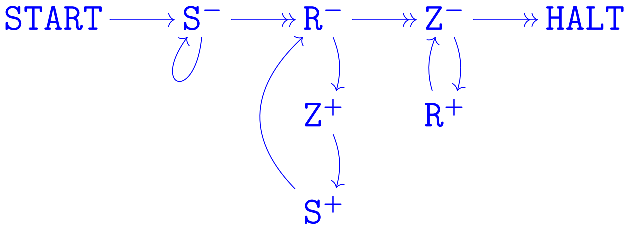
The program START⟶ S ::= R ⟶HALT

to copy the contents of R to S can be implemented by



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

to copy the contents of R to S can be implemented by



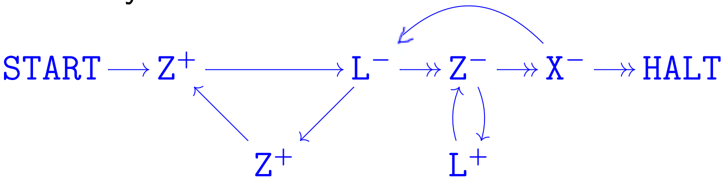
precondition:
R = $\boldsymbol{x}$
S = $\boldsymbol{y}$
Z = $\mathbf{0}$

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

The program START $\longrightarrow$ [*push* X *to* L] $\longrightarrow$ HALT

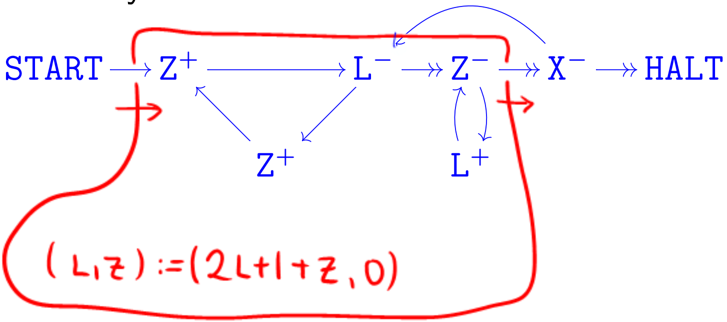
$2^{\mathtt{X}}(2\mathtt{L}+1)$

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



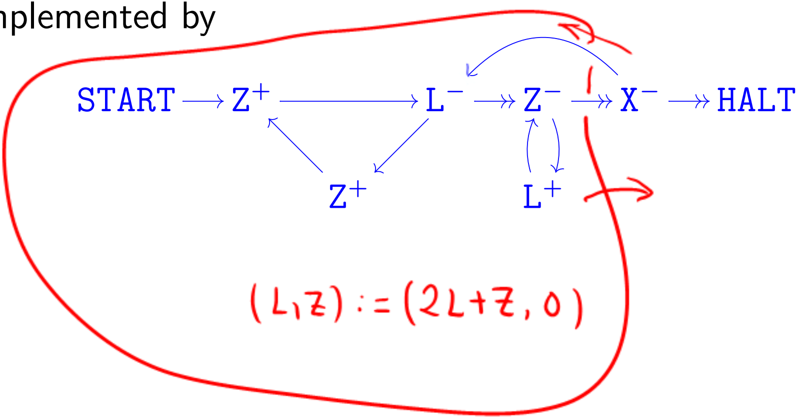
The program START → *push* X *to* L → HALT

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



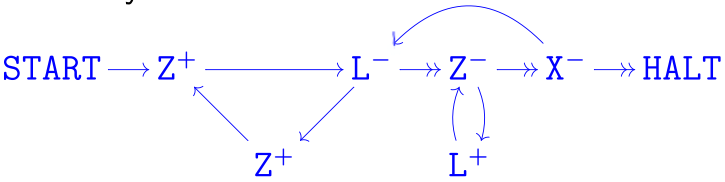
The program $\texttt{START} \longrightarrow \boxed{\textit{push}\ \texttt{X}\ \textit{to}\ \texttt{L}} \longrightarrow \texttt{HALT}$

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



The program START $\longrightarrow$ [*push* X *to* L] $\longrightarrow$ HALT

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



precondition:

$\mathtt{X} = x$

$\mathtt{L} = \ell$

$\mathtt{Z} = 0$

postcondition:

$\mathtt{X} = 0$

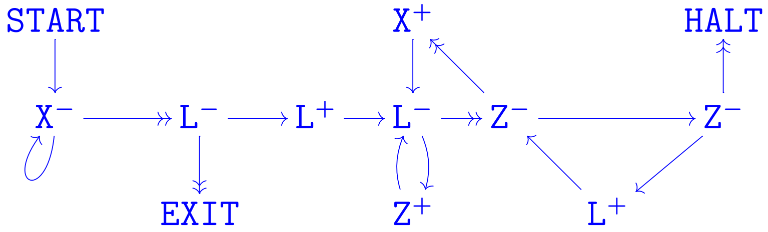
$\mathtt{L} = \langle\langle x, \ell \rangle\rangle = 2^x(2\ell + 1)$

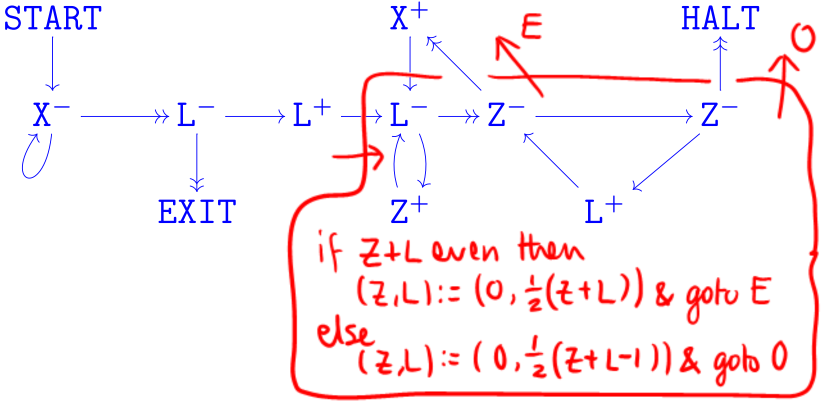
$\mathtt{Z} = 0$

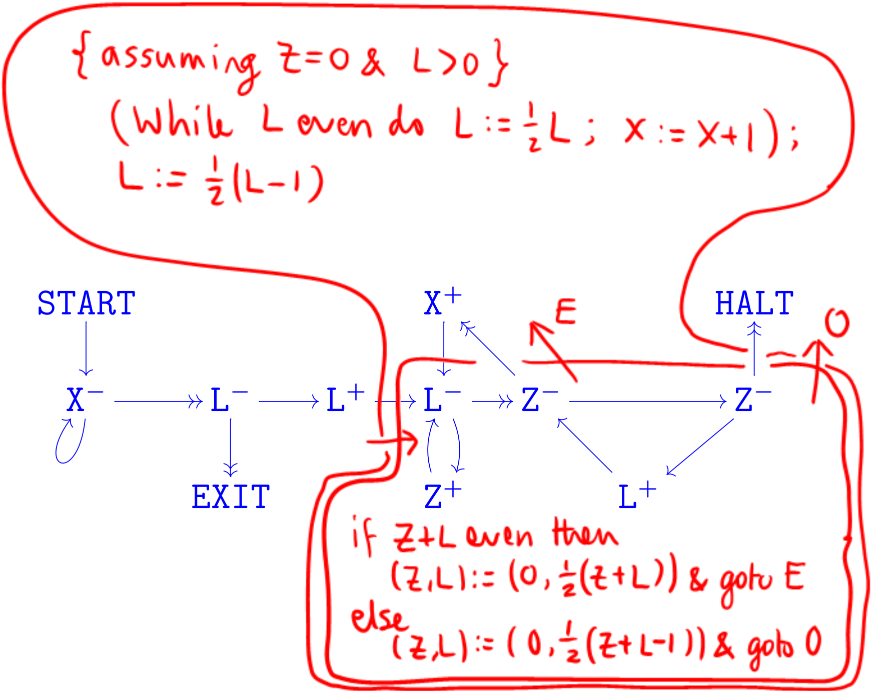
The program $\texttt{START} \longrightarrow \boxed{\textit{pop}\ \texttt{L}\ \textit{to}\ \texttt{X}} \begin{array}{l} \longrightarrow \texttt{HALT} \\ \twoheadrightarrow \texttt{EXIT} \end{array}$ specified by

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

can be implemented by



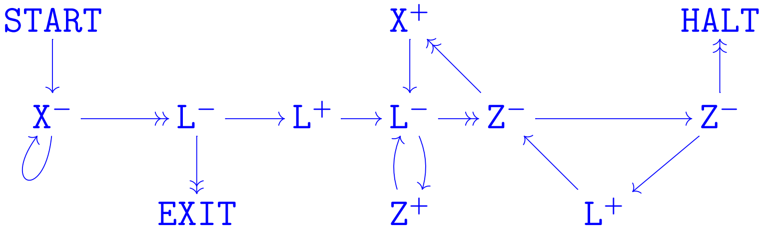

START
X⁻
L⁻
EXIT
L⁺
X⁺
L⁻
Z⁺
Z⁻
E
HALT
Z⁻
O
L⁺
if Z+L even then
(Z,L) := (0, ½(Z+L)) & goto E
else
(Z,L) := (0, ½(Z+L−1)) & goto O


{assuming Z=0 & L>0}
(While L even do L := ½L ; X := X+1);
L := ½(L−1)
START
X⁻
L⁻
L⁺
EXIT
X⁺
L⁻
Z⁺
Z⁻
E
HALT
O
Z⁻
L⁺
if Z+L even then
(Z,L) := (0, ½(Z+L)) & goto E
else
(Z,L) := (0, ½(Z+L−1)) & goto O

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

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

can be implemented by



Overall structure of U's program

1 copy PCth item of list in P to N (halting if PC $>$ length of list); goto 2

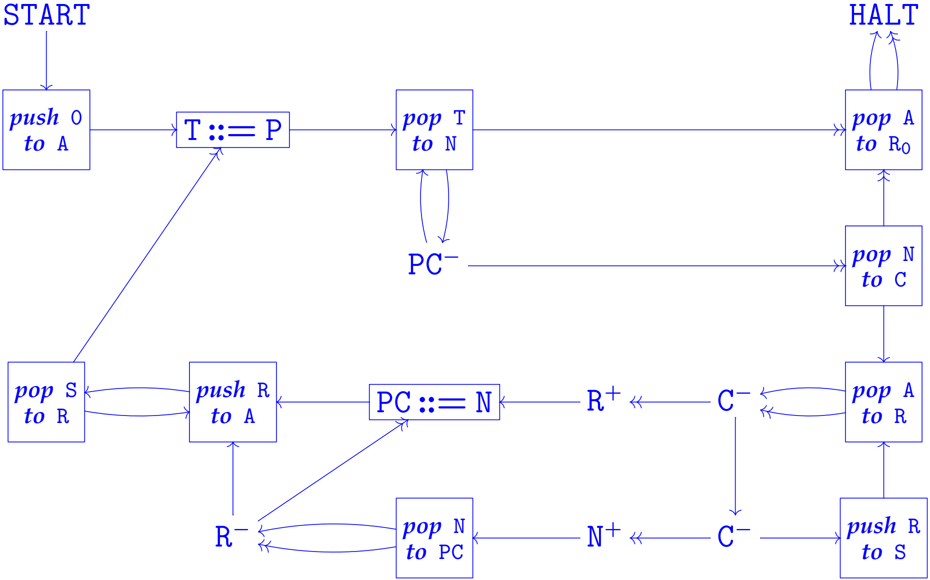
2 if N $= 0$ then copy 0th item of list in A to R_0 and halt, else (decode N as $\langle\langle y, z\rangle\rangle$; C $::= y$; N $::= z$; goto 3)

{at this point either C $= 2i$ is even and current instruction is $R_i^+ \to L_z$,
or C $= 2i+1$ is odd and current instruction is $R_i^- \to L_j, L_k$ where $z = \langle j, k\rangle$}

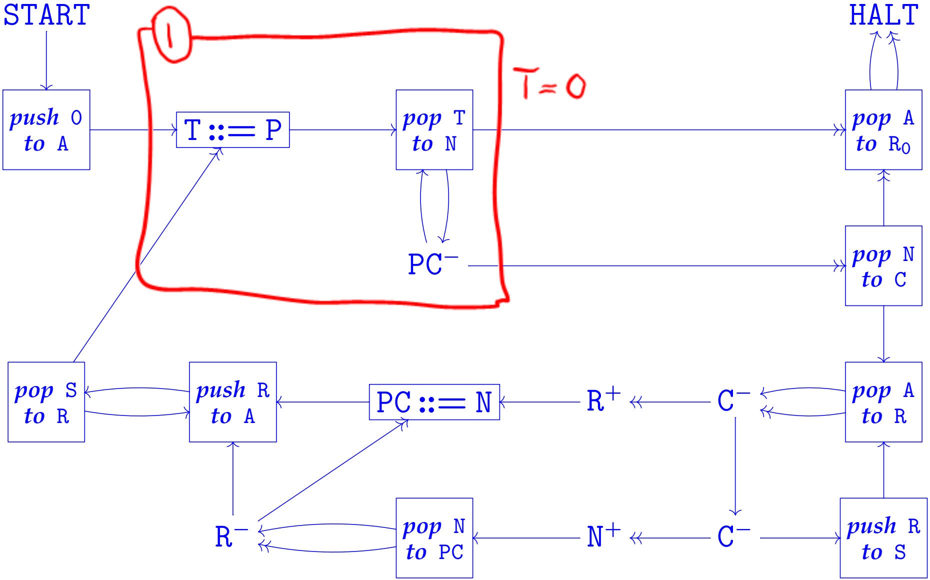
3 copy ith item of list in A to R; goto 4

4 execute current instruction on R; update PC to next label; restore register values to A; goto 1

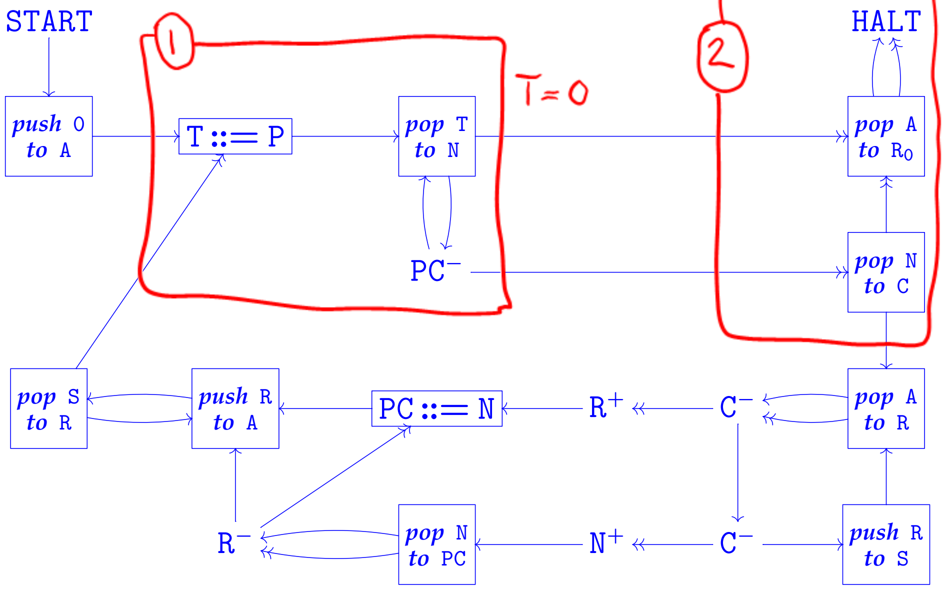
The program for *U*



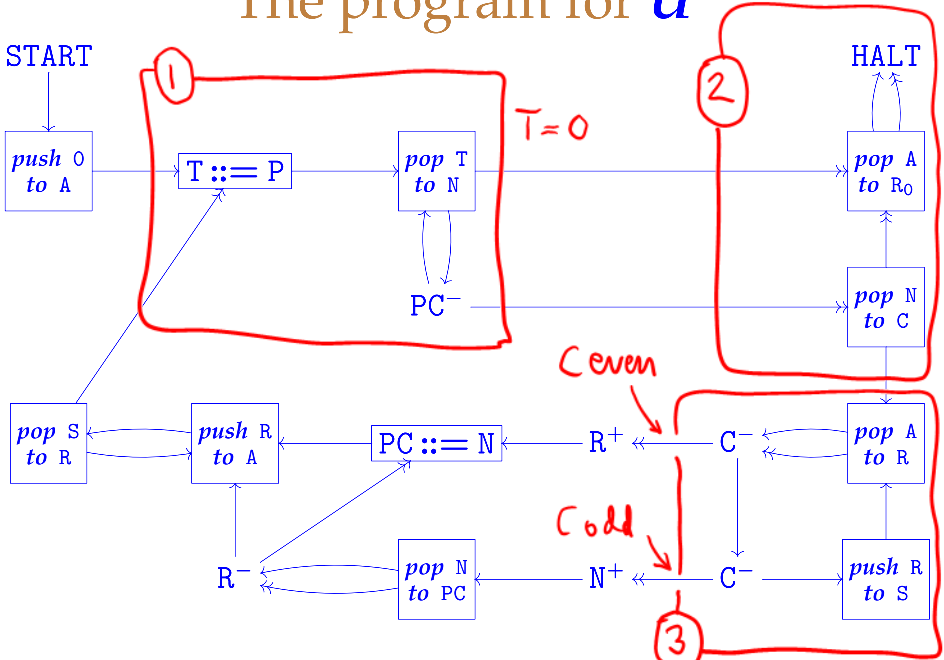
The program for *U*



The program for *U*



The program for *U*



The program for U