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 ≡ P code of the RM to be simulated

R_2 ≡ A code of current register contents of simulated RM

R_3 ≡ PC program counter—number of the current instruction (counting from 0)

R_4 ≡ N code of the current instruction body

R_5 ≡ C type of the current instruction body

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

R_7 ≡ S, R_8 ≡ T and R_9 ≡ Z are auxiliary registers.

Overall structure of U's program

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

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

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

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

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

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

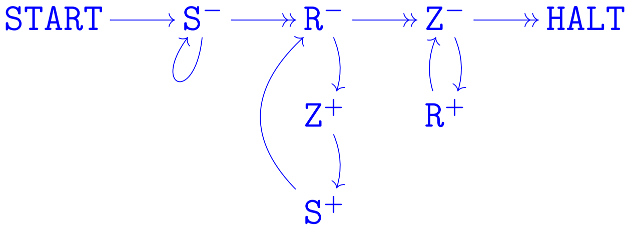
3 copy ith item of list in A to R; goto 4

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

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

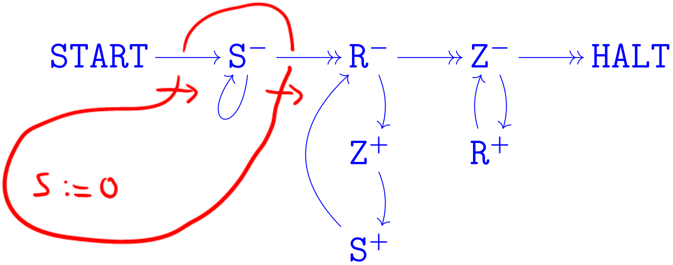
The program START⟶ $\boxed{\texttt{S ::= R}}$ ⟶HALT

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



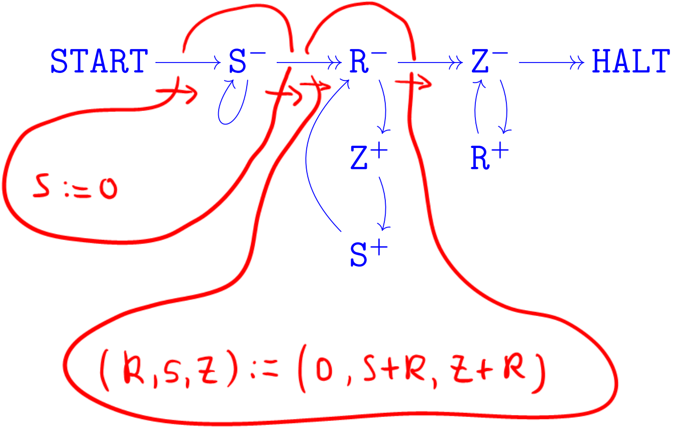
The program START $\longrightarrow$ $\boxed{\texttt{S ::= R}}$ $\longrightarrow$ HALT

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



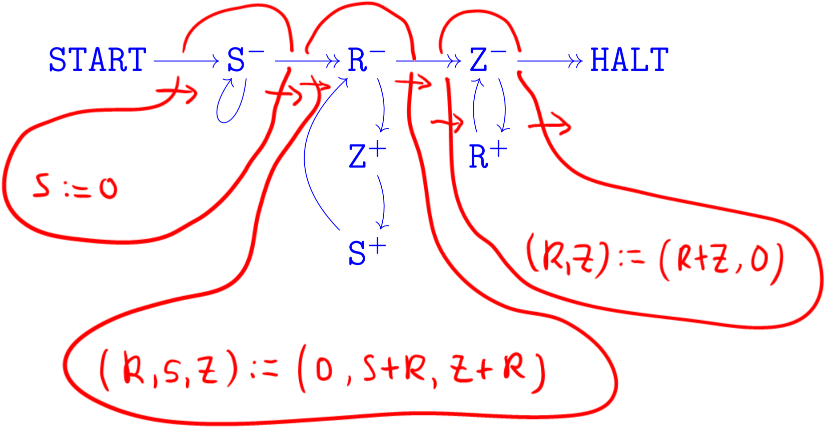
The program START $\longrightarrow$ $\boxed{S ::= R}$ $\longrightarrow$ HALT

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



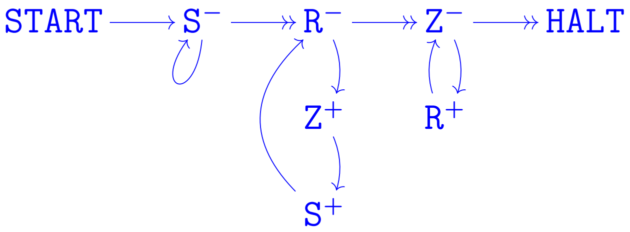
The program START→ S ::= R →HALT

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



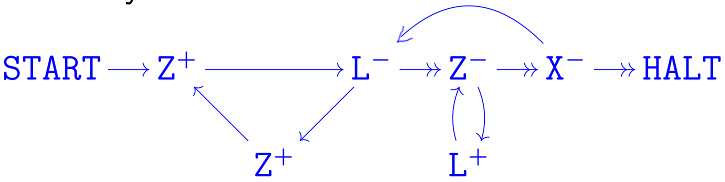
precondition:
$\texttt{R} = x$
$\texttt{S} = y$
$\texttt{Z} = 0$

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

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

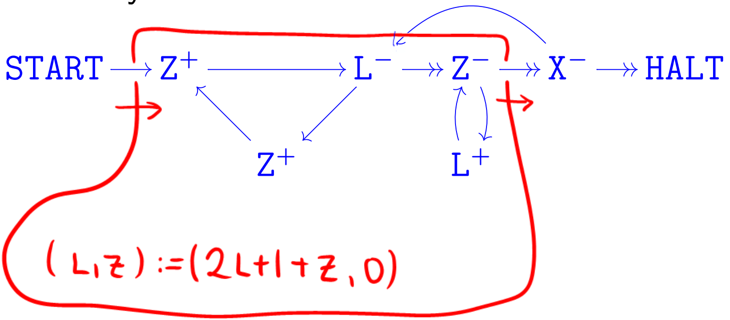
$2^{\mathtt{X}}(2\mathtt{L}+1)$

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



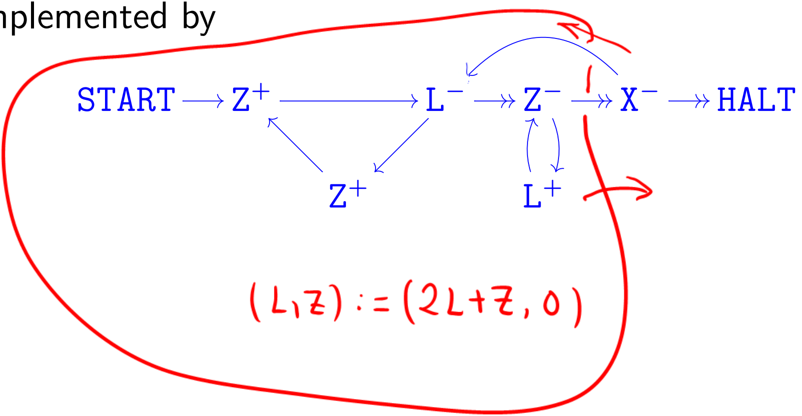
The program START ⟶ **push** X **to** L ⟶ HALT

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



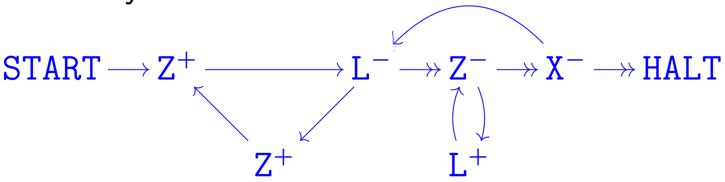
The program START → [*push* X *to* L] → HALT

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



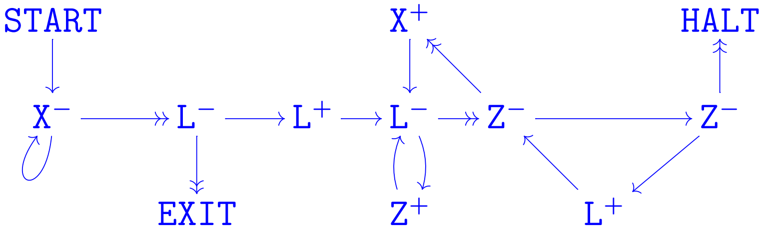
precondition:
$\mathtt{X} = x$
$\mathtt{L} = \ell$
$\mathtt{Z} = 0$

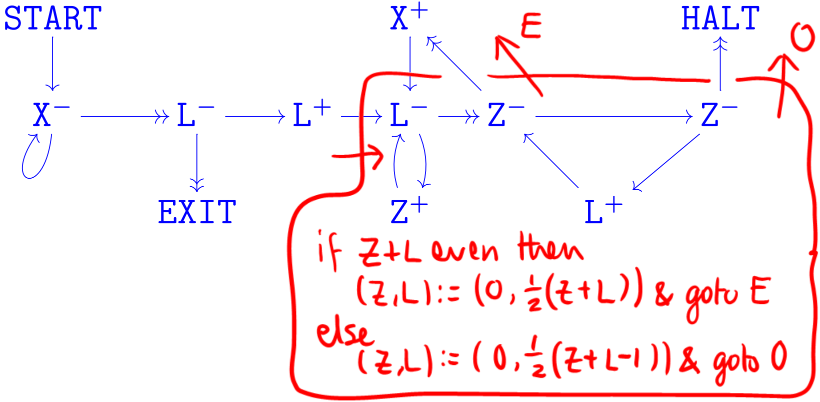
postcondition:
$\mathtt{X} = 0$
$\mathtt{L} = \langle\langle x, \ell \rangle\rangle = 2^x(2\ell + 1)$
$\mathtt{Z} = 0$

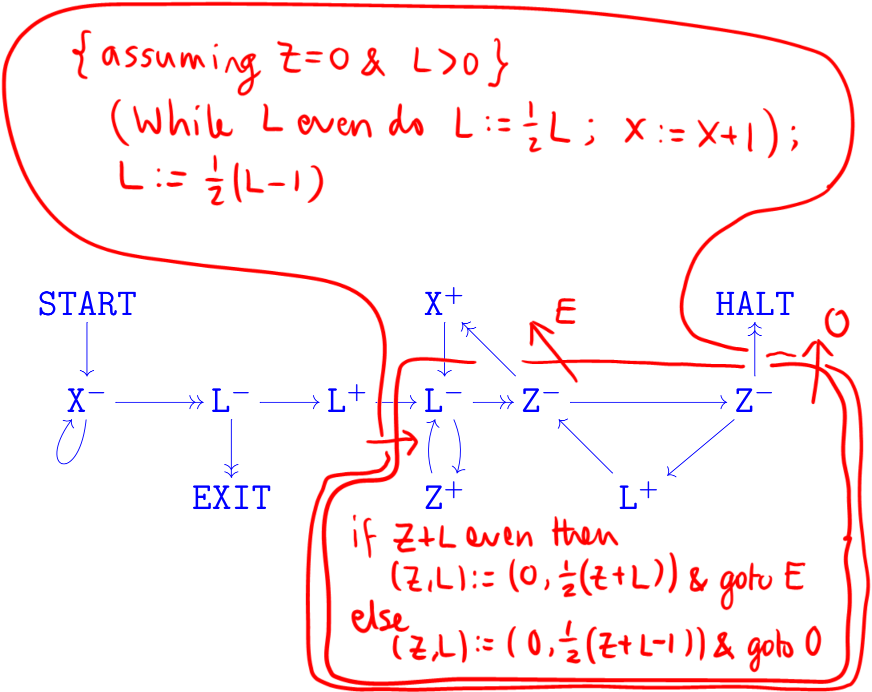
The program START $\longrightarrow$ $\boxed{\textit{pop}\ \mathtt{L}\ \textit{to}\ \mathtt{X}}$ $\longrightarrow$ HALT, $\twoheadrightarrow$ EXIT specified by

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

can be implemented by



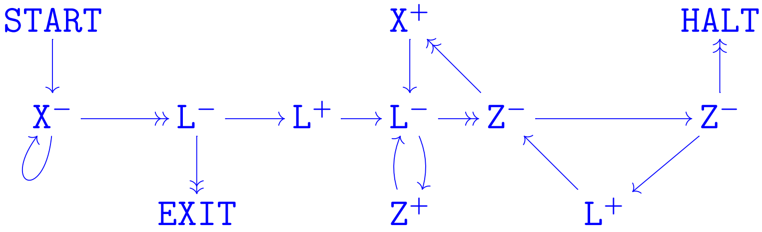

START
X⁻
L⁻
EXIT
L⁺
X⁺
L⁻
Z⁺
Z⁻
E
HALT
O
Z⁻
L⁺
if Z+L even then
(Z,L) := (0, ½(Z+L)) & goto E
else
(Z,L) := (0, ½(Z+L−1)) & goto O


{assuming Z=0 & L>0}
(While L even do L := ½L; X := X+1);
L := ½(L−1)
START
X⁻
EXIT
L⁻
L⁺
X⁺
L⁻
Z⁺
Z⁻
E
L⁺
Z⁻
HALT
O
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 START $\longrightarrow$ [*pop* L *to* X] $\longrightarrow$ HALT, $\twoheadrightarrow$ EXIT specified by

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

can be implemented by



Overall structure of U's program

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

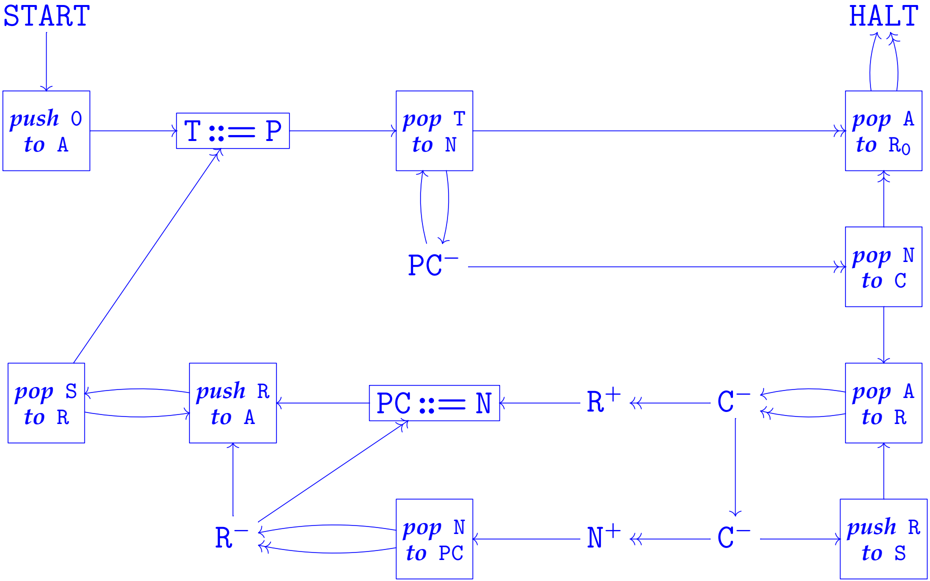
2 if $\texttt{N} = 0$ then copy 0th item of list in $\texttt{A}$ to $\texttt{R}_0$ and halt, else (decode $\texttt{N}$ as $\langle\!\langle y, z \rangle\!\rangle$; $\texttt{C} ::= y$; $\texttt{N} ::= z$; goto 3)

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

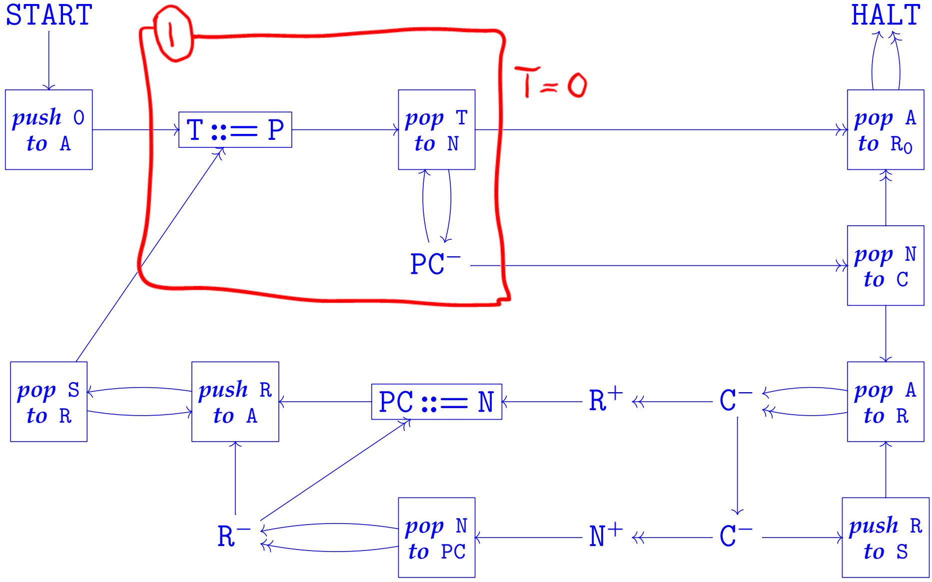
3 copy ith item of list in $\texttt{A}$ to $\texttt{R}$; goto 4

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

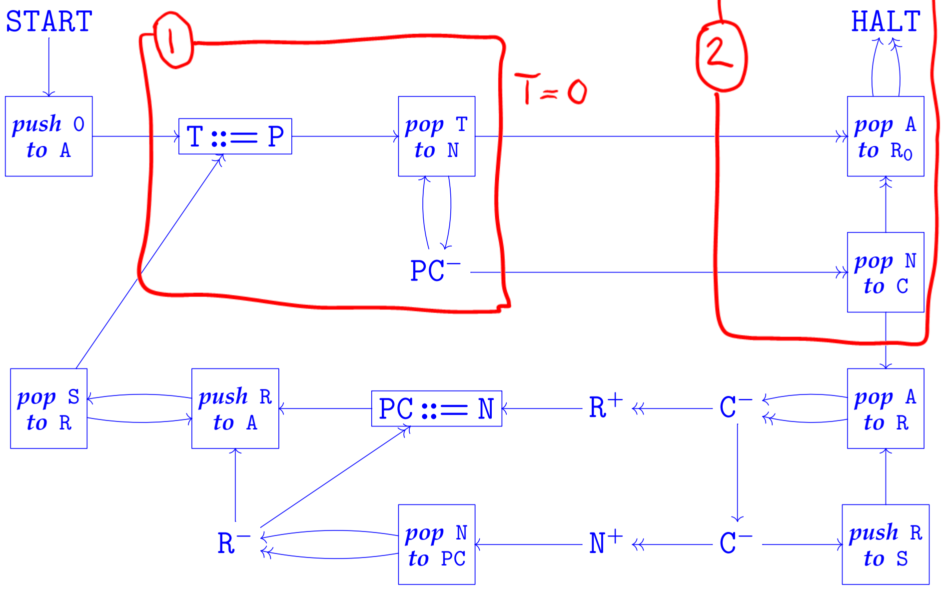
The program for U



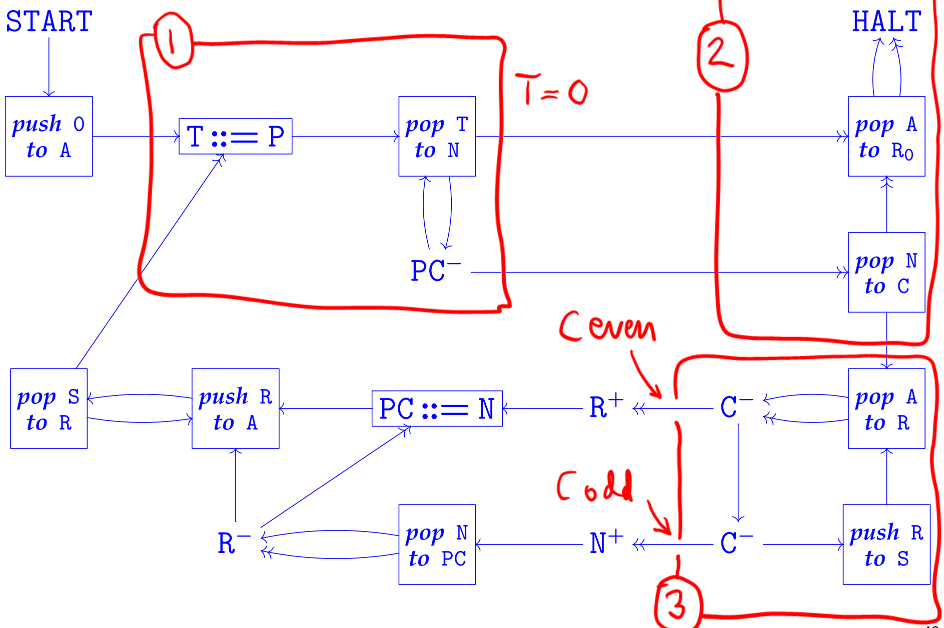
The program for U



The program for U



The program for U



The program for $\boldsymbol{U}$