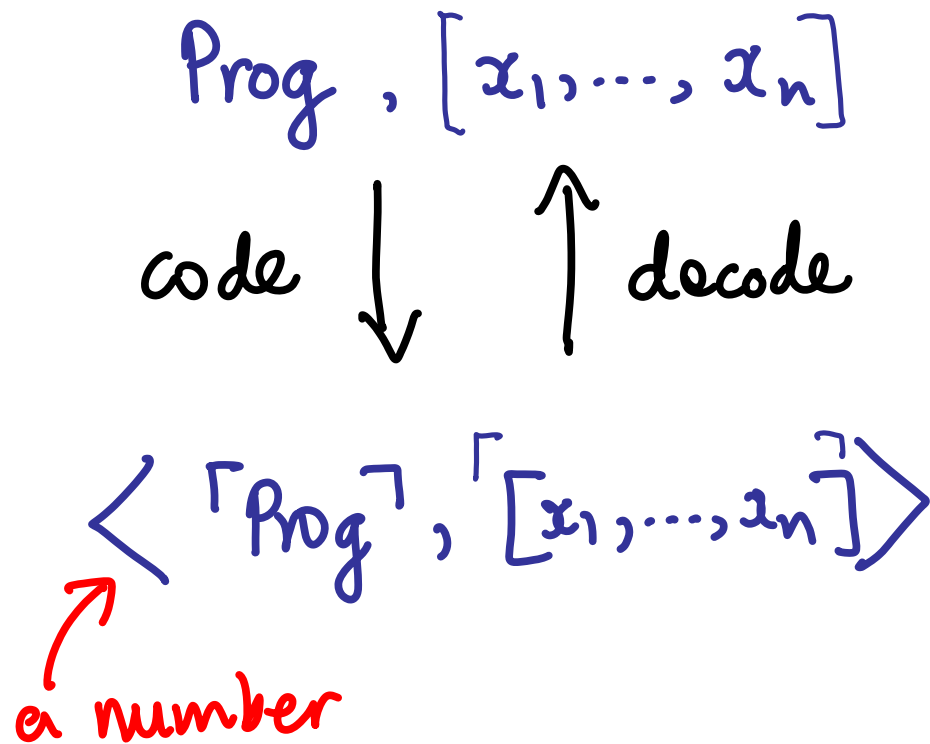


N.B. from Monday
Comp. Th. lectures
are 10 - 11 am

Universal register machine, ***U***

"Effective" numerical codes



Want numerical codings

$\langle _ _ \rangle, \lceil _ \rceil, \lceil _ _ _ _ \rceil$

So that

$\cdot \xrightarrow{\text{decode}} \cdot \xrightarrow{\text{run}} \cdot$

is RM computable

High-level specification

Universal RM U carries out the following computation, starting with $R_0 = \mathbf{0}$, $R_1 = e$ (code of a program), $R_2 = a$ (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, \dots, a_n
- ▶ carry out the computation of the RM program P starting with $R_0 = \mathbf{0}, R_1 = a_1, \dots, R_n = a_n$ (and any other registers occurring in P set to $\mathbf{0}$).

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.

R_0 result of the simulated RM computation (if any).

Overall structure of U 's program

1 copy PC th 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 $R_i^+ \rightarrow L_z$, or $C = 2i + 1$ is odd and current instruction is $R_i^- \rightarrow L_j, L_k$ where $z = \langle j, k \rangle$ }

3 copy i th 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

Overall structure of U 's program

1 copy PC th 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 $R_i^+ \rightarrow L_z$, or $C = 2i + 1$ is odd and current instruction is $R_i^- \rightarrow L_j, L_k$ where $z = \langle j, k \rangle$ }

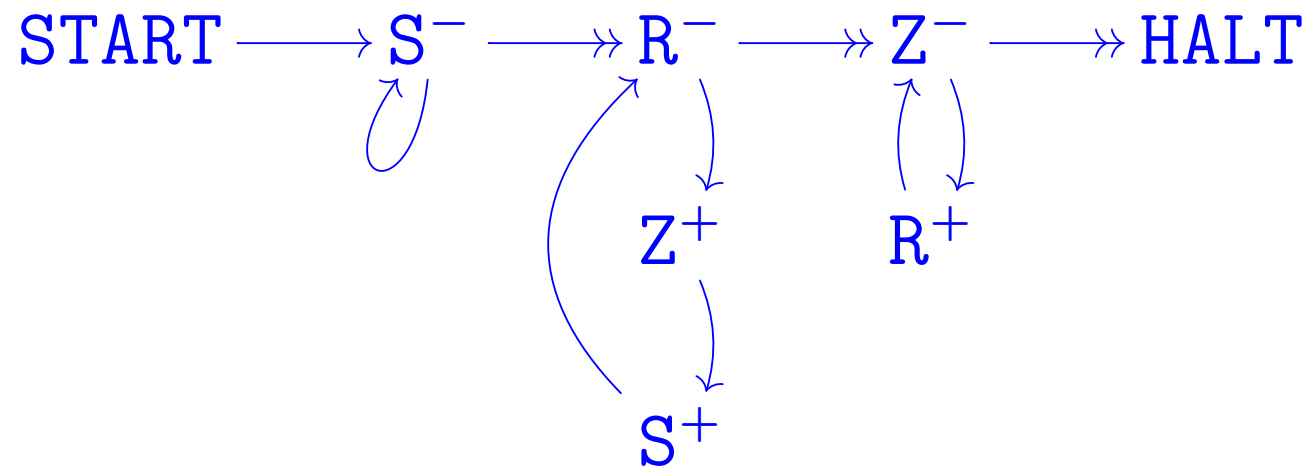
3 copy i th item of list in A to R ; goto 4

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...

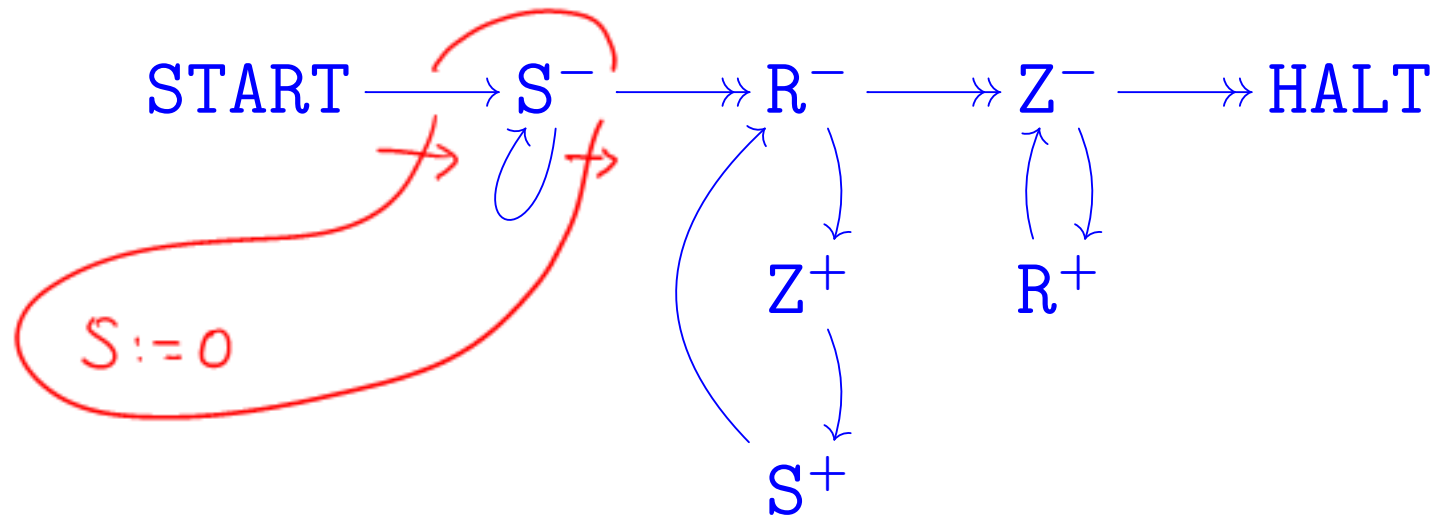
The program $START \rightarrow \boxed{S ::= R} \rightarrow HALT$

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



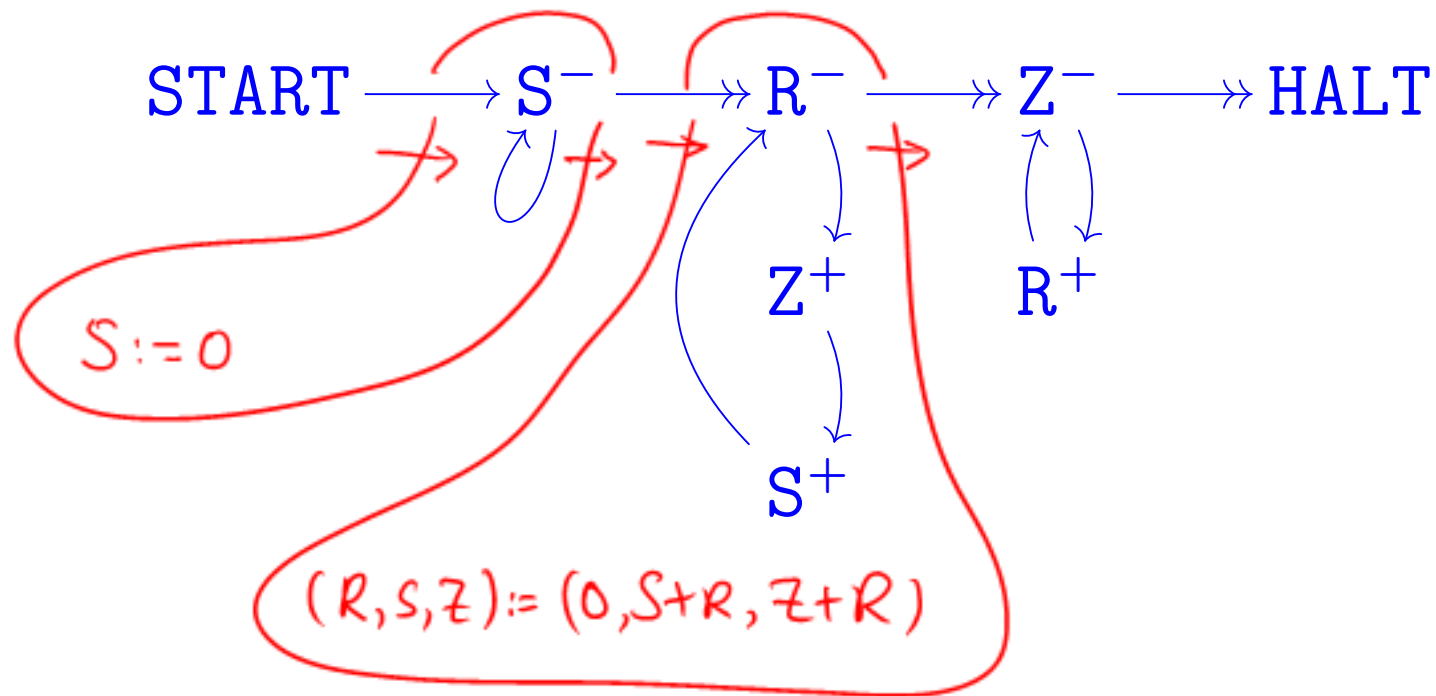
The program $START \rightarrow \boxed{S ::= R} \rightarrow HALT$

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



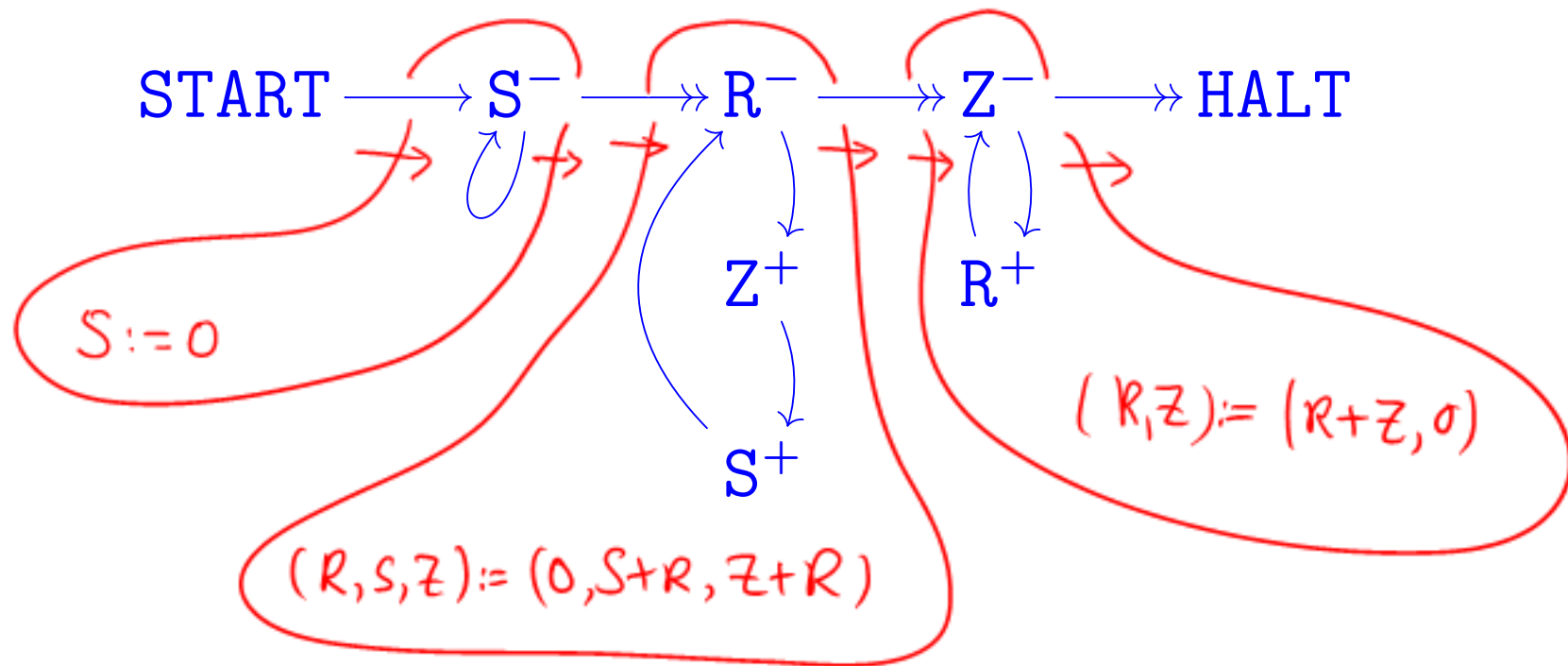
The program $START \rightarrow \boxed{S ::= R} \rightarrow HALT$

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



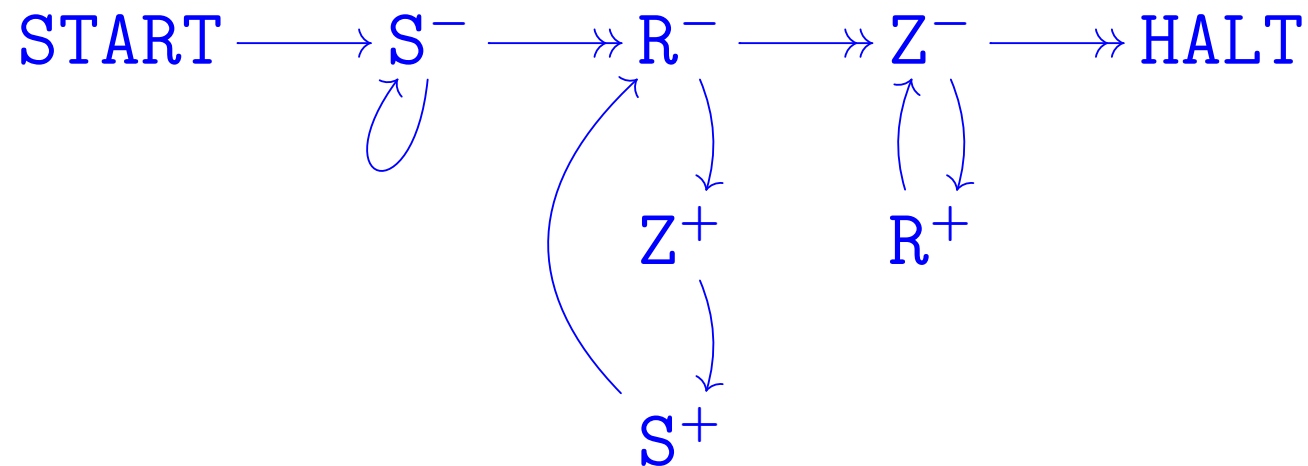
The program $START \rightarrow \boxed{S ::= R} \rightarrow HALT$

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



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

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



precondition:

$R = x$

$S = y$

$Z = 0$

postcondition:

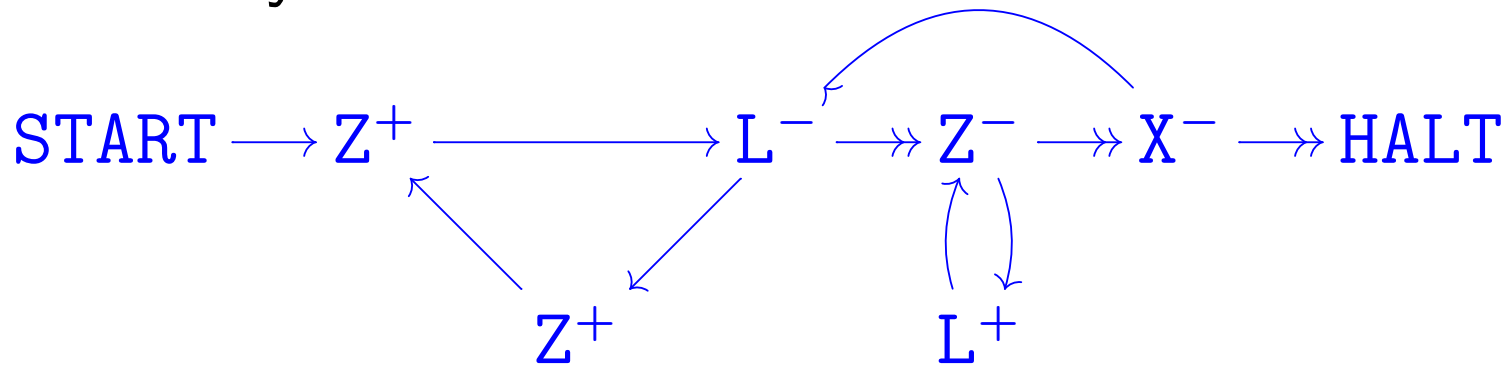
$R = x$

$S = x$

$Z = 0$

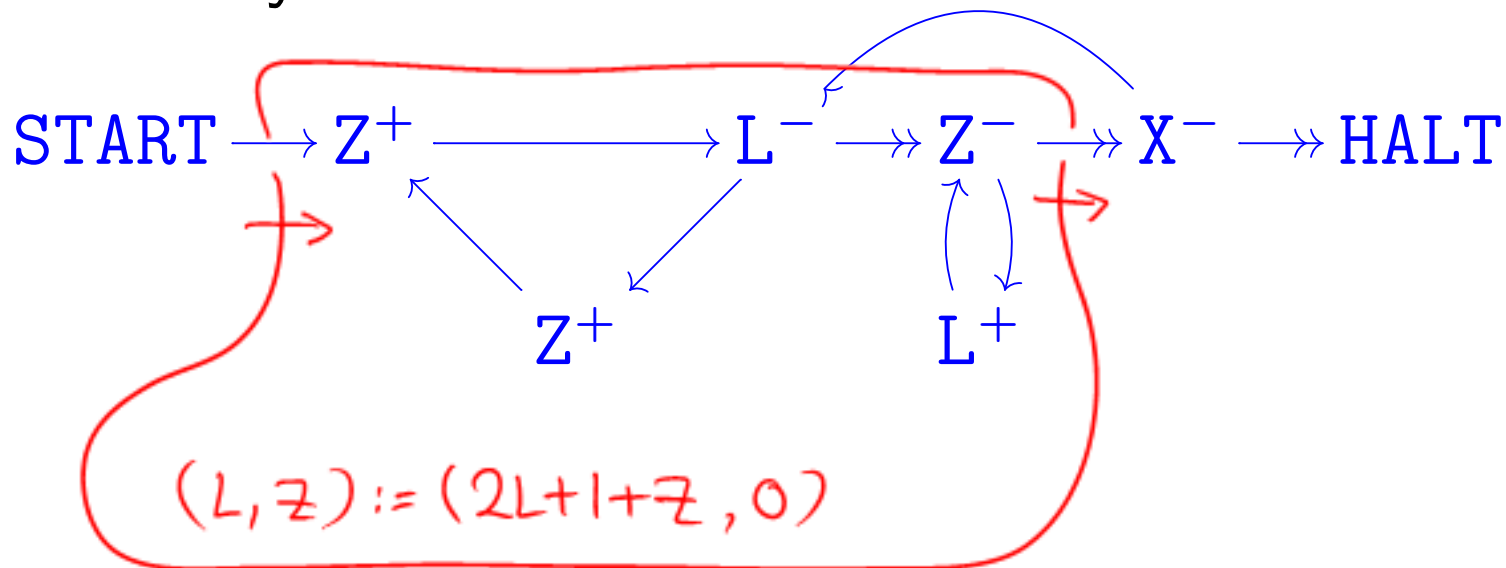
The program $\text{START} \rightarrow \boxed{\begin{array}{l} \text{push } X \\ \text{to } L \end{array}} \rightarrow \text{HALT}$ $2^X(2L + 1)$

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



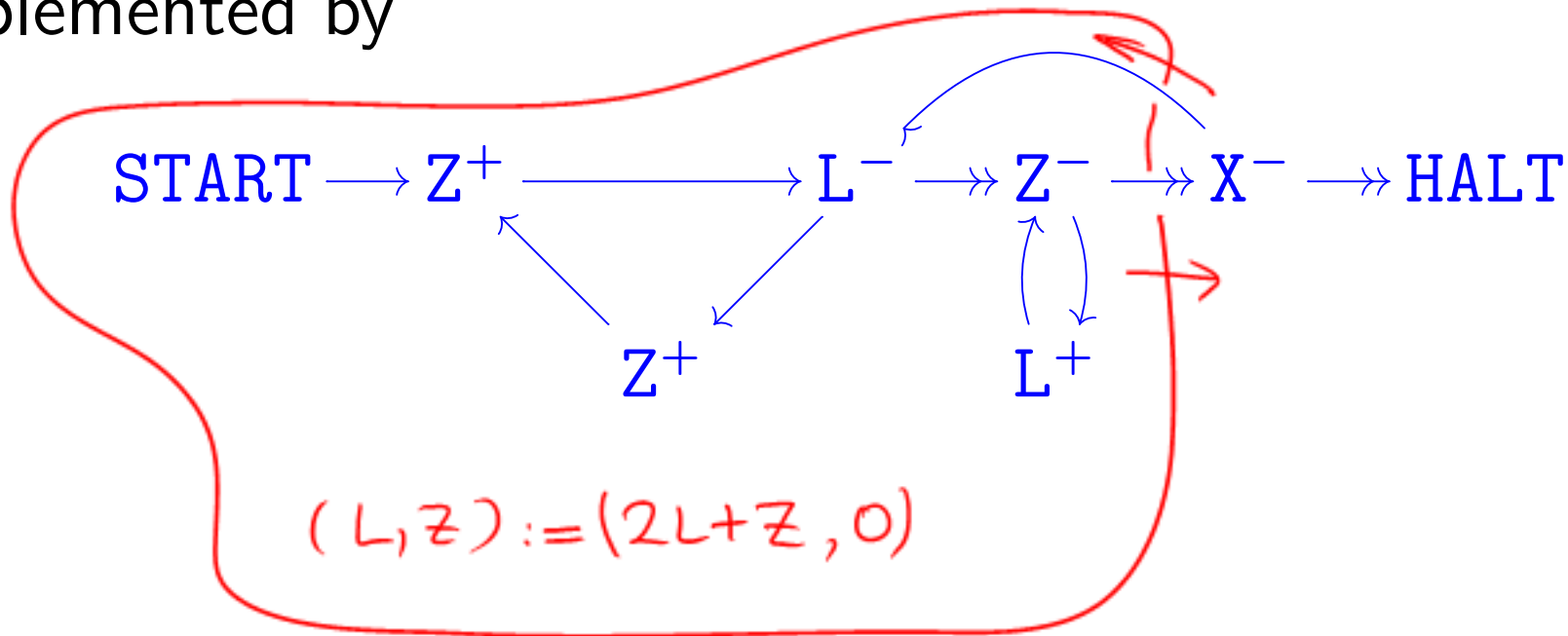
The program $\text{START} \rightarrow \boxed{\begin{array}{l} \text{push } X \\ \text{to } L \end{array}} \rightarrow \text{HALT}$

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



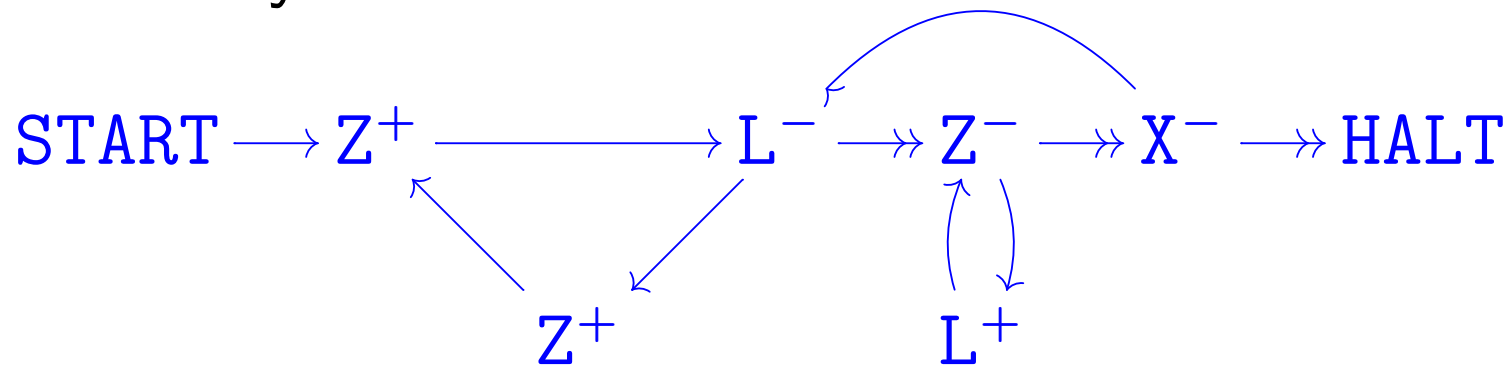
The program $\text{START} \rightarrow \boxed{\begin{array}{l} \text{push } X \\ \text{to } L \end{array}} \rightarrow \text{HALT}$

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



The program $\text{START} \rightarrow \boxed{\begin{array}{l} \text{push } X \\ \text{to } L \end{array}} \rightarrow \text{HALT}$

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



precondition:

$$X = x$$

$$L = \ell$$

$$Z = 0$$

postcondition:

$$X = 0$$

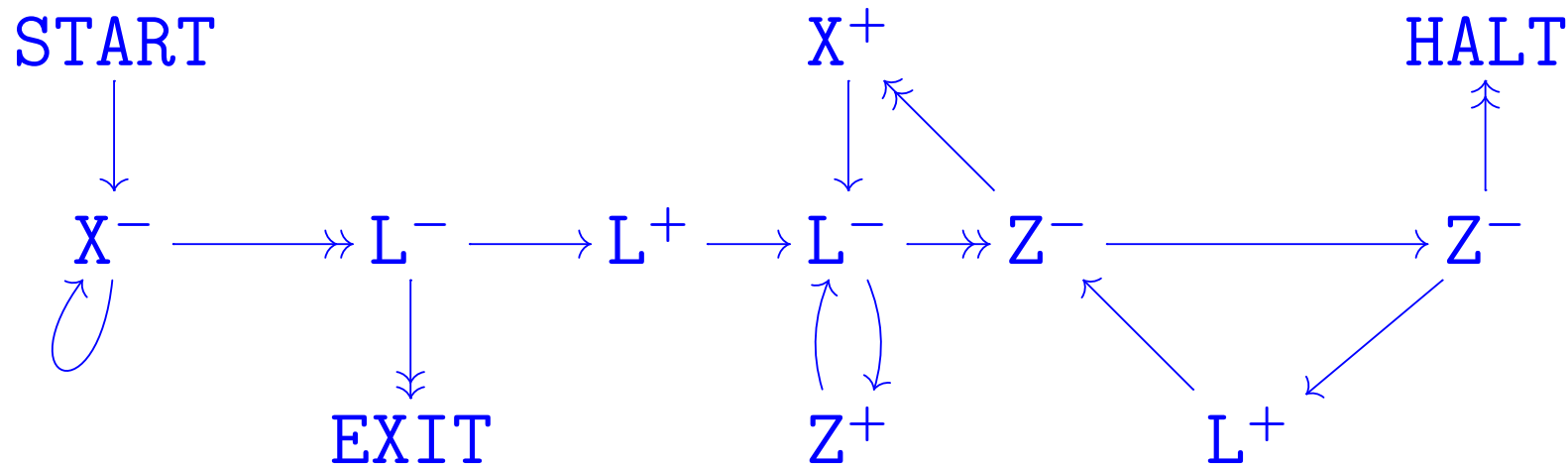
$$L = \langle\langle x, \ell \rangle\rangle = 2^x(2\ell + 1)$$

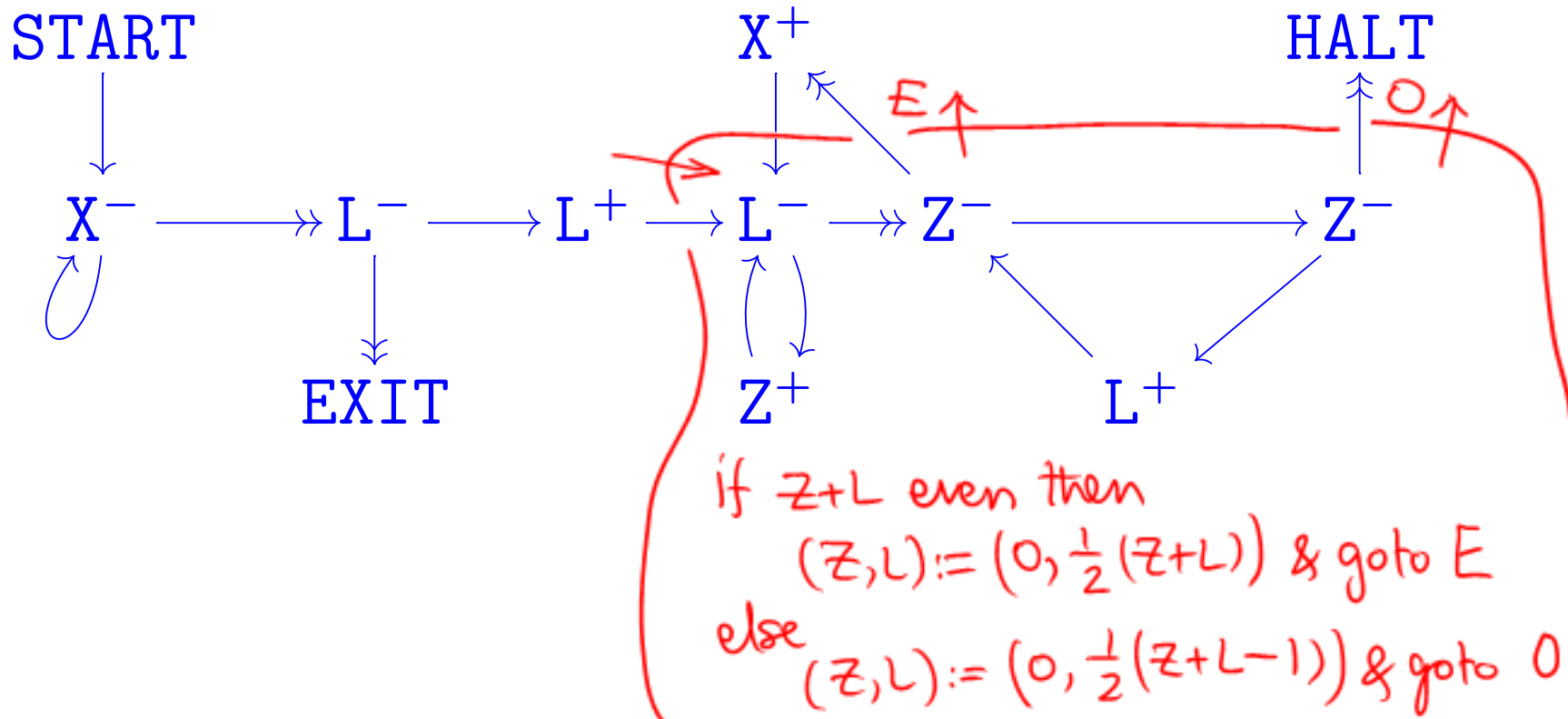
$$Z = 0$$

The program $START \rightarrow \left[\begin{array}{l} \text{pop } L \\ \text{to } X \end{array} \right] \begin{array}{l} \rightarrow \text{HALT} \\ \rightarrow \text{EXIT} \end{array}$ specified by

“*if* $L = 0$ *then* ($X ::= 0$; *goto* EXIT) *else*
let $L = \langle\langle x, \ell \rangle\rangle$ *in* ($X ::= x$; $L ::= \ell$; *goto* HALT)”

can be implemented by



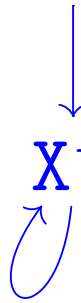


{ assuming $Z=0$ & $L>0$ }

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

$L := \frac{1}{2}(L-1)$

START



X^+



$E \uparrow$



L^+

HALT

$0 \uparrow$



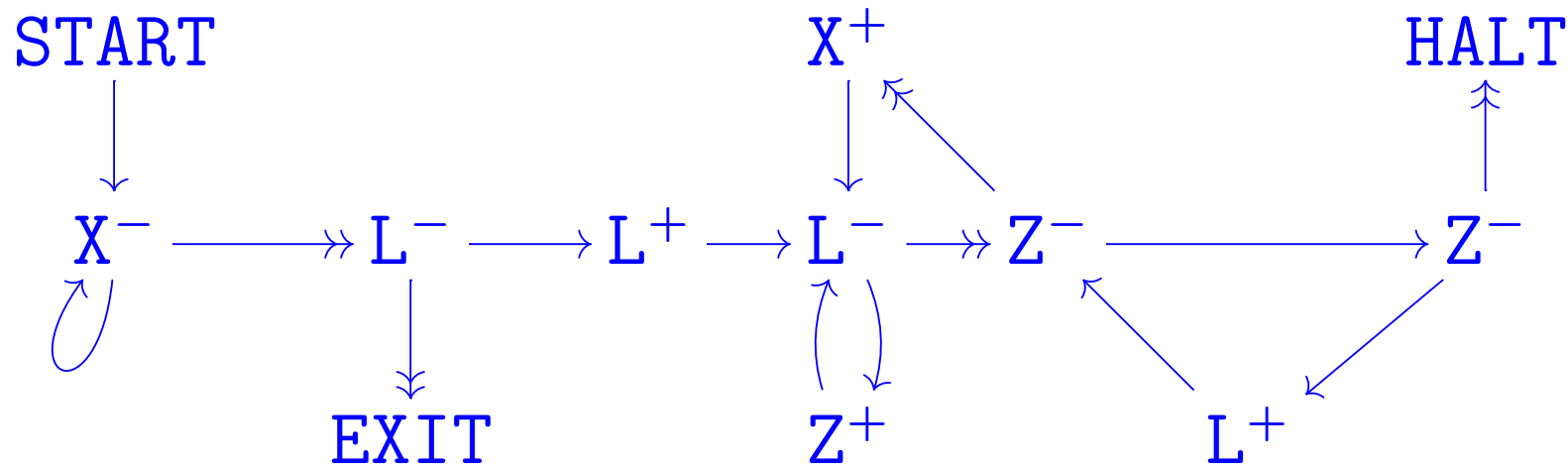
EXIT

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 0

The program $\text{START} \rightarrow \left[\begin{array}{l} \text{pop } L \\ \text{to } X \end{array} \right] \begin{array}{l} \rightarrow \text{HALT} \\ \rightarrow \text{EXIT} \end{array}$ specified by

“*if* $L = 0$ *then* ($X ::= 0$; *goto* EXIT) *else*
let $L = \langle\langle x, \ell \rangle\rangle$ *in* ($X ::= x$; $L ::= \ell$; *goto* HALT)”

can be implemented by



Overall structure of U 's program

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

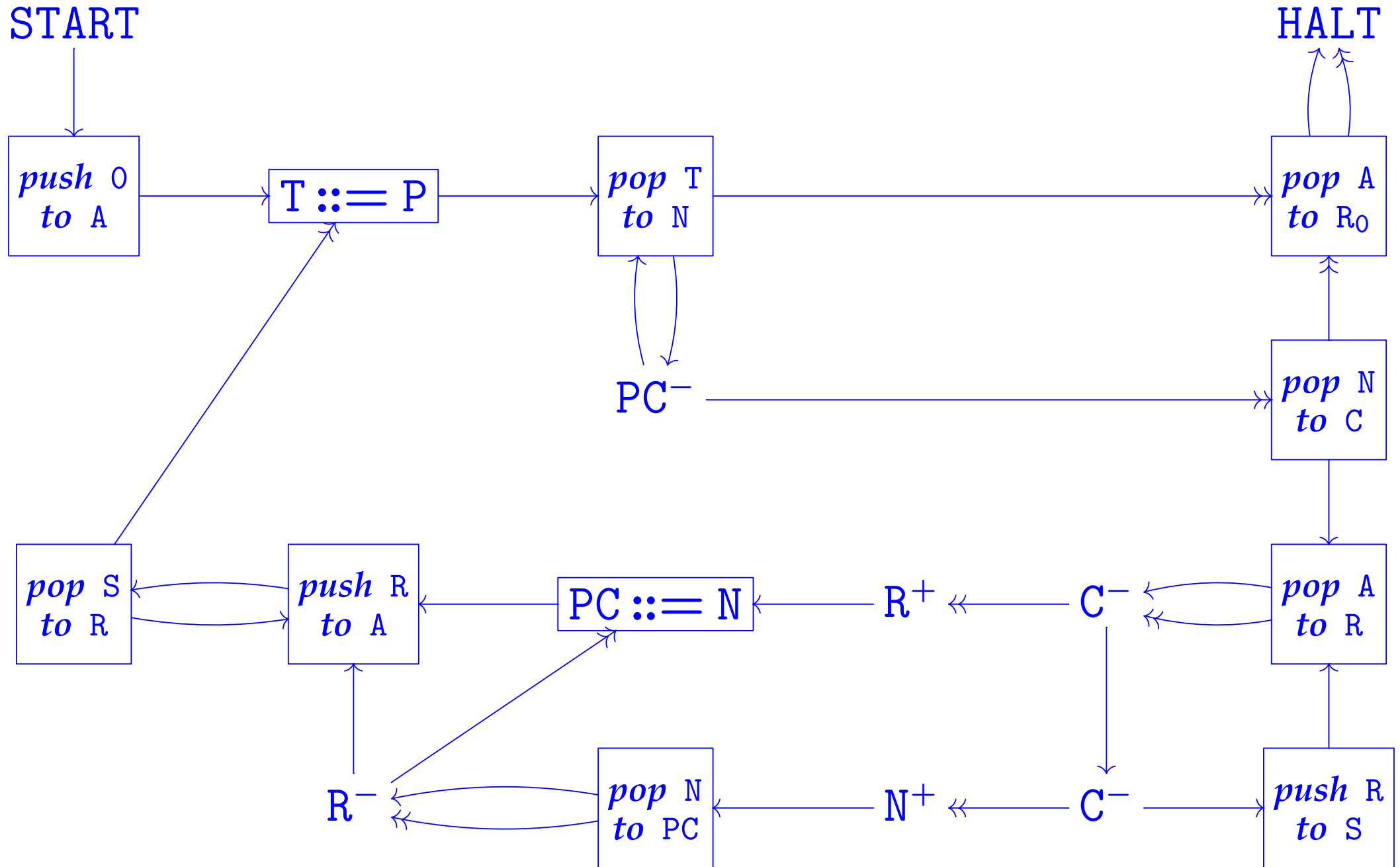
2 if $N = 0$ then copy 0 th item of list in A to R_0 and halt, else (decode N as $\langle\langle y, z \rangle\rangle$; $C ::= y$; $N ::= z$; goto **3**)

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

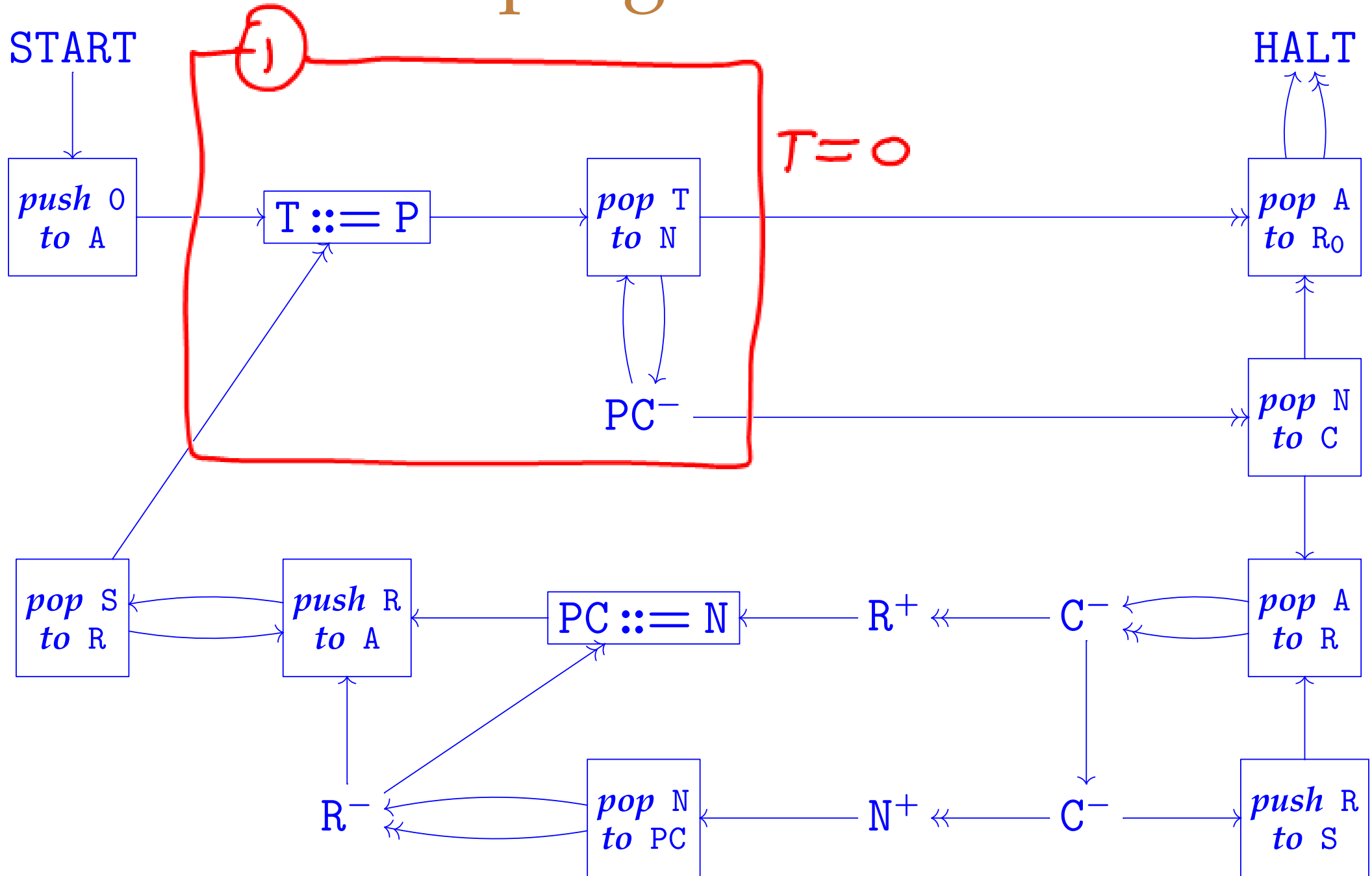
3 copy i th item of list in A to R ; goto **4**

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

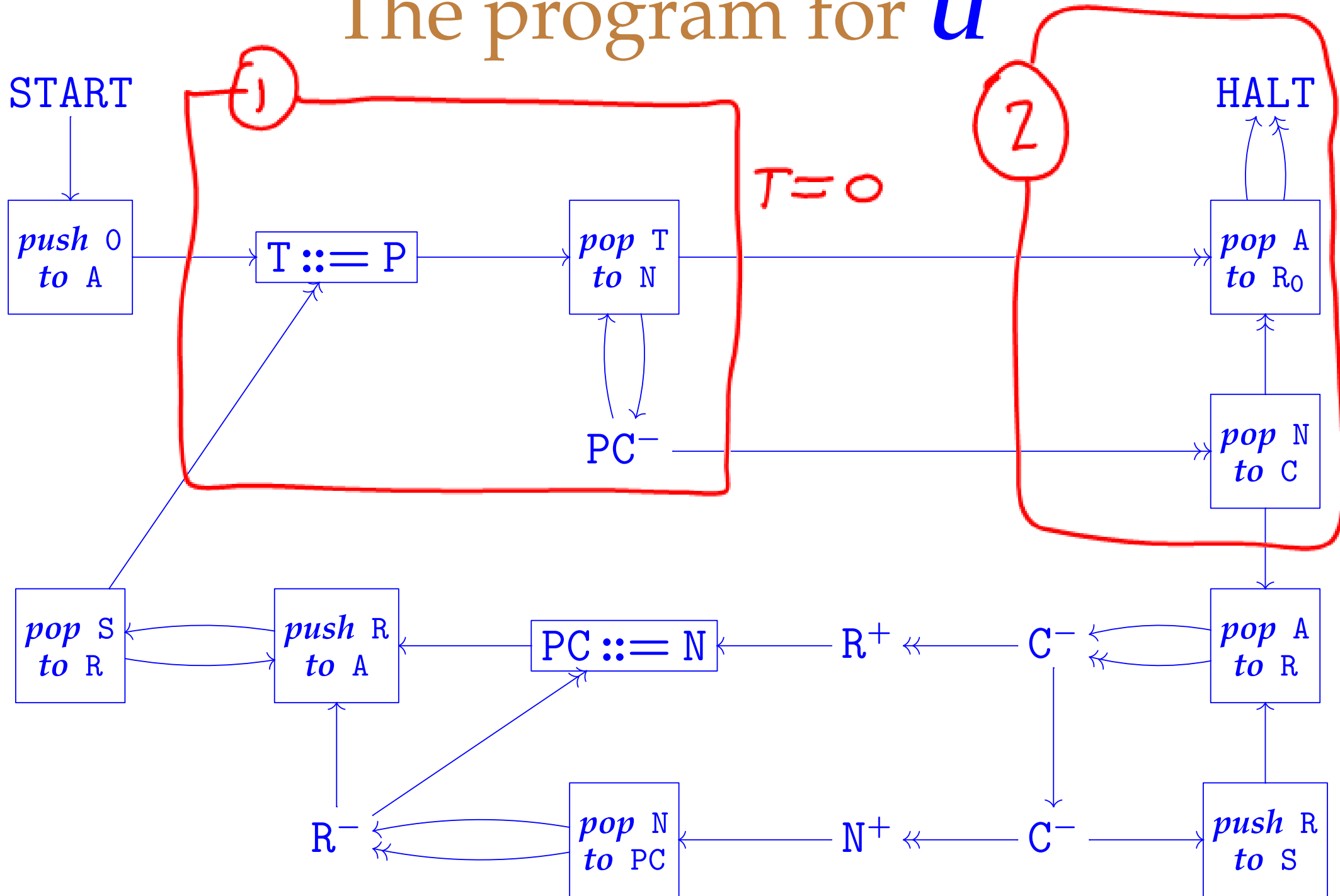
The program for U



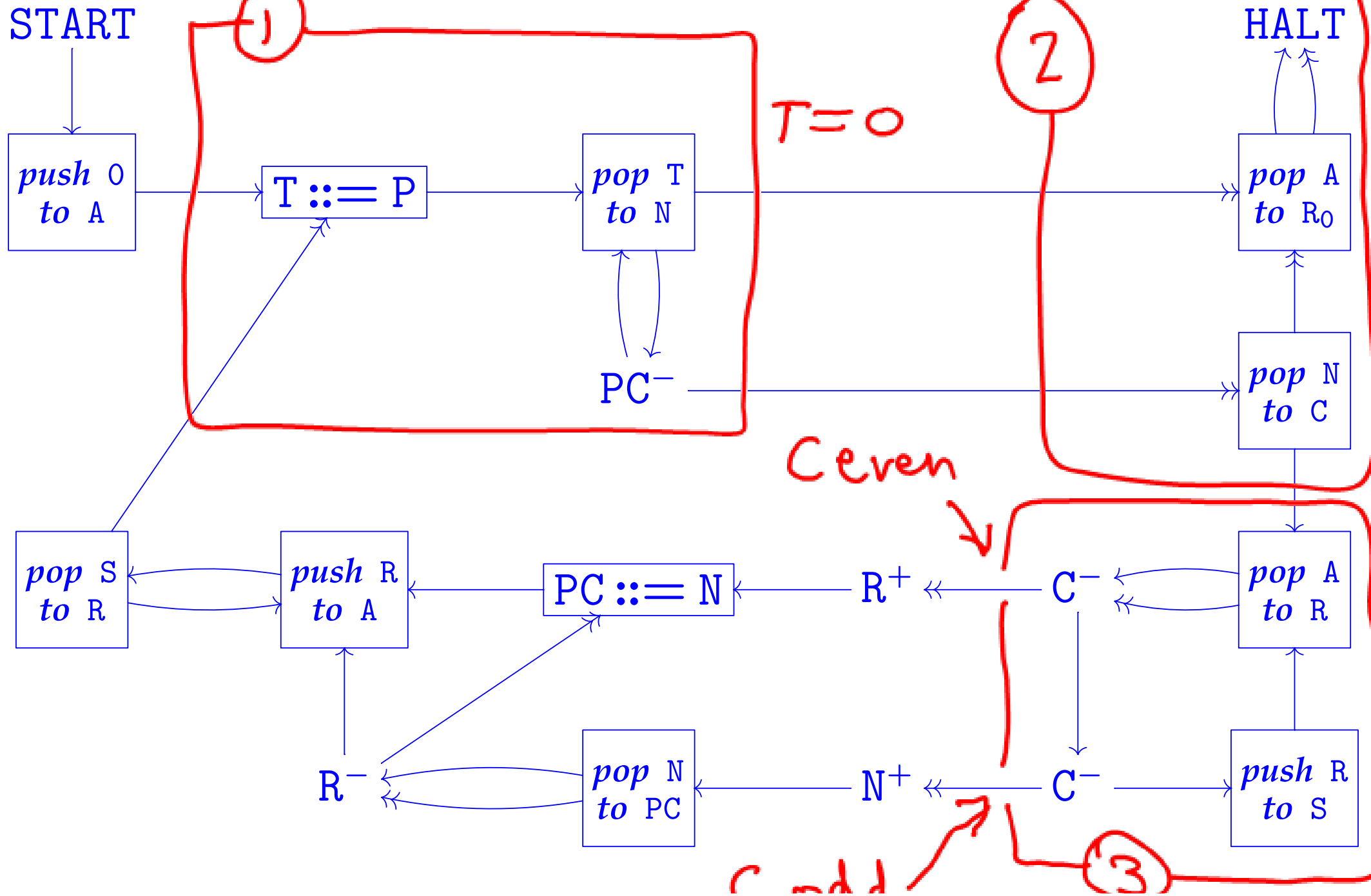
The program for U



The program for U



The program for U



The program for U

