Universal register machine, u

High-level specification

Universal RM U carries out the following computation, starting with $R_0 = 0$, $R_1 = e$ (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 $a_1, \ldots, a_n$
- ▸ carry out the computation of the RM program P starting with $R_0 = 0, R_1 = a_1, \ldots, R_n = a_n$ (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 `PC`th item of list in `P` to `N` (~~halting if PC > length of list~~); goto $\boxed{2}$

$\boxed{2}$ if $N = 0$ then copy $\mathbf{0}$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 $\boxed{3}$)

{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 $z = \langle j, k \rangle$}
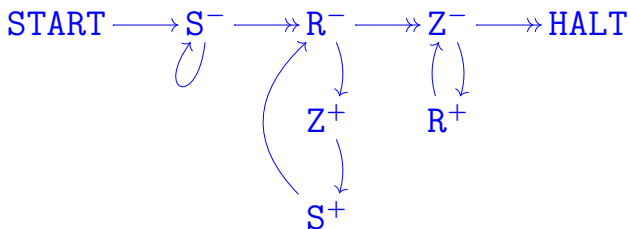
$\boxed{3}$ copy ith item of list in `A` to `R`; goto $\boxed{4}$

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

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

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

to copy the contents of R to S can be implemented by
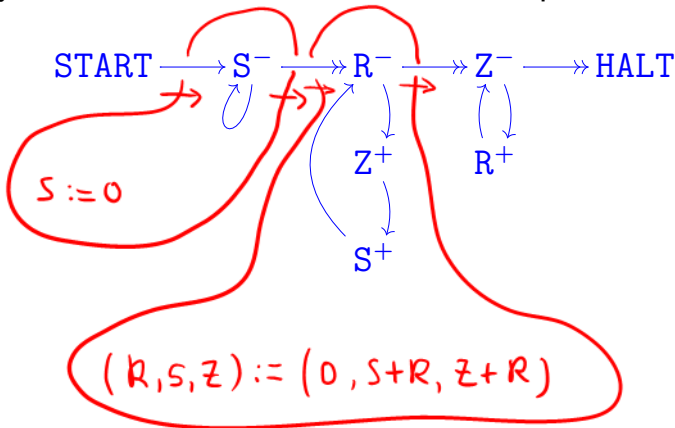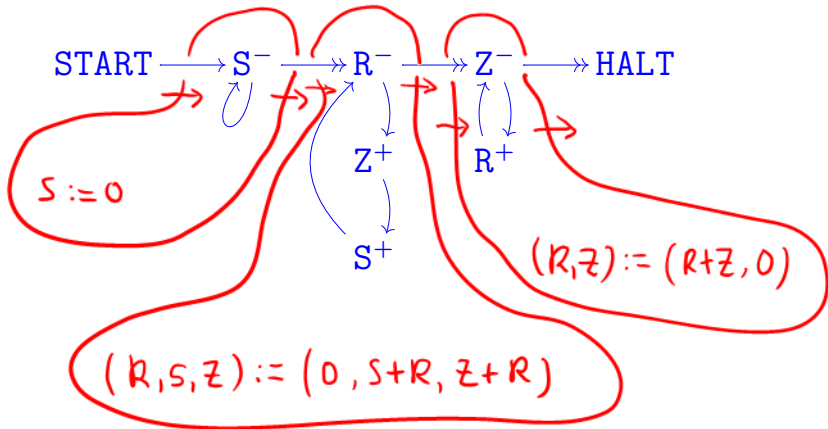
The program START$\rightarrow$ $\boxed{\text{S} ::= \text{R}}$ $\rightarrow$HALT

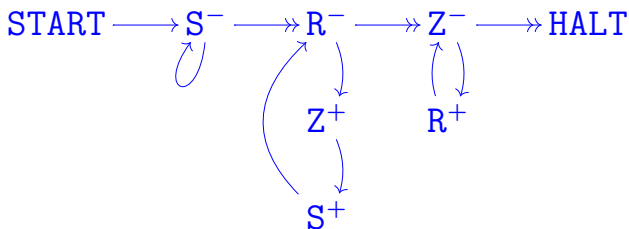to copy the contents of R to S can be implemented by

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 START→ $\boxed{\text{S} ::= \text{R}}$ →HALT

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

$$\text{START} \longrightarrow \text{S}^- \longrightarrow \text{R}^- \longrightarrow \text{Z}^- \longrightarrow \text{HALT}$$

$$\text{Z}^+ \qquad \text{R}^+$$

$$\text{S}^+$$

$S := 0$

$(R, Z) := (R+Z, 0)$

$(R, S, Z) := (0, S+R, Z+R)$

The program START→ S ::= R →HALT

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

START ⟶ S⁻ ⟶» R⁻ ⟶» Z⁻ ⟶» HALT
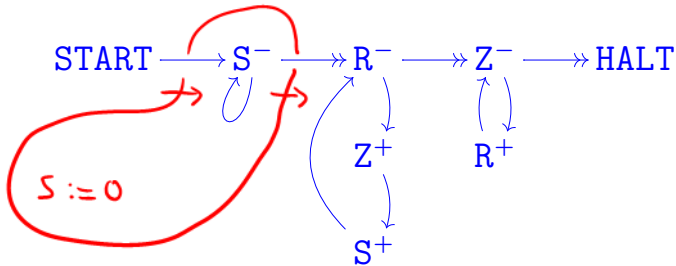
precondition:
$R = x$
$S = y$
$Z = 0$

postcondition:
$R = x$
$S = x$
$Z = 0$

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

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

$$\mathrm{START} \longrightarrow \mathrm{Z}^{+} \longrightarrow \mathrm{L}^{-} \longrightarrow\!\!\!\!\rightarrow \mathrm{Z}^{-} \longrightarrow\!\!\!\!\rightarrow \mathrm{X}^{-} \longrightarrow\!\!\!\!\rightarrow \mathrm{HALT}$$
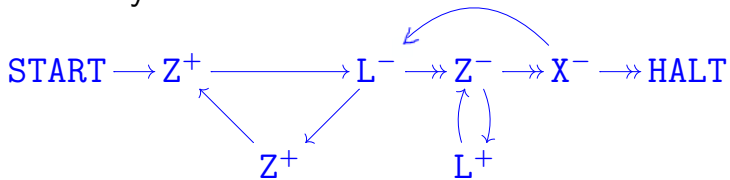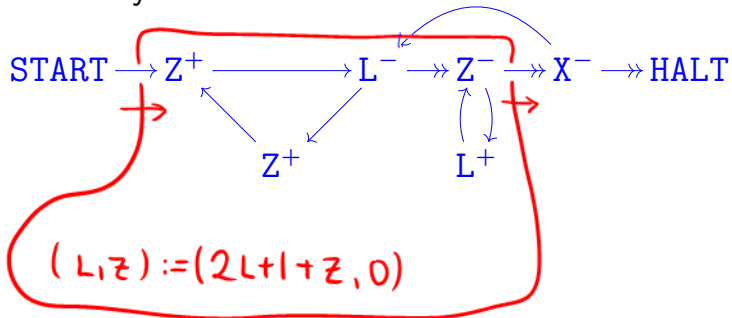
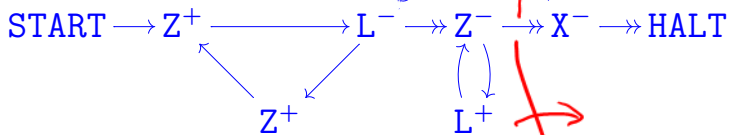$$\mathrm{Z}^{+} \qquad\qquad \mathrm{L}^{+}$$

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

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

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

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

$\text{START} \longrightarrow \text{Z}^+ \longrightarrow \text{L}^- \longrightarrow\!\!\!\!\twoheadrightarrow \text{Z}^- \longrightarrow\!\!\!\!\twoheadrightarrow \text{X}^- \longrightarrow \text{HALT}$

$\text{Z}^+ \qquad\qquad \text{L}^+$

$(L, Z) := (2L + Z, 0)$

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

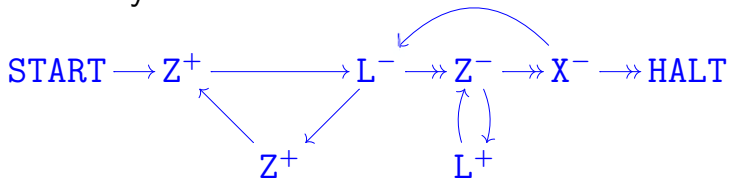to carry out the assignment $(\text{X}, \text{L}) ::= (\mathbf{0}, \text{X} :: \text{L})$ can be implemented by

$$\text{START} \longrightarrow \text{Z}^+ \longrightarrow \text{L}^- \twoheadrightarrow \text{Z}^- \longrightarrow \text{X}^- \twoheadrightarrow \text{HALT}$$

$$\text{Z}^+ \qquad\qquad \text{L}^+$$

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

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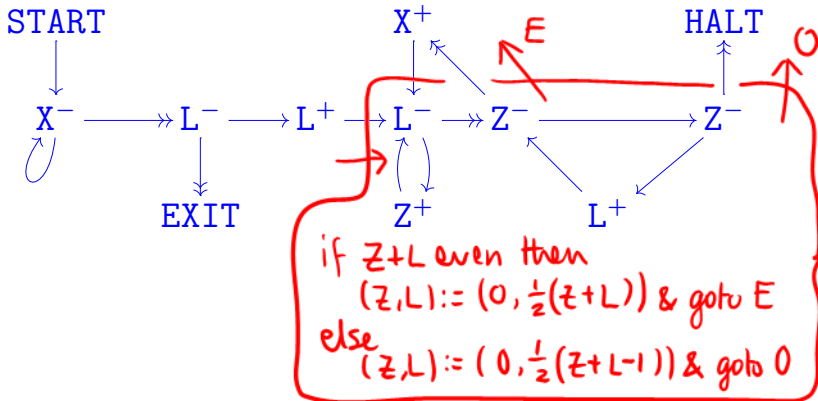$\text{Z} = \mathbf{0}$

The program $\text{START} \rightarrow \boxed{\begin{array}{l} pop \; \text{L} \\ to \; \text{X} \end{array}} \begin{array}{l} \rightarrow \text{HALT} \\ \twoheadrightarrow \text{EXIT} \end{array}$ specified by

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

can be implemented by

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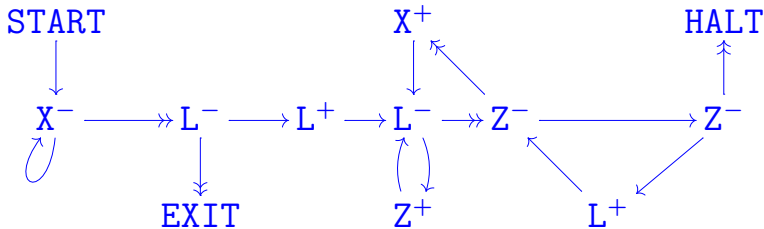
{assuming Z=0 & L>0}
  (while L even do L := ½ L ; X := X+1);
  L := ½ (L−1)

START

X⁻ → L⁻ → L⁺ → L⁻ ⇀ Z⁻ → Z⁻

X⁺   E   HALT

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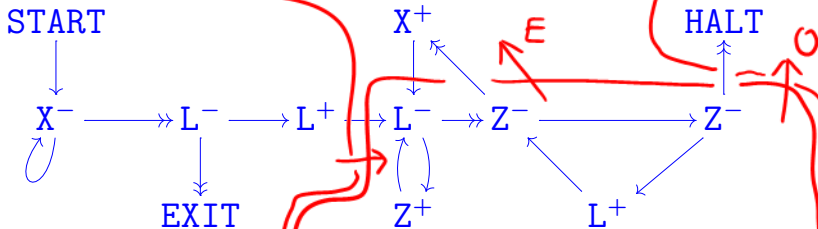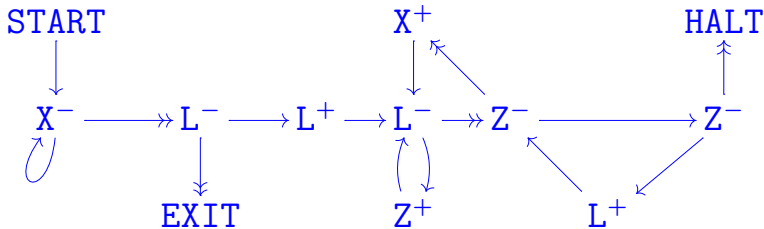
if Z+L even then
    (Z,L):= (0, ½(Z+L)) & goto E
else
    (Z,L):= (0, ½(Z+L−1)) & goto 0

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

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

can be implemented by

Overall structure of U's program

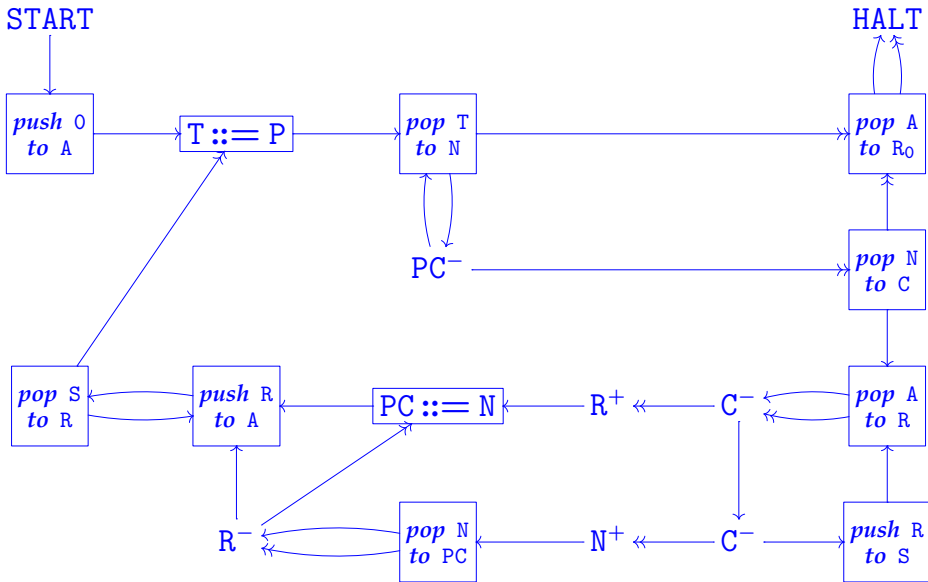1. copy `PC`th item of list in `P` to `N` (~~halting if `PC` > length of list~~); goto 2.

2. if $N = 0$ then copy $\mathbf{0}$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 $R_i^+ \to L_z$, or $C = 2i + 1$ is odd and current instruction is $R_i^- \to L_j, L_k$ where $z = \langle j, k \rangle$}

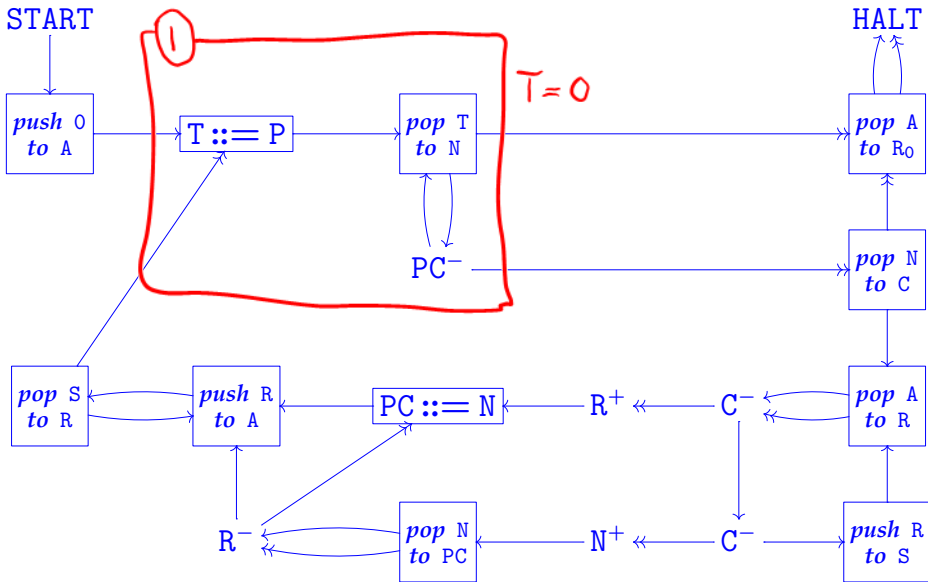3. copy ith item of list in `A` to `R`; goto 4.

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*

START

push 0 to A → T ::= P → pop T to N

T = 0

pop S to R ⇄ push R to A → PC ::= N ← R⁺ ⇐ C⁻

PC⁻

R⁻ ⇄ pop N to PC ← N⁺ ⇐ C⁻

HALT

pop A to R₀

pop N to C

pop A to R ⇄

push R to S

The program for *u*

The program for u

START

① T ::= P → *pop* T *to* N

push 0 *to* A

$T = 0$

PC⁻

② HALT

pop A *to* R_0

pop N *to* C

C even

④ *pop* S *to* R ⇄ *push* R *to* A ← PC ::= N ← R^+ ← C^- ⇄ *pop* A *to* R

R^- ⇄ *pop* N *to* PC ← N^+ ← C^- → *push* R *to* S

C odd

③

L4

49