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

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

2. If $N = 0$ then copy 0th item of list in $A$ to $R_0$ and halt, else (decode $N$ as $\langle y, z \rangle$; $C ::= y$; $N ::= z$; 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 \textit{U}'s program

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

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

\{at this point either \textit{C} = 2\textit{i} is even and current instruction is \textit{R}_{i}^{+} \rightarrow \textit{L}_{z},

or \textit{C} = 2\textit{i} + 1 is odd and current instruction is \textit{R}_{i}^{-} \rightarrow \textit{L}_{j}, \textit{L}_{k} where \textit{z} = \langle \langle \textit{j}, \textit{k} \rangle \rangle\}

3. copy \textit{i}th item of list in \textit{A} to \textit{R}; goto 4

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

To implement this, we need RMs for manipulating (codes of) lists of numbers. . .
The program $\text{START} \rightarrow \boxed{S ::= R} \rightarrow \text{HALT}$

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

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

$$S := 0$$

$$(R, S, Z) := (0, S+R, Z+R)$$
The program \( \text{START} \rightarrow \boxed{S := R} \rightarrow \text{HALT} \) 

to copy the contents of \( R \) to \( S \) can be implemented by

\[
S := 0
\]

\[
(R, S, Z) := (0, S + R, Z + R)
\]

\[
(R, Z) := (R + Z, 0)
\]
The program \( \text{START} \rightarrow [S := R] \rightarrow \text{HALT} \)

to copy the contents of \( R \) to \( S \) can be implemented by

\[
\text{START} \rightarrow S^- \rightarrow R^- \rightarrow Z^- \rightarrow \text{HALT} \\
\quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
S^+ \quad Z^+ \quad R^+ \\
\quad \downarrow \quad \downarrow \quad \downarrow \\
\text{START} \rightarrow [S := R] \rightarrow \text{HALT}
\]

precondition:

\[
R = x \\
S = y \\
Z = 0
\]

postcondition:

\[
R = x \\
S = x \\
Z = 0
\]
The program \( \text{START} \rightarrow \begin{array}{c} \text{push } X \\ \text{to } L \end{array} \rightarrow \text{HALT} \)

\[ 2^X(2L + 1) \]

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

\[
\begin{align*}
\text{START} & \rightarrow Z^+ \rightarrow L^- \rightarrow Z^- \rightarrow X^- \rightarrow \text{HALT} \\
Z^+ & \rightarrow L^- \\
L^+ & \rightarrow Z^- \\
L^+ & \rightarrow X^- 
\end{align*}
\]
The program \[ \text{START} \rightarrow \boxed{\text{push } X \text{ to } L} \rightarrow \text{HALT} \]

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

\[(L, Z) := (2L + 1 + Z, 0)\]
The program \( \text{START} \rightarrow \text{push } x \rightarrow \text{L} \rightarrow \text{HALT} \)

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

\[
\text{START} \rightarrow Z^+ \rightarrow L^- \rightarrow Z^- \rightarrow X^- \rightarrow \text{HALT}
\]

\((L, Z) := (2L + Z, 0)\)
The program $\text{START} \rightarrow \begin{array}{c} \text{push} \\ X \\ \text{to} \\ L \end{array} \rightarrow \text{HALT}$

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

$\begin{array}{c}
\text{START} \rightarrow Z^+ \\ Z^+ \rightarrow L^- \\ L^- \rightarrow Z^- \\ Z^- \rightarrow X^- \\ X^- \rightarrow \text{HALT}
\end{array}$

precondition:
$X = x$
$L = \ell$
$Z = 0$

postcondition:
$X = 0$
$L = \langle x, \ell \rangle = 2^x \cdot (2\ell + 1)$
$Z = 0$
The program \( \text{START} \rightarrow \) \( \text{pop } L \rightarrow \text{HALT} \) specified by

\[
\text{“if } L = 0 \text{ then } (X ::= 0; \text{ goto EXIT) else let } L = \langle x, \ell \rangle \text{ in } (X ::= x; L ::= \ell; \text{ goto HALT)“}
\]

can be implemented by
START

$X^-$ \rightarrow L^- \rightarrow L^+ \rightarrow L^- \rightarrow Z^- \rightarrow Z^-$

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 $O$

HALT

EXIT
\{ 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 $\text{START} \rightarrow \begin{array}{c} \text{pop} \quad L \\ \text{to} \quad X \\ \rightarrow \text{HALT} \end{array} \rightarrow \text{EXIT}$ specified by

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

can be implemented by

\[
\begin{array}{cccccc}
\text{START} & \overrightarrow{X^-} & \overrightarrow{L^-} & \overrightarrow{L^+} & \overrightarrow{Z^-} & \overrightarrow{Z^-} \\
& \downarrow & \downarrow & \downarrow & \downarrow & \\
\overset{\text{EXIT}}{X^-} & L^- & L^+ & Z^- & \\
& \uparrow & \uparrow & \uparrow & \\
& \overset{\text{EXIT}}{L^-} & L^+ & Z^+ & L^+ & \end{array}
\]
Overall structure of \( U \)'s program

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

2. if \( N = 0 \) then copy 0\textsuperscript{th} item of list in \( A \) to \( R_0 \) and halt, else (decode \( N \) as \( \langle y, z \rangle \); \( C \leftarrow y \); \( N \leftarrow z \); 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 \( i \)\textsuperscript{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 $U$

START

`push 0 to A`

`T ::= P`

`pop T to N`

`PC ->`

`pop S to R`

`push R to A`

`push A to R`

`PC ::= N`

`R + <- C -`

`pop N to C`

`pop A to R0`

`push R to S`
The program for $U$

- **START**
  - $push\ 0\ to\ A$
  - $T::=P$
  - $pop\ to\ N$
  - $PC^-$

- **HALT**
  - $pop\ A\ to\ R_0$
  - $pop\ N\ to\ C$
  - $pop\ A\ to\ R$
  - $push\ R\ to\ S$

- **Flow**
  - $R^-$
  - $pop\ N\ to\ PC$
  - $N^+$
  - $C^-$

$T=0$
The program for $U$

START

1

$T := P$

push 0 to A

pop T to N

PC

pop A to R0

pop N to C

HALT

2

$T = 0$

pop A to R

push R to S

push R to N

R

pop S to R

push R to A

PC := N

R

pop N to PC

R

pop A to R

R

push to A

push to A

R

pop to A

R

pop to A

R

pop to A

R
The program for $U$

1. START
   - push 0 to A
   - T ::= P
   - pop T to N
   - PC

2. HALT
   - pop A to $R_0$
   - pop N to C

3. C even
   - pop R to S
   - R
   - PC ::=: N
   - R
   - N

4. C odd
   - push R to S
   - C

$T = 0$
The program for $U$

START

1

$push \ 0 \ to \ A$

2

$pop \ T \ to \ N$

$PC^{-}$

3

$pop \ S \ to \ R$

$R^{-}$

4

push $R$ to $A$

$R^{+}$

$pop \ R \ to \ A$

$C^{+}$

$PC := N$

$pop \ N \ to \ PC$

$N^{-}$

$pop \ N \ to \ C$

$C^{-}$

$R^{+}$

$C^{+}$

$R^{-}$

$C^{-}$

$pop \ A \ to \ R$

$push \ R$ to $S$

$T = 0$