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 $PC >$ 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

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

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

```
START → S^- → R^- → Z^- → HALT
```

$\text{START}$

$S^-$

$R^-$

$Z^-$

$HALT$

$Z^+$

$R^+$

$S^+$
The program \( \text{START} \rightarrow [S ::= R] \rightarrow \text{HALT} \)

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

\[
\text{START} \rightarrow \text{S}^{-} \rightarrow \text{R}^{-} \rightarrow \text{Z}^{-} \rightarrow \text{HALT}
\]

\( S := 0 \)
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 $\text{START} \rightarrow S := R \rightarrow \text{HALT}$

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

$S := 0$

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

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

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

precondition:
$R = x$
$S = y$
$Z = 0$

postcondition:
$R = x$
$S = x$
$Z = 0$
The program \( \text{START} \rightarrow \text{push} \ X \rightarrow L \rightarrow \text{HALT} \) to carry out the assignment \( (X, L) ::= (0, X :: L) \) can be implemented by

\[
\text{START} \rightarrow Z^+ \rightarrow L^- \rightarrow Z^- \rightarrow X^- \rightarrow \text{HALT}
\]
The program $\text{START} \rightarrow \text{push } X \to L \rightarrow \text{HALT}$

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

$(L, Z) := (2L + 1 + Z, 0)$
The program \[ \text{START} \rightarrow \begin{array}{c} \text{push} \\ X \\ \text{to} \\ L \end{array} \rightarrow \text{HALT} \]

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

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

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

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

precondition:
\[
\begin{align*}
X &= x \\
L &= \ell \\
Z &= 0
\end{align*}
\]

postcondition:
\[
\begin{align*}
X &= 0 \\
L &= \langle x, \ell \rangle = 2^x(2\ell + 1) \\
Z &= 0
\end{align*}
\]
The program specified by

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

can be implemented by
START

\[ X^- \rightarrow L^- \rightarrow L^+ \rightarrow L^- \rightarrow Z^- \rightarrow Z^- \]

EXIT

\[ X^+ \rightarrow L^+ \rightarrow Z^+ \rightarrow L^+ \rightarrow E \]

HALT

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
\{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 \( \text{START} \rightarrow \text{pop } L \rightarrow \text{HALT} \) specified by

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

can be implemented by
Overall structure of \textbf{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 \textcolor{red}{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 \textcolor{red}{3})

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

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

4. execute current instruction on \texttt{R}; update \texttt{PC} to next label; restore register values to \texttt{A}; goto \textcolor{red}{1}
The program for $U$

START

`push 0 to A`

`T ::= P`

`pop T to N`

`PC -`

`pop N to C`

`pop A to R_0`

`pop N to S`

`pop A to R`

`push R to S`

`R -`

`pop N to PC`

`R +`

`C -`

`pop A to R`
The program for $U$

1. **START**
   - push 0 to A

2. **T := P**

3. **pop T to N**

4. **PC := N**
   - $T = 0$
   - $T = 0$

5. **pop A to R₀**

6. **pop N to C**

7. **pop A to R**

8. **pop S to R**

9. **push R to A**

10. **PC := N**

11. **R := N**

12. **N := C**

13. **push R to S**
The program for $U$

START

$push \ 0 \ to \ A$

$T ::= P$

$pop \ T \ to \ N$

$PC^-$

$R^-$

$PC ::= N$

$R^+$

$C^-$

$pop \ N \ to \ PC$

$L4 49$

$N^+$

$C^-$

end

HALT

$pop \ A \ to \ R_0$

$pop \ A \ to \ S$

$pop \ N \ to \ C$

$pop \ A \ to \ R$

$pop \ A \ to \ R$
The program for U

START

push 0 to A

T ::= P

pop T to N

PC⁻

pop S to R

push R to A

PC ::= N

R⁺

C even

R⁻

pop N to PC

N⁺

C odd

R⁺

pop N to C

pop to C

pop to A

pop to R₀

HALT
The program for $U$

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

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

3. 
   - $pop \ N$ to $PC$
   - $N^+$
   - $C^-$
   - $push \ R$ to $S$

4. 
   - $pop \ S$ to $R$
   - $push \ R$ to $A$
   - $PC ::= N$
   - $R^+$
   - $C^-$
   - $C$ even

5. 
   - $R^-$
   - $pop \ N$ to $PC$
   - $N^+$
   - $C^-$
   - $C$ odd