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.
- $R_0$ result of the simulated RM computation (if any).
Overall structure of \textit{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 halt, else decode \( N \) as \( \langle y, z \rangle \); \( C \coloneqq y \); \( N \coloneqq z \); 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 PCth item of list in P to N (halting if PC > length of list); goto 2

2. if N = 0 then 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 $z = \langle j, k \rangle$\}\n
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

To implement this, we need RMs for manipulating (codes of) lists of numbers...
The program \texttt{START$\rightarrow$S ::= R$\rightarrow$HALT} to copy the contents of \texttt{R} to \texttt{S} can be implemented by

\[
\begin{align*}
\texttt{START} & \rightarrow \texttt{S}^- \rightarrow \texttt{R}^- \rightarrow \texttt{Z}^- \rightarrow \texttt{HALT} \\
\texttt{Z}^+ & \rightarrow \texttt{R}^+ \\
\texttt{S}^+ & \rightarrow \texttt{Z}^+ \\
\texttt{R}^+ & \rightarrow \texttt{S}^+ 
\end{align*}
\]
The program \texttt{START}→\texttt{S ::= R}→\texttt{HALT} to copy the contents of \texttt{R} to \texttt{S} can be implemented by

\begin{center}
\begin{tikzcd}
\texttt{START} & \texttt{S}^- & \texttt{R}^- & \texttt{Z}^- & \texttt{HALT} \\
\texttt{S} := 0 & \texttt{Z}^+ & \texttt{R}^+ & \texttt{S}^+
\end{tikzcd}
\end{center}
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

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

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

$S := 0$

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

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

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

precondition:

\[
\begin{align*}
R &= x \\
S &= y \\
Z &= 0
\end{align*}
\]

postcondition:

\[
\begin{align*}
R &= x \\
S &= x \\
Z &= 0
\end{align*}
\]
The program $\text{START} \rightarrow \text{push } X \text{ to } L \rightarrow \text{HALT}$

$2^X(2L + 1)$

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

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

$Z^+ \rightarrow Z^-$

$L^+ \rightarrow L^-$
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

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

$(L, Z) ::= (2L+1+Z, 0)$
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 \( \text{START} \rightarrow \text{push} \, X \rightarrow L \rightarrow \text{HALT} \)

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

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

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

\[
\text{START} \xrightarrow{\text{pop } L} \text{to } X \xrightarrow{} \text{HALT} \xrightarrow{} \text{EXIT}
\]

"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
If $Z+L$ even then

$(Z,L) := (0, \frac{1}{2}(Z+L)) \& \text{go to } E$

Else

$(Z,L) := (0, \frac{1}{2}(Z+L-1)) \& \text{go to } O$
{assuming \( z=0, L \geq 0 \)}

(while \( L \) even do

\[
L := \frac{1}{2} L; \quad X := X + 1;
\]

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

if \( z+L \) even then

\[
(\bar{z}, L) := (0, \frac{1}{2} (z+L)) & \text{goto } E
\]

else

\[
(\bar{z}, L) := (0, \frac{1}{2} (z+L-1)) & \text{goto } O
\]
The program specified by

\[
\text{START} \rightarrow \text{pop } L \rightarrow X \rightarrow \text{HALT} \\
\rightarrow \text{EXIT}
\]

“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 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 halt, else decode N as ⟨⟨y, z⟩⟩; C ::= y; N ::= z; goto 3

   {at this point either C = 2i is even and current instruction is R_i^+ → L_z, or C = 2i + 1 is odd and current instruction is R_i^- → L_j, L_k where z = ⟨j, k⟩}

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 $U$

START ----------------- T ::= P ----------------- pop $^T$ to $^N$ ----------------- HALT

pop $^S$ to $^R$ ----------------- push $^R$ to $^A$ ----------------- PC ::= $^N$

PC$^-$ ----------------- R$^+$ ----------------- C$^-$

R$^-$ ----------------- pop $^N$ to PC ----------------- N$^+$

N$^+$ ----------------- C$^-$

C$^-$ ----------------- push $^R$ to $^S$
```
The program for $U$

START $\rightarrow$ $T ::= P$

$T$ to $N$

$T = 0$

$N$ to $C$

$PC^-$

$N$ to $R$

$R$ to $S$

$S$ to $R$

$R$ to $A$

$A$ to $PC$

$PC ::= N$

$R^+$

$C^-$

$N^+$

$C^-$

$R^-$

$N$ to $PC$

$A$ to $R$

$R$ to $S$

$Computation Theory, L 4 57/170
The program for $U$

1. START
   - $T ::= P$
   - $pop_T to N$
   - $PC^-$

2. HALT
   - $pop_N to C$

- $pop_S to R$
- $push_R to A$
- $PC ::= N$
- $R^+$
- $C^-$

- $R^-$
- $pop_N to PC$
- $N^+$
- $C^-$

- $push_R to S$
The program for \( U \)

1. \text{START} \rightarrow \text{T} ::= \text{P} \rightarrow \text{pop} _{\text{T}} \text{to N} \rightarrow \text{HALT}

2. \text{PC} ::= \text{N} \rightarrow \text{R}^{+} \rightarrow \text{C}^{-} \rightarrow \text{push} _{\text{R}} \text{to S}

3. \text{R}^{-} \rightarrow \text{pop} _{\text{N}} \text{to PC} \rightarrow \text{N}^{+} \rightarrow \text{C}^{-} \rightarrow \text{push} _{\text{R}} \text{to S}

\text{Computation Theory, L 4}
The program for $U$:

1. START \[ T ::= P \]
2. pop $T$ to $N$
3. pop $N$ to $C$
4. pop $S$ to $R$
5. push $R$ to $A$
6. PC $::= N$
7. $R^+ \leftarrow C^-$
8. $N^+ \leftarrow C^-$
9. pop $N$ to PC
10. $R^- \leftarrow C^-$
11. push $R$ to $S$
12. $C^{\text{even}}$
13. $C^{\text{odd}}$
14. HALT

Computation Theory, L 4 57/170