Universal register machine, $U$
Audience asked: what happens (on Slide 42) if \( e = \text{"U"} \) and \( a = 0 \) ?

Universal register machine, \( U \)

Answer: \( U \) runs \( U \) with \( R_1 = R_2 = 0 \) and since \( 0 = \text{"empty program"} \) (and empty program immediately halts) it then halts with \( R_0 \) still = 0

So we compute the constantly-zero function in this case.
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)

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 $\langle y, z \rangle$; $C ::= y$; $N ::= z$; 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 \( \text{START} \rightarrow \boxed{S ::= R} \rightarrow \text{HALT} \) to copy the contents of \( R \) to \( S \) can be implemented by
The program \texttt{START}→\texttt{S ::= R}→\texttt{HALT}

to copy the contents of \texttt{R} to \texttt{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

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

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

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

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

precondition:

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

postcondition:

$R = x$
$S = x$
$Z = 0$
The program $\text{START} \rightarrow \text{push } X \text{ to } L \rightarrow \text{HALT}$

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

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

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

\[(L, Z) := (2(L+1)+Z, 0)\]
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

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

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

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

precondition:
\[
X = x \\
L = \ell \\
Z = 0
\]

postcondition:
\[
X = 0 \\
L = \langle x, \ell \rangle = 2^x(2\ell + 1) \\
Z = 0
\]
The program \( \text{START} \rightarrow \text{pop} \ L \to X \rightarrow \text{HALT} \rightarrow \text{EXIT} \) 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
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 \ 0
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
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
The program for $U$

START

$\text{push } 0 \text{ to } A$

$T ::= P$

$\text{pop } T \text{ to } N$

$\text{PC}^-$

$\text{pop } A \text{ to } R_0$

$\text{pop } N \text{ to } C$

$\text{pop } S \text{ to } R$

$\text{push } R \text{ to } A$

$\text{PC} ::= N$

$R^+$

$\text{pop } A \text{ to } R$

$\text{push } R \text{ to } S$

$N^+ \leftarrow C^-$

$R^- \leftarrow \text{pop } R \text{ to } PC$

$N^- \leftarrow \text{pop } N \text{ to } R$

$C^- \leftarrow \text{pop } N \text{ to } C$

HALT
The program for $U$

- **START**
  - push 0 to A
  - pop S to R
  - push R to A
  - PC := N

- **pop to T**
  - T := P
  - pop to N
  - PC := N

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

- $T = 0$
The program for $U$

START

$push\ 0$ to $A$

$T::=P$

$pop\ T$ to $N$

$PC^-$

$T=0$

HALT

$pop\ A$ to $R_0$

$pop\ N$ to $C$

$pop\ A$ to $R$

$push\ R$ to $S$

$pop\ N$ to $PC$

$N^+$

$C^-$

$R^+$

$C^-$

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

START

1

push 0 to A

T ::= P

pop T to N

PC-

pop S to R

push R to A

push R to A

PC ::= N

R+

pop N to PC

R-

C even

N+

C odd

push R to S

HALT

2

pop A to R

C

pop N to C

3

push R to S
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 $C$
   - $C \text{ even}$
   - $pop\ A$ to R
   - $push\ R$ to S

3. $R^+$

4. $N^+$

5. $C^-$

6. $3$

- $C \text{ odd}$

- $4$

- $L4$