Interview 1 Answers

The following answers are **not** model answers, they are just one possible answer.

These answers do not make use of register machines from previous questions, however answers that do are equally valid.

PS: Register machines are analogous to Turing machines: a mathematical model of computation. If you are interested in reading more, you can read the Cambridge University lecture notes on register machines [here].

Below we use the notation $R_n := m$ to represent setting the register R_n to the value m, and the notation $R_n := R_m$ to represent setting the register R_n to the value stored in register R_m.

$(R_n := R_m), (R_m := 0)$ represents setting register R_n to the value of register R_m, and subsequently setting the register R_m to the value 0.

Introduction

Can you informally reason about why this machine represents the function $f(x) = x$ for all x?

This machine works by incrementing R_0 each time we decrement R_1. The machine only ends once R_1 reaches 0. To reach this point, we must decrement from R_1 x number of times, and so $R_0 = x$ at the end of the program.

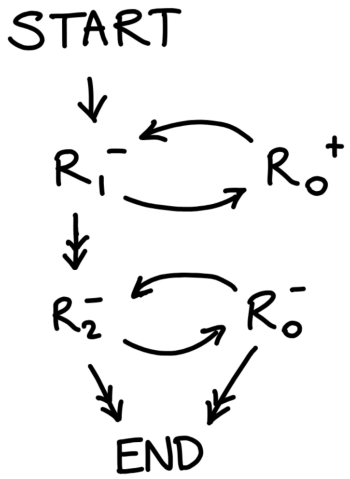
What function does the following two-argument register machine represent?

$f(x, y) = x + y$

We first perform $R_0 := R_0 + x$, followed by $R_0 := R_0 + y$. Since R_0 starts at 0, the end result must be $R_0 = x + y$.

Question 1

The idea here is to set $R_0 := R_1$, and keep subtracting R_2 from R_0 until either R_2 or R_0 reaches 0.



Question 2

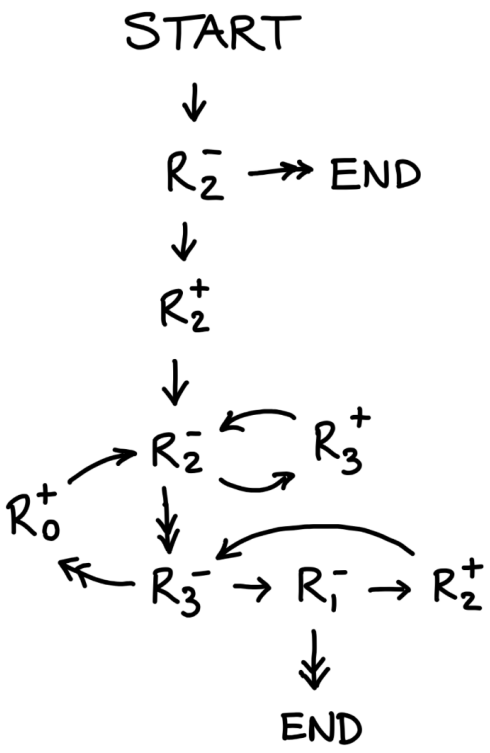
The idea here is to keep subtracting R_2 from R_1 until R_1 reaches zero. We will keep track of the number of complete subtractions in the result register R_0.

If R_2 is 0 we want to immediately exit with $R_0 = 0$.

Else we want to count the number of times we can subtract R_2 from R_1. We do this by introducing a dummy register R_3, and setting $(R_3 := R_2), (R_2 := 0)$ so that now R_3 contains y, and R_2 contains 0.

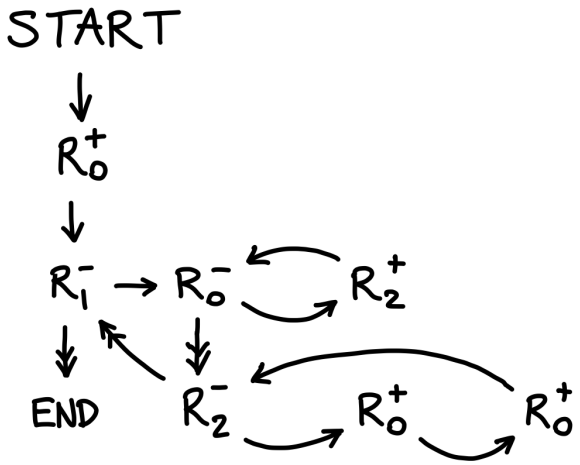
We then subtract R_3 from R_1 (using similar ideas to Question 1), and for each decrement of R_1 we increment R_2. Hence, when R_3 reaches 0, R_2 now contains y. By swapping R_2 and R_3, we reset the program state so that once again R_3 contains y, and R_2 contains 0.

We can then continue subtracting y from R_1; each time we complete a full subtraction we increment R_0 by one. Once R_1 reaches 0, we exit the program.



Question 3

This machine works by setting $R_0 := 1$ and then doubling this register R_1 number of times. The doubling works by setting $(R_2 := R_0), (R_0 := 0)$, and then adding R_2 back to R_0 twice. Hence, at the end of the loop, $(R_0 := R_2 + R_2), (R_2 := 0)$.



Question 4

The idea is to extend our answer to question 2, by counting the number of times we can divide R_1 by 2.

We first set $R_2 := 2$, to act as our logarithm base. To do this, we clear the value of R_2 with continuous decrements until it reaches 0, and then increment R_2 twice (clearing R_2 only becomes relevant once we loop back around to these nodes).

We then divide R_1 by R_2 and keep the result in R_4. If R_4 is 0 then no more divisions of R_1 can be made and so we immediately exit. Else, we set $(R_1 := R_4), (R_4 := 0), (R_3 := 0)$ to set R_1 to its halved value, and reset R_4 and R_3 back to 0.

We then increment R_0 by 1 as R_1 has been halved, and repeat this loop.