Definition 2: Register machine computation

A **computation** of a register machine is a (finite or infinite) sequence of configurations

$$c_0, c_1, c_2, \ldots$$

where

- c_0 is an initial configuration,
- each $c = (\ell, r_0, \ldots, r_n)$ in the sequence determines the next configuration in the sequence (if any) by carrying out the program instruction labelled L_ℓ with registers containing $r_0, \ldots, r_n$.

Halting For a finite computation $c_0, c_1, \ldots, c_m$, the last configuration $c_m = (\ell, r_0, \ldots)$ must be a *halting configuration*, i.e. ℓ must satisfy:

either: the ℓ^{th} instruction in the program has the body `HALT` (a "proper halt");

or: ℓ is greater than the number of instructions in the program, so that there is no instruction labelled L_ℓ (an "erroneous halt").

N.B. a program can always be modified (without affecting its computations) to turn all erroneous halts into proper halts by adding extra `HALT` instructions to the list with appropriate labels.

Note that computations may never halt. For example, the following register machine with one register R_0 has only infinite computation sequences of the form $(0, r), (0, r+1), (0, r+2), \ldots$.

$$\begin{array}{lcl} L_0 & : & R_0^+ \to L_0 \\ L_1 & : & \texttt{HALT} \end{array}$$

2.2 Graphical Representation

A register machine can be represented by a graph with one node (vertex) for each instruction. The arcs (edges) represent jumps between instructions and thereby replace the labels. Because the sequential ordering of instructions is lost, we need to indicate the initial instruction with `START`.

program code	graphical representation
$R^+ \to L$	$R^+ \to [L]$
$R^- \to L, L'$	$R^- \begin{array}{l} \nearrow [L] \\ \searrow [L'] \end{array}$
HALT	HALT
L_0	START $\longrightarrow [L_0]$