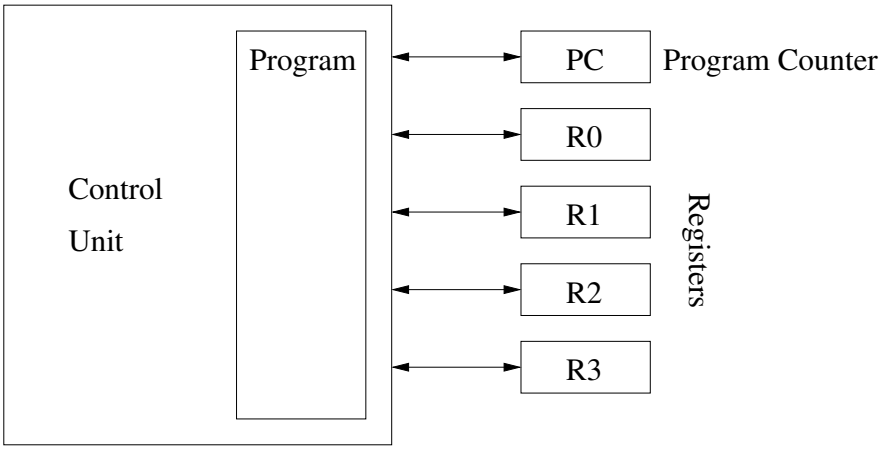
Random Access Machines

A random access machine (RAM) is a simple *model of computation*. Its memory consists of an unbounded sequence of registers. Each of the registers may hold an integer value.

The *control unit* of a RAM holds a *program*, i.e. a numbered list of statements. The *program counter* determines which statement is to be executed next.



Rules for executing a RAM-program:

- in each work cycle the RAM executes one statement of the program;
- the program counter specifies the number of the statement that is to be executed;
- the program ends when the program counter takes an invalid value (i.e. there's no statement in the program that has the specified number)

To "run" a program in the RAM, we need to:

- define the program, i.e. the exact list of statements;
- define starting values for the registers (the *input*);
- define starting values for the program counter (usually, we'll start with the first statement);

Statements of a RAM

Notation:

`<Ri>`	$\widehat{=}$	the integer stored in the `i`-th register
`<Ri> := x`	$\widehat{=}$	let integer `x` be the content of the `i`-th register

List of Statements:

Statement	Effect on registers	Program Counter
`Ri` $\leftarrow$ `Rj`	`<Ri>` := `<Rj>`	`<PC>` := `<PC> + 1`
`Ri` $\leftarrow$ `RRj`	`<Ri>` := `<R<Rj>>`	`<PC>` := `<PC> + 1`
`RRi` $\leftarrow$ `Rj`	`<R<Ri>>` := `<Rj>`	`<PC>` := `<PC> + 1`
`Ri` $\leftarrow$ `k`	`<Ri>` := `k`	`<PC>` := `<PC> + 1`
`Ri` $\leftarrow$ `Rj + Rk`	`<Ri>` := `<Rj> + <Rk>`	`<PC>` := `<PC> + 1`
`Ri` $\leftarrow$ `Rj - Rk`	`<Ri>` := $\max\{0,$`<Rj> - <Rk>`$\}$	`<PC>` := `<PC> + 1`
`GOTO m`		`<PC>` := `m`
`IF Ri=0 GOTO m`		`<PC>` := $\begin{cases} \texttt{m} & \texttt{if <Ri>} = 0 \\ \texttt{<PC> + 1} & \texttt{otherwise.} \end{cases}$
`IF Ri>0 GOTO m`		`<PC>` := $\begin{cases} \texttt{m} & \texttt{if <Ri>} > 0 \\ \texttt{<PC> + 1} & \texttt{otherwise.} \end{cases}$