Random Access Machines (RAMs)

Variants of RAMs exist. We describe the version in *The Design and Analysis of Computer Algorithms*, Aho, Hopcroft, Ullman, Addison-Wesley, 1974.

A RAM has three parts:

- The memory consisting of infinitely many cells r_i, $i \geq 0$. The cell r_0 is the *accumulator* where all arithmetic and comparison operations are performed.
- The read-only input tape consisting of infinitely many cells, together with the read-head.
- The write-only output tape consisting of infinitely many cells, together with the write-head.
- The read/write tape cells and the memory cells r_i may contain integers only.
- Initially the memory cells r_i, $i \geq 0$, contain 0.

An instruction operand takes one of the two forms:

- $=i$, indicating integer literal i
- i, indicating the contents of memory cell r_i

Define v(a), the value of operand a, by:

- $v(=i) = i$
- $v(i)$ = the contents of r_i

The following is the instruction set.

instruction	operational semantics
Load a	$r_0 \leftarrow v(a)$
Store i	$r_i \leftarrow r_0$
Add a	$r_0 \leftarrow r_0 + v(a)$
Sub a	$r_0 \leftarrow r_0 - v(a)$
Mult a	$r_0 \leftarrow r_0 * v(a)$
Div a	$r_0 \leftarrow \lfloor r_0 / v(a) \rfloor$
Jump L	unconditionally jump to label L
Jgtz L	if $r_0 > 0$, jump to label L; otherwise go to the next instruction
Jzero L	if $r_0 = 0$, jump to label L; otherwise go to the next instruction
Read i	$r_i \leftarrow$ value of input tape cell pointed by read-head; read-head moves one position to right.
Write a	v(a) is written in output tape cell pointed by write-head; write-head moves one position to right.
Halt	halt execution

The following are an algorithm to compute the factorial function n!, $n \geq 1$, and a RAM program implementing it. The value of n is initially given on the input tape.

```
r1 ← n; // read n
r2 ← r1; // r2 accumulates the product value n(n-1)···(n-i).
r3 ← r1-1; // r3 is a counter for (n-i), to be decremented by 1 in each iteration.
while ( r3 > 0 )
{
```

```
    r2 ← r2*r3;
    r3 ← r3−1;
}
write r2;
```

```
          Read 1       // r1 ← n stored on the input tape
          Load 1       // r0 ← r1
          Store 2      // r2 ← r0, r2 = n
          Sub =1       // r0 ← r0−1
          Store 3      // r3 ← r0, r3 = n−1
while:    Load 3       // r0 ← r3
          Jgtz continue     // if r0 > 0 then jump to "continue"
          Jump endwhile     // jump to "endWhile"
continue: Load 2       // r0 ← r2
          Mult 3       // r0 ← r0*r3
          Store 2      // r2 ← r0
          Load 3       // r0 ← r3
          Sub =1       // r0 ← r0−1
          Store 3      // r3 ← r0
          Jump while   // jump to "while"
endwhile: Write 2      // write r2 on the output tape
          Halt
```