RAM - The Random Access Machine Model

Features:

- read-only input tape
- write-only output tape
- random access memory
- finite program
- each tape square and memory cell can hold arbitrarily large integer
- accumulator - register 0 - where arithmetic is performed
RAM instructions have a single operand, which can be

1. =i, meaning the integer i itself
2. i, meaning the contents of register i
3. *i, meaning indirect addressing (the contents of the register given by the contents of register i)

The RAM instruction set follows. We use the following notation:

\[ c(i) = \text{the contents of register } i. \]

The function \( v(a) \) is defined as follows:

\[
\begin{align*}
v(a) &= \frac{\text{b},}{\text{c(b)},} \quad \text{if } a = \text{"=b";} \\
v(a) &= \frac{\text{c(b)},}{\text{c(c(b))},} \quad \text{if } a = \text{"*b".}
\end{align*}
\]

I'm definitely NOT going to test you on the intricacies of the instructions in this particular RAM model. It's much more important for you to have a general idea of what can be done in a RAM and how it might be simulated by a TM.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. LOAD a</td>
<td>c(0) := v(a)</td>
</tr>
<tr>
<td>2. STORE i</td>
<td>c(i) := c(0)</td>
</tr>
<tr>
<td>STORE *i</td>
<td>c(c(i)) := c(0)</td>
</tr>
<tr>
<td>3. ADD a</td>
<td>c(0) := c(0) + v(a)</td>
</tr>
<tr>
<td>4. SUB a</td>
<td>c(0) := c(0) - v(a)</td>
</tr>
<tr>
<td>5. MULT a</td>
<td>c(0) := c(0) * v(a)</td>
</tr>
<tr>
<td>6. DIV a</td>
<td>c(0) := floor( c(0)/v(a) )</td>
</tr>
<tr>
<td>7. READ i</td>
<td>c(i) := current input symbol</td>
</tr>
<tr>
<td>READ *i</td>
<td>c(c(i)) := current input symbol</td>
</tr>
<tr>
<td>8. WRITE a</td>
<td>v(a) is written on output tape</td>
</tr>
<tr>
<td>9. JUMP b</td>
<td>location counter set to b</td>
</tr>
<tr>
<td>10. JUMP&gt;=0 b</td>
<td>if c(0) &gt;= 0, set location ctr to b; otherwise increment location ctr</td>
</tr>
<tr>
<td>11. JUMP&gt;0 b</td>
<td>if c(0) &gt; 0, set location ctr to b; otherwise increment location ctr</td>
</tr>
<tr>
<td>12. JUMP=0 b</td>
<td>if c(0)=0, set location ctr to b; otherwise increment location ctr</td>
</tr>
<tr>
<td>13. HALT</td>
<td>stop execution</td>
</tr>
</tbody>
</table>
A Sample RAM Program to Compute
  GCD(m, n)
Where m and n Are the Next Inputs

1. READ 1    /* set c(1) := m
2. READ 2    /* set c(2) := n
3. LOAD 2    /* set c(0) := n
4. STORE 3   /* set c(3) := n
5. JUMP=0 15 /* if n = 0 goto 15
6. LOAD 1    /* set c(0) := m
7. DIV 2     /* set c(0) := floor(m/n)
8. MULT 2    /* set c(0) := n*floor(m/n)
9. SUB 1     /* set c(0) := n*floor(m/n) - m
10. MULT=-1  /* set c(0) := m - n*floor(m/n)
11. STORE 2  /* set c(2) := m mod n
12. LOAD 3   /* set c(0) := n
13. STORE 1  /* set c(1) := n
14. JUMP 3   /* end of loop
15. WRITE 1  /* if n = 0 then gcd(m,n) = m
16. HALT
Simulating a RAM with a TM

1. READ 1 Δ 2 READ 2 Δ 3 LOAD
   RAM PROGRAM

2. 2 3 7 6 Δ 3 5 6 6 9 Δ ...
   INPUT TO RAM

3. # ...
   ACCUMULATOR

4. # 1: 2 3 7 6 Δ 2: 0 Δ 3: 3 5 6 6 9 ...
   MEMORY unordered

5. # ...
   OUTPUT

6. # 8 Δ ...
   PROGRAM COUNTER

7. # ...
   ADDRESS FOR INDIRECT ADDRESSING