# Turing Machines

- In 1936, A.M. Turing proposed the Turing machine as a model of *any possible computation*.
- This model was *built* using electro-mechanical devices after several years.
- Turing machine long has been recognized as an accurate model for what any physical computing device is capable of doing.

# Turing Machines

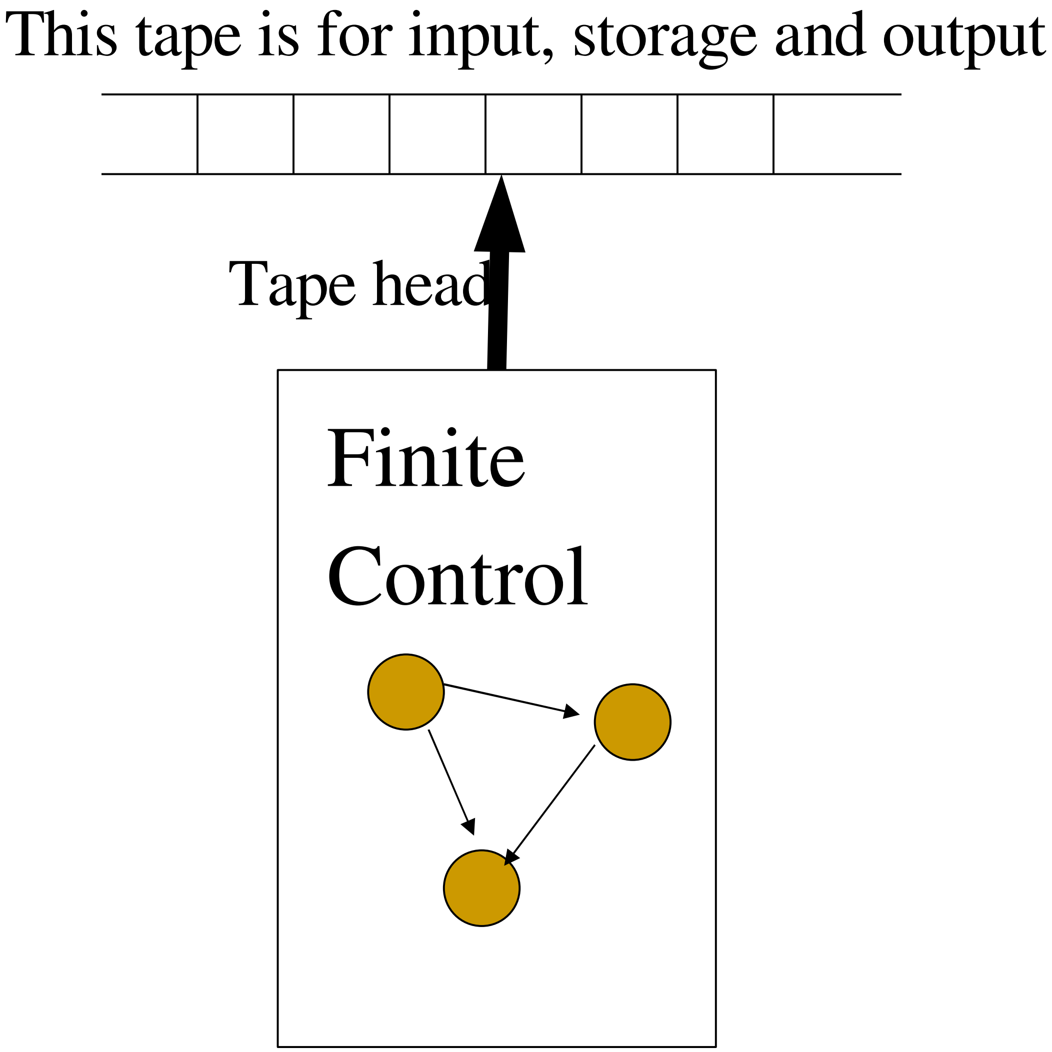
Turing formalized the idea of “mechanical procedure”:

- Can be described by a finite number of instructions.
- Instructions are simple and mechanical.
- Have a finite number of internal states.
- Can deal with input not restricted in size.
- Have an unlimited storage space for calculations.
- Can produce output of unlimited size.

# Turing Machine

- Not a real machine, but a model of computation
- Components:
  - 1-way infinite tape: unlimited memory
  - Store input, output, and intermediate results
  - Infinite cells
  - Each cell has a symbol from a finite alphabet
- Tape head:
  - Point to one cell
  - Read or write a symbol to that cell
  - move left or right

# Turing Machines



**Definition: A Turing Machine is a 7-tuple**

**$T = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:**

**Q** is a finite set of states

**(** $\Sigma$ is the input alphabet, where $\square \notin \Sigma$ ($\square$ blank))

$\Gamma$ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$

$q_0 \in Q$ is the start state

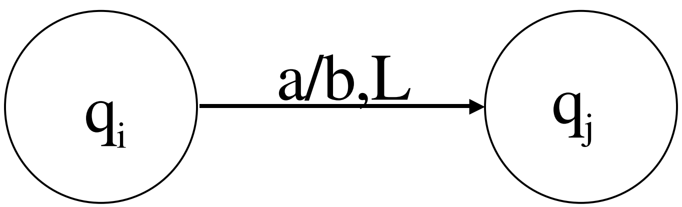
**$q_{accept} \in Q$ is the accept state**

**$q_{reject} \in Q$ is the reject state, and $q_{reject} \neq q_{accept}$**

# Turing Machines

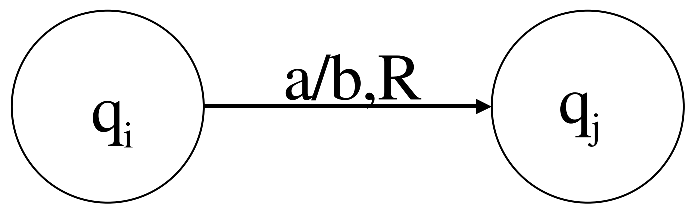
$\delta$ is a move function mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L,R\}$

Replace the current symbol "a" by "b", and move to the left.




$\delta(q_i, a) = (q_j, b, L)$

Replace the current symbol "a" by "b", and move to the right.




$\delta(q_i, a) = (q_j, b, R)$

Note that if a = b, we write "a,L" instead of "a/a,L" along the edge.

# A TM for **`Successor`** Program

Sample Rules:

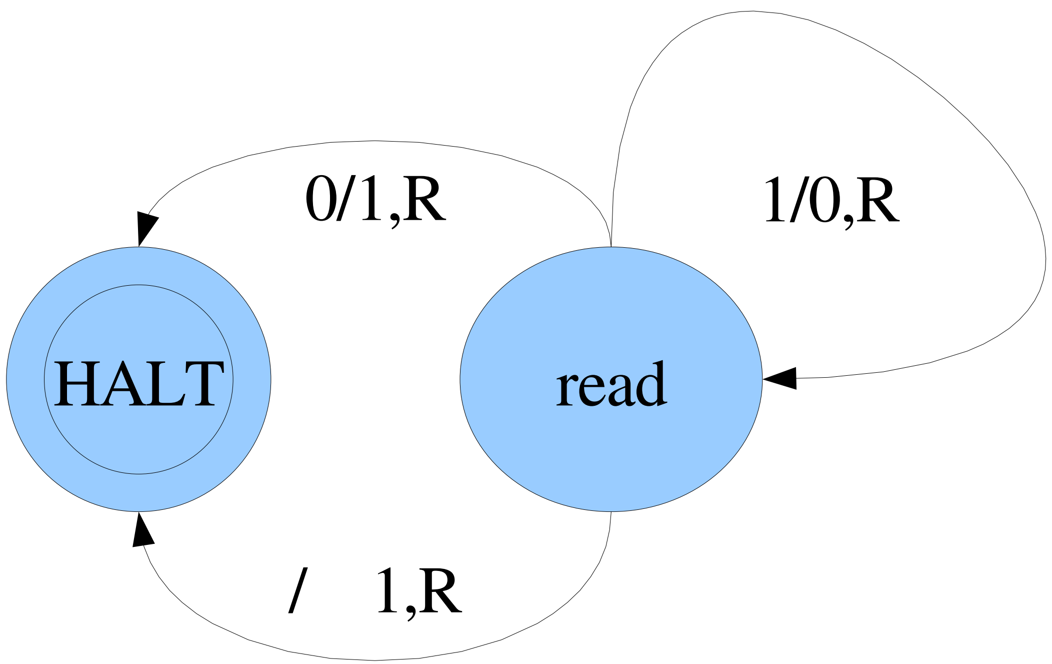
If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read   , write 1, HALT!

Let’s see how they are carried out on a piece of paper that contains the *reverse* binary representation of 47:

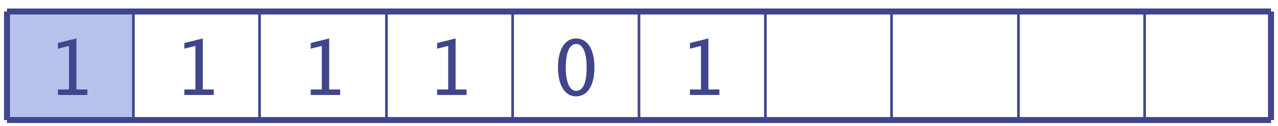
# TM successor



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

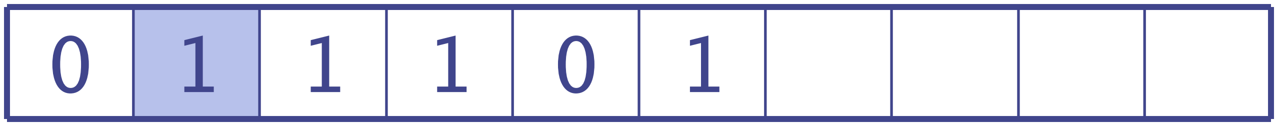
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

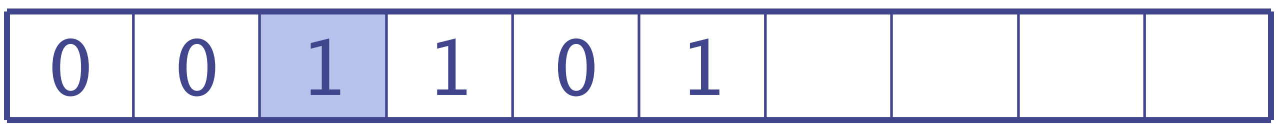
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

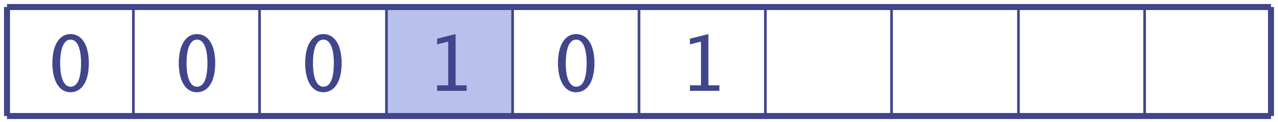
If read , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

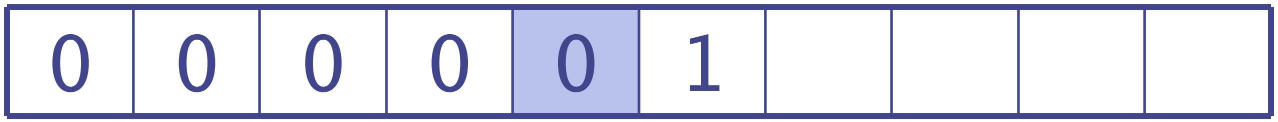
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read    , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read   , write 1, HALT!

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | | | | |

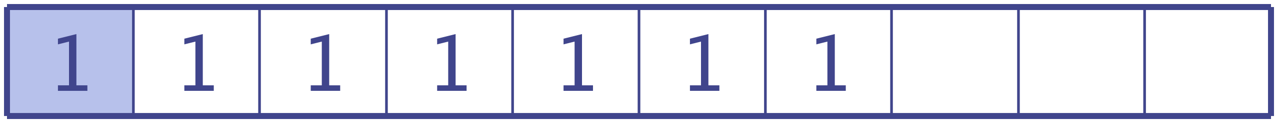
So the successor's output on 111101 was 000011 which is the reverse binary representation of 48.

Similarly, the successor of 127 should be 128:

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

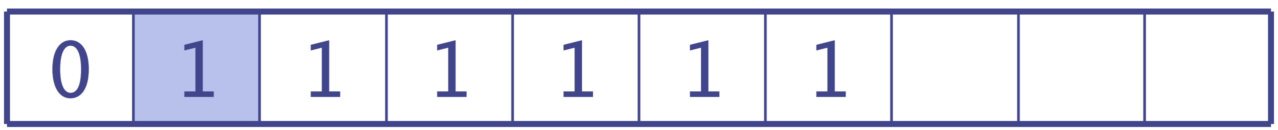
If read  , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

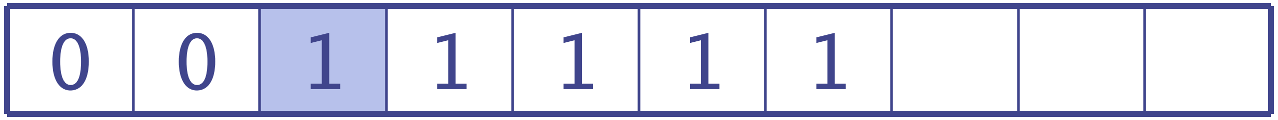
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

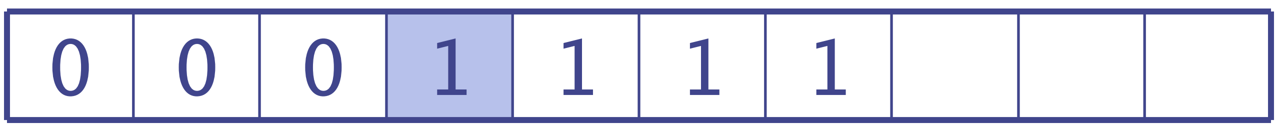
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

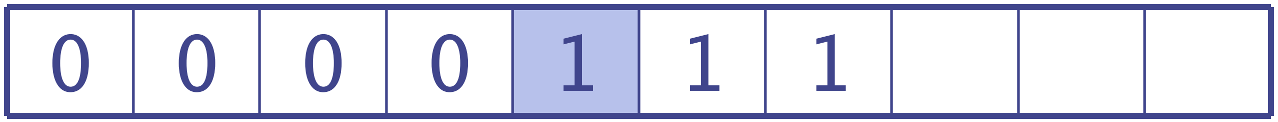
If read , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

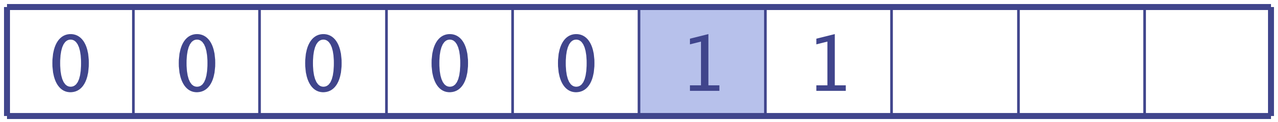
If read   , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

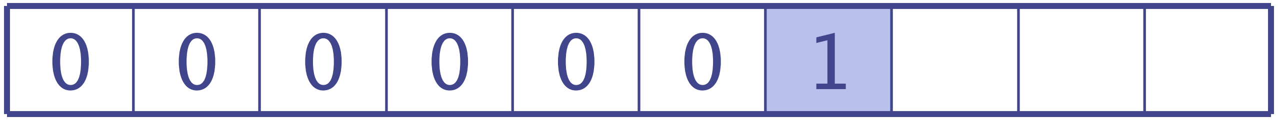
If read  , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

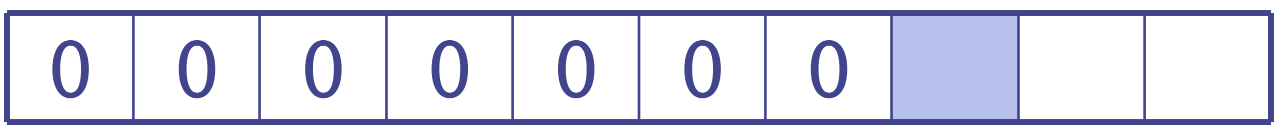
If read  , write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read   , write 1, HALT!



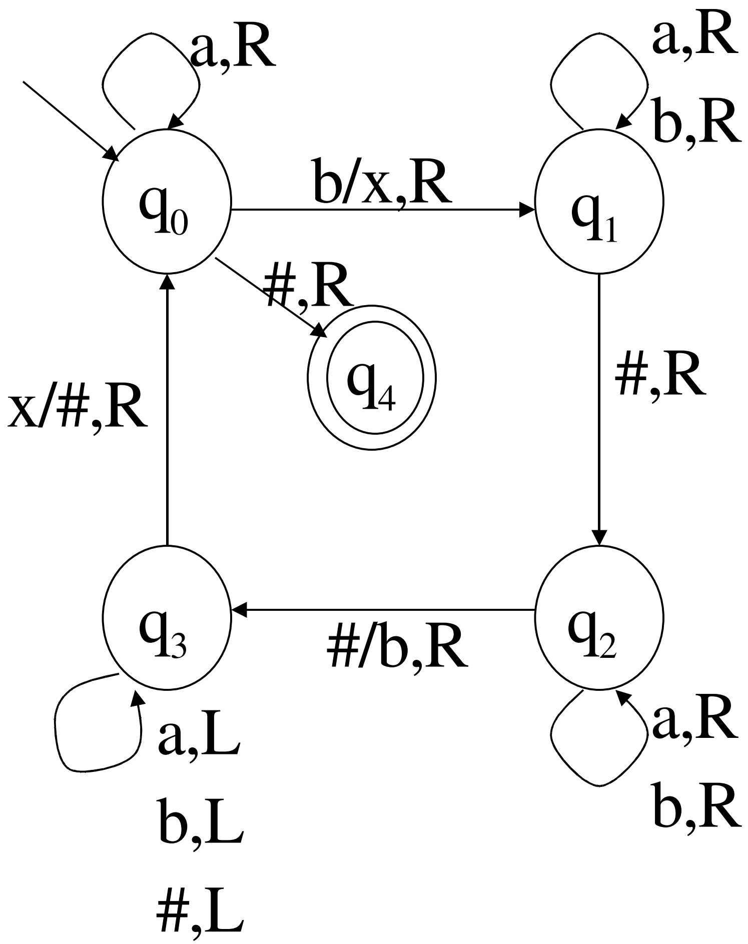
If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read  , write 1, HALT!

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | |

# An Example




$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Gamma = \{a, b, x, \#\}$

$B = \#$

$\Sigma = \{a, b\}$

$q_0$ is the start state

$F = \{q_4\}$

Consider the input:

baaba

What is the output?

# Turing Machines

- By using this powerful but simple model, we can start looking at the question of what languages can be defined (equivalently, what problems can be solved) by a computational device. Is there any problem that a computer cannot solve, and what are they?
- We will see in some later classes that there are a lot of them, and they are called “undecidable” problems.

# Other Examples

TM can do all sorts of things. Try the followings:

- Add 1 to a unary number. (Easy)
- Add 1 to a binary number.
- Convert a unary number to binary, and vice versa.
- Compare two unary numbers.
- Add two unary numbers.
- Compare two binary numbers x and y. If $x > y$, output 1. Otherwise, output 0.
- ……

# TM Simulator

- http://www.igs.net/~tril/tm/tm.html
- http://www.cheransoft.com/vturing/download.html
- chi scrive un simulatore salta la parte teorica sulle TM