Finite State Automata with Output

Finite automata (FA) may have outputs corresponding to each transition. There are two types of **Finite State Machines (FSM)** that generate output.

- **Moore Machine**
- **Mealy Machine**

Note: Till now we have seen automata with no output those are language accepters.
Note: Now we will see language transducers means they will convert input to output.

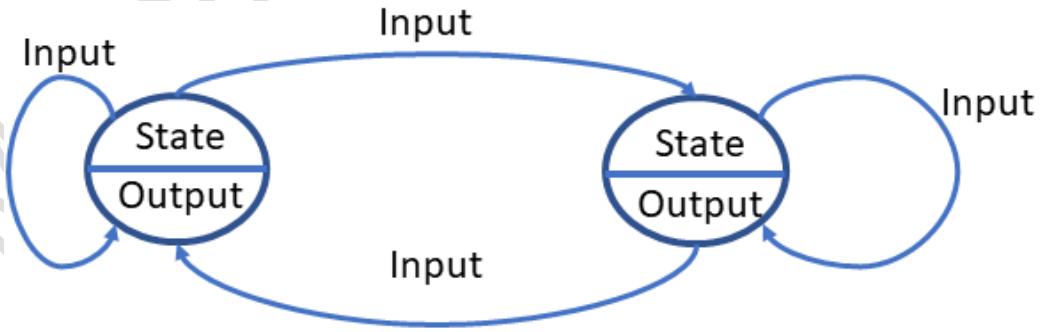
Moore Machine

Moore machine is a finite state machine in which the next state is decided by the current state and current input symbol. The output symbol at a given time depends only on the present state of the machine. Moore machine shows output on state itself so if input ε then also we will get one output. It is very similar to a **Finite Automaton (FA)**, with a few key differences:

- It has **no final states**.
- It does not accept or reject input string, instead, it **generates output from input**.
- Moore machines **cannot have nondeterministic states**.

Moore machine can be described by 6 parameters/tuples $(Q, \sum, q_0, \Delta, \lambda, \delta)$ where,

Parameter Name	Description
Q	is a finite set of states
Σ	is a finite set of input symbols called the input alphabet
q_0	is the initial state from where any input is processed ($q_0 \in Q$)
Δ	is a finite set of symbols called the output alphabet
λ	is the output transition function where $\lambda: Q \rightarrow \Delta$
δ	is the input transition function where $\delta: Q \times \sum \rightarrow Q$



Generic Moore model

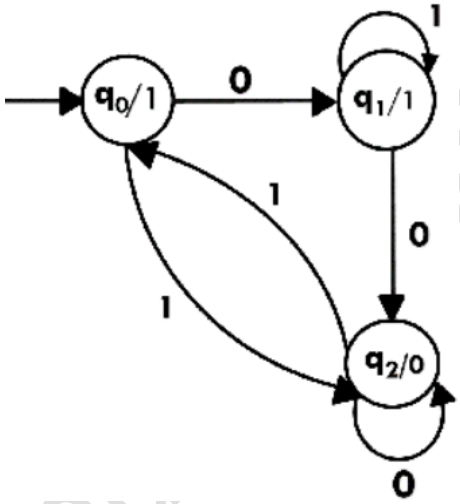
Examples of Moore Machine

Example 1:

The state table of a Moore Machine is shown below:

Current State	Next State (δ)		Output (λ)
	0	**1**	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

The state diagram of the above Moore Machine is:



In the above Moore machine, the output is represented with each input state separated by /. The output length for a Moore machine is greater than input by 1.

In each state we stop, we print out what inside that state (it's content) so the output will be more than input by one because we start with the start state and print out its content before we trace the input string.

Input: 010

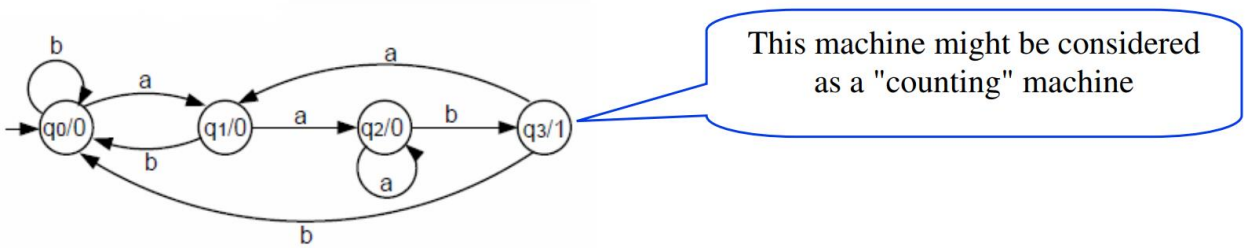
Transition: $\delta(q_0, 0) \Rightarrow \delta(q_1, 1) \Rightarrow \delta(q_1, 0) \Rightarrow q_2$

Output: 1110 (1 for q_0, 1 for q_1, again 1 for q_1, 0 for q_2)

Example 2:

Suppose we were interested in knowing exactly how many times the substring **aab** occurs in a long input string. The following Moore machine will "count" this:

(*) أستاذ مساعد، قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل.



Input string	aaababbaabb
State	q0q1q2q2q3q1q0q0q1q2q3q0
Output	000010000010

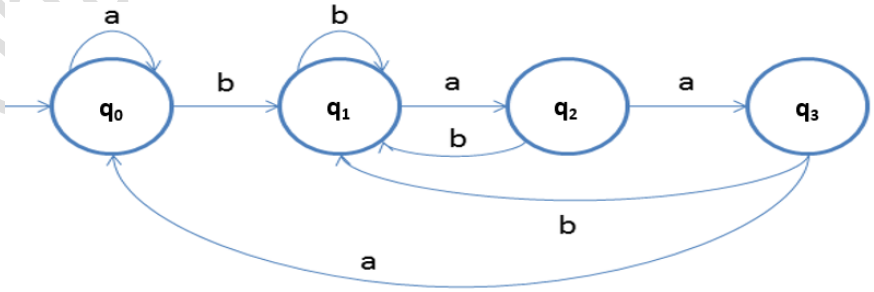
this machine gives 1 after each aab

- Every state of this machine prints out the character 0 except for state q_3, which prints 1.
- To go to state q_3, we have to come from state q_2 and we have to read b.
- To go to state q_2, we have to read at least two a's in a row, having started in any state.
- After finding the substring **aab** and tallying a 1 for it, the machine looks for the next aab. Hence, the number of 1's in the output string is exactly the number of substrings **aab** in the input string.
- To count up how many 1's are in the output string we could use bit collection methods from assembly-language programming, depending on the application we have in mind.
- The example above is part of a whole class of useful Moore machines.
- The 1's in any output sequence mark the end position of all substrings of the input string starting from the first letter of the words in L.
- The machine above with q_0 = -, q_3 = + accepts all words that end in **aab**.

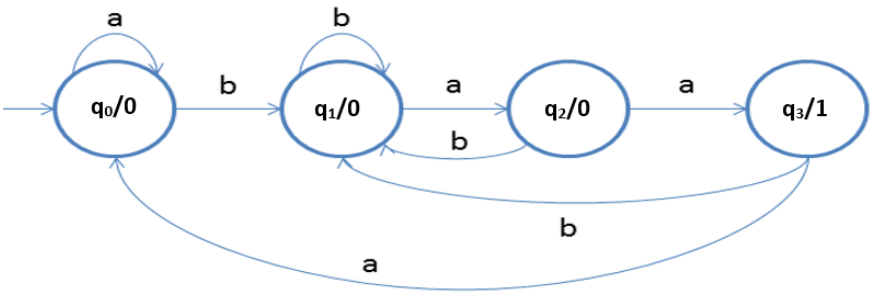
HW. *Input string:* **aabaabaaababaab**

Example 3:
Design Moore machine for counting baa's in a given string.

Solution:
Firstly, we will design a DFA for accepting baa's then we will write output inside states.



Now we will assign the output inside states so that we can count baa's. For accepting state q_3 we will assign '1' and for other states we will assign '0'.

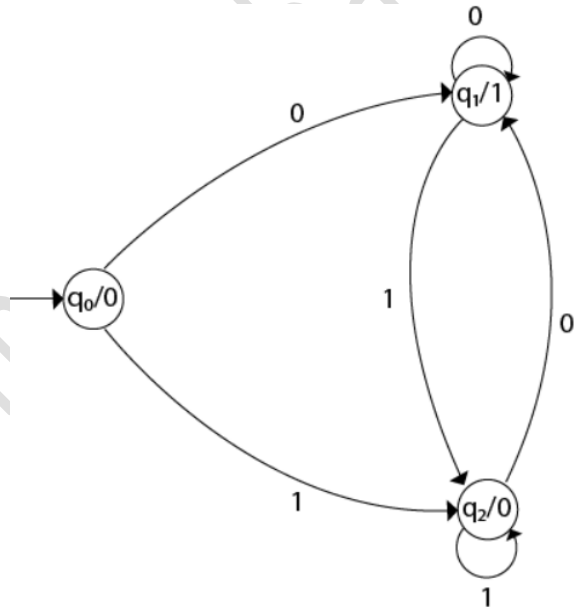


Input string		b	a	a	b	a	a
State	q_0	q_1	q_2	q_3	q_1	q_2	q_3
Output	0	0	0	1	0	0	1

As we can see, we got '1' twice.

Example 4:
Design a Moore machine to generate 1's complement of a given binary number.

Solution: To generate 1's complement of a given binary number, the simple logic is that if the input is 0 then the output will be 1 and if the input is 1 then the output will be 0. That means there are three states. One state is start state. The second state is for taking 0's as input and produces output as 1. The third state is for taking 1's as input and producing output as 0. Hence the Moore machine will be:



For instance, take one binary number 1011 then:

Input string		1	0	1	1
State	q_0	q_2	q_1	q_2	q_2
Output	0	0	1	0	0

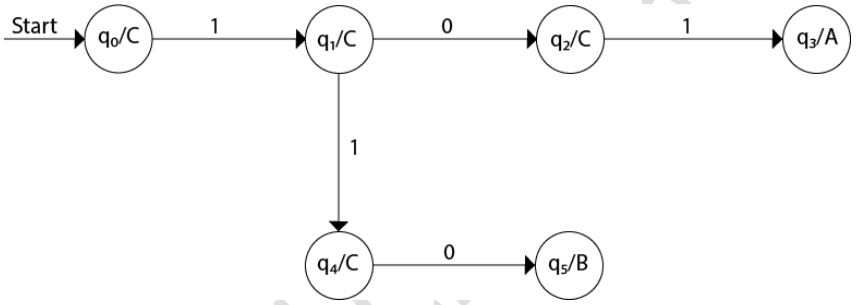
Thus, we get 00100 as 1's complement of 1011, we can neglect the initial 0 and the output which we get is 0100 which is 1's complement of 1011. The transaction table is as follows:

(*) أستاذ مساعد، قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل.

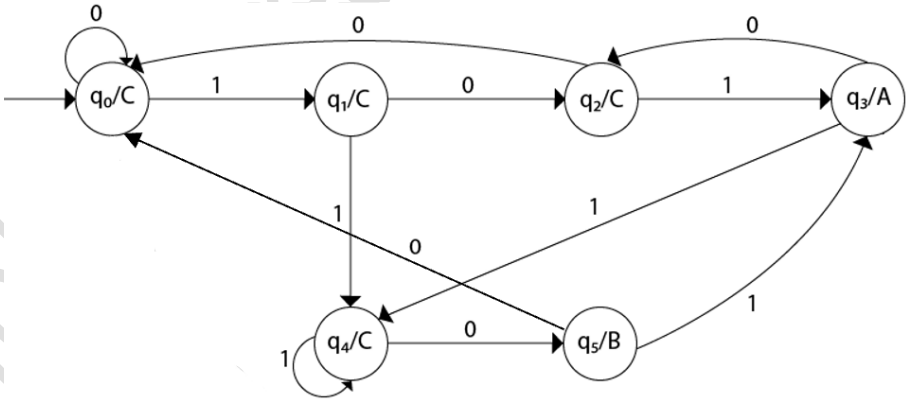
Current State	Next State (δ)		Output (λ)
	0	1	
q_0	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

Example 5:
Design a Moore machine for a binary input sequence such that if it has a substring 101, the machine output A, if the input has substring 110, it outputs B otherwise it outputs C.

Solution: For designing such a machine, we will check two conditions, and those are 101 and 110. If we get 101, the output will be A, and if we recognize 110, the output will be B. For other strings, the output will be C.
The partial diagram will be:



Now we will insert the possibilities of 0's and 1's for each state. Thus, the Moore machine becomes:

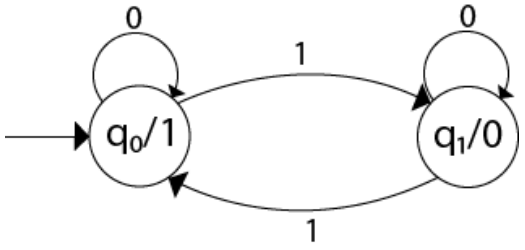


Input string		0	1	1	0	1	1
State	q_0	q_0	q_1	q_4	q_5	q_3	q_4
Output	C	C	C	C	B	A	C

Example 6:
Construct a Moore machine that determines whether an input string contains an even or odd number of 1's. The machine should give 1 as output if an even number of 1's are in the string and 0 otherwise.

(*) أستاذ مساعد، قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل.

Solution:
The Moore machine will be:

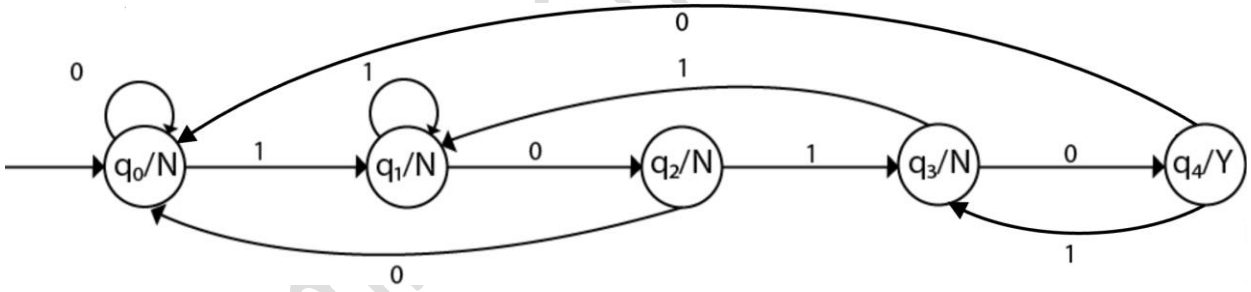


Input string		0	1	1	0	0	1
State	q_0	q_0	q_1	q_0	q_0	q_0	q_1
Output	1	1	0	1	1	1	0

This is the required Moore machine. In this machine, state q_1 accepts an odd number of 1's and state q_0 accepts even number of 1's. There is no restriction on a number of zeros. Hence for 0 input, self-loop can be applied on both the states.

Example 7:
Design a Moore machine with the input alphabet {0, 1} and output alphabet {Y, N} which produces Y as output if input sequence contains 1010 as a substring otherwise, it produces N as output.

Solution:
The Moore machine will be:



Input string		0	1	0	1	0	1	0	1	0	1
State	q_0	q_0	q_1	q_2	q_3	q_4	q_3	q_4	q_3	q_4	q_3
Output	N	N	N	N	N	Y	N	Y	N	Y	N

HW 1. *Construct a Moore machine that outputs a binary string contains 1 for every double letter substring in an input string composed of a's and b's. For example, if abba is the input string 0010 is the corresponding output.*

HW 2. *Construct a Moore machine that takes set of all string over {0, 1} as input and produce 'A' as output if the input ends with '10' or produce 'B' as output if the input ends with '11' otherwise produce 'C' as the output.*

HW 3. *Construct a Moore machine that the machine's output will be a "1" whenever the sequences of 010 or 000 are detected; otherwise the output will be a "0".*

(*) أستاذ مساعد، قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل.