THEORY OF COMPUTATION

UNIT - 3

K CHAIRMADURAI ME,.. AP

Department of Computer Science and Engineering

Adhiparasakthi Engineering College , Melmaruvathur

Pumping Lemma for Regular Languages

- The class of languages known as the **regular languages** has at least four different descriptions. They are the languages accepted by **DFA's**, by **NFA's**, by **epsilon NFA's** they are also the languages defined by **regular expressions**.
- Not every language is a regular language.
- Pumping lemma for showing certain languages not to be regular.

Pumping Lemma

- **For any regular language L, there exists an integer n, such that for all**

  $x \in L$ with $|x| \geq n$, there exists u, v, w $\in \Sigma*$, such that $x = uvw$, and

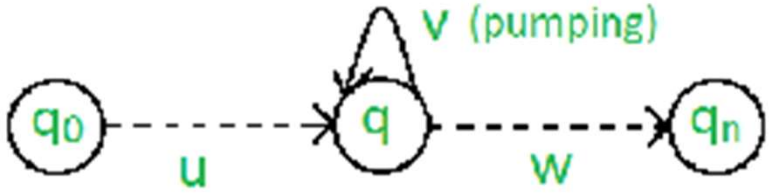
  (1) $|uv| \leq n$

  (2) $|v| \geq 1$

  (3) for all $i \geq 0$: $uv^iw \in L$

Pumping Lemma for Regular Languages

- In simple terms, this means that if a string v is ‘pumped’, i.e., if v is inserted any number of times, the resultant string still remains in L.



- Pumping Lemma is used as a proof for irregularity of a language. Thus, if a language is regular, it always satisfies pumping lemma.
- If there exists at least one string made from pumping which is not in L, then L is surely not regular.

Proving Languages not to be Regular

- This pumping lemma based on **proof by contradiction**
- Steps to proving languages not to be regular

1. Assume **L** is Regular and also Assume n be the number of states in the finite automato accepting L.
2. Choose the string **w** {word in L}

   Such that **|w| ≥ n**
3. The string w can be split into three parts x y z

   Such that (i) **|xy| ≤ n** (ii) $y \neq \epsilon$ (or) $y>0$
4. Choose a suitable integer **i**,

   Such that **i ≥ 0**
5. Assume the string $\mathbf{xy^iz}$ where **i ≥ 0**

   Such that $\mathbf{xy^i z \notin L}$
6. Assumption is False. So L is Not Regular

Example

- **Prove that $L = \{0^n1^n \mid n \geq 1\}$ is not regular.**

soln:

1. Let us assume that **L is regular**, then by Pumping Lemma the above given rules follow.

   L={01,0011,000111,.....} Assume **n=2**

2. Now, let **w ∈ L** and **|w| ≥ n**. So, by Pumping Lemma.

   w=0011 , |4| ≥ 2



3. The string w can be split into three parts x y z

   w=xyz

   x=0, y=0, z=11 **|xy| ≤ n** ----- 2≤2

   and **|v| > 0** ----- 1 >0

4. Choose the suitable integer **i=2, i ≥ 0**

5. Assume the string **xy^iz** where **i ≥ 0**

   $00^2 11$

   $000111 \notin L$

6. Our assumption is False. So L is Not Regular

Home Work

1. Show that L={a^{i2} | i ≥1 } is not regular.
2. show that L={ a^{p} p is a prime} is not regular.
3. show that L={ $0^{i}1^{i}$ | i ≥ 1} is not regular.
4. Show that L={ 0^{n} 1^{m} | n>m and n ≥0} is not regular.
5. Show that L={ ww^{R} | w Σ (a,b)*} is not regular.
6. show that L={$a^{n}b^{n}c^{n}$ | n ≥ 0 } is not regular.
7. show that L={a^{2n} | n ≥1 } is not regular.
8. Show that L={ $a^{n!}$ | n>0} is not regular.

Home Work

* a) $\{0^n \mid n \text{ is a perfect square}\}$.

b) $\{0^n \mid n \text{ is a perfect cube}\}$.

c) $\{0^n \mid n \text{ is a power of 2}\}$.

d) The set of strings of 0's and 1's whose length is a perfect square.

Closure properties of regular languages

- Closure properties on regular languages are defined as certain operations on regular language which are guaranteed to produce regular language.
- Closure refers to some operation on a language, resulting in a new language that is of same “type” as originally operated on i.e., regular.
- A set is closed under an operation if, whenever the operation is applied to members of the set, the result is also a member of the set.

Closure properties of regular languages

- The principal closure properties of regular languages are as follows,

1. The **union** of two regular languages is regular.
2. The **intersection** of two regular languages is regular.
3. The **complement** of a regular language is regular.
4. The **difference** of two regular languages is regular.
5. The **reversal** of two regular languages is regular.
6. The **closure** of two regular languages is regular.
7. The **concatenation** of two regular languages is regular.
8. The **homomorphism** of a regular language is regular.
9. The **inverse homomorphism** of a regular language is regular.

Closure under Union

Theorem:

If L and M are regular languages, then so is LUM is also regular.

Proof :

Let L and M be the languages of regular expressions R and S, respectively.

Then R+S is a regular expression whose language is L U M

Let, L = L [R]

M = L[S]

LUM = L[R] U L [S]

= L [R+S]

Closure under Intersection

Theorem

If L and M are two regular languages, then L∩M is also regular language.

Proof:

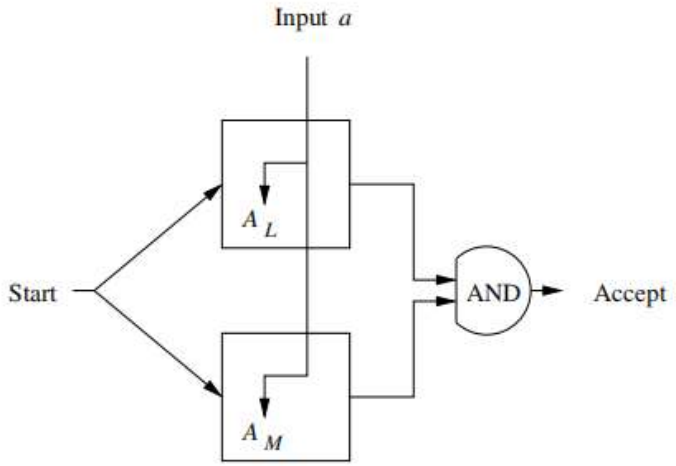
Let L and M be the languages of automata

$A_L = (Q_L, \sum, \delta_L, q_L, F_L)$ and $A_M = (Q_M, \sum, \delta_M, q_M, F_M)$

$$A = (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)$$

where $\delta((p,q),a) = (\delta_L(p,a), \delta_M(q,a))$.

$$L[A] = L[A_L] \cap L[A_M]$$



- By induction on W ,

$$\hat{\delta}((q_L, q_M), w) = (\hat{\delta}_L(q_L, w), \hat{\delta}_M(q_M, w))$$

But A accepts w if and only if $\hat{\delta}((q_L, q_M), w)$ is a pair of accepting states

(i.e.,) must be in F_L and

must be in F_M

Putting in another way, " w is accepted by A if and only if both A_L and A_M accepts w.

Thus A accepts the intersection of L and M.

Closure under Intersection

$A_A = (Q_A, \sum, \delta_A, q_a, F_A)$

$A_B = (Q_B, \sum, \delta_B, q_B, F_B)$

$A \cap B = (Q_A \times Q_B, \sum, \delta(q_A \times q_B, F_A \times F_B))$

Where

$\delta((p, q), a) = \delta_L(p, a), \delta_M(q, a))$

Here,

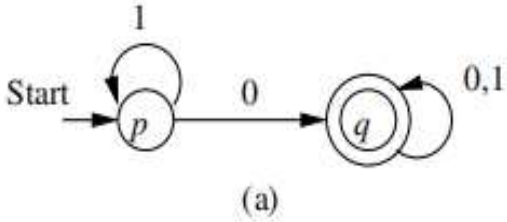
$Q_A \times Q_B = \{p,q\} \times \{r,s\}$

$= \{(p, r), (p, s), (q, r), (q, s)\}$

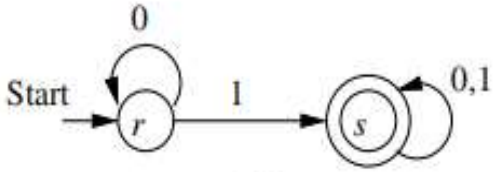
$\bar{Z} = \{0, 1\}$

$q_A \times q_B = \{p, r\}$

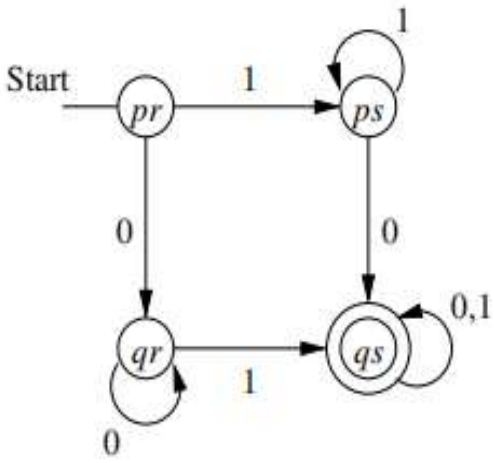
$F_A \times F_B = \{q, s\}$



(a)



(b)



(c)

Closure under Reversal

Theorem

If L is a regular language, then L^R is also regular.

Proof

Assume L is defined by regular expression E. The proof is a structural induction on the size of E. We show that there is another regular expression.

E^R such that $L(E^R) = (L(E))^R$.

That is the language E^R is the reversal of the language of E.

Closure under Reversal

Let w=0010

w^R = 0100

L={001,1110}

L^R= {100,0111}

- Basis

  If E = Є / Φ / a, then

  E^R =Є / Φ / a

  E = E^R

- Induction

  There are 3 cases depending on the form of E.

  1) E = E1 + E2 then **E^R = $E1^R$ + $E2^R$**

  2) E = E1.E2 then **E^R = $E1^R$. $E2^R$**

  3) E=E1* then **E^R = ($E1^R$)***

Closure under Complementation

Theorem

If L is a regular language over the alphabet∑, $\bar{L} = \sum^* - L$ then is also a regular language.

Proof

Let L= L [A] for some DFA, where

A = (Q, ∑, δ , q0, F)

Then, $\bar{L}$ = L [B], where

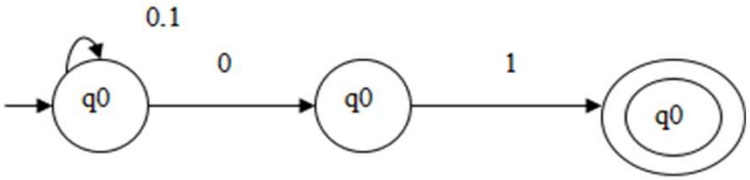
B = (Q, ∑, δ , q0, Q-F)

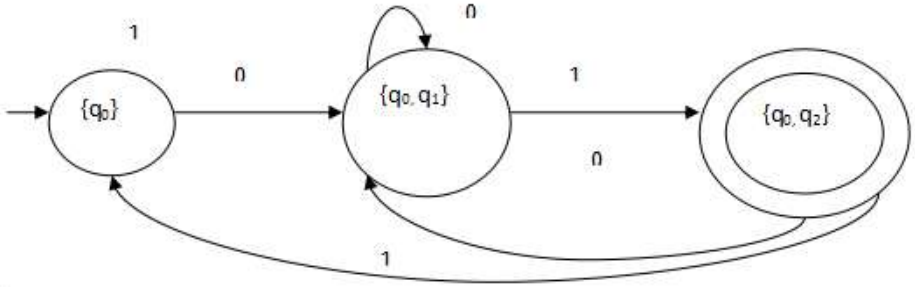
B is exactly like A, but the accepting states of A have not become the accepting states of B and vice-versa.

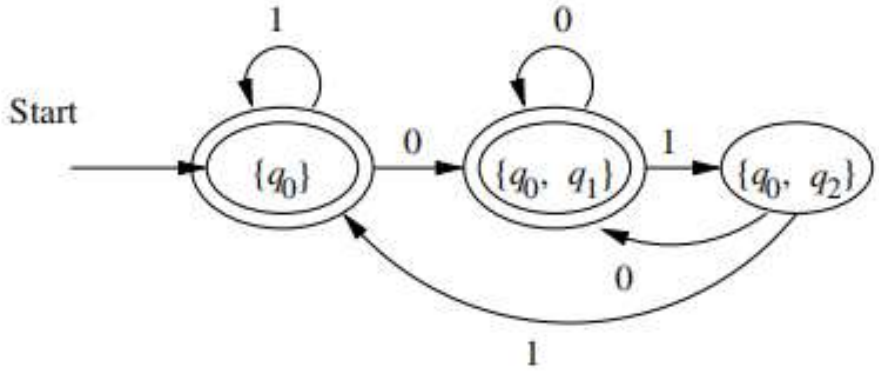
Then w is in L [B], if and only if, $\hat{\delta}(q_0, w)$ is in $\{Q - F\}$ is in , which occur if and only if w is not in L [A].

Closure under Complementation

Find the complement of (0+1)*01







: DFA accepting the complement of the language $(\mathbf{0+1})^*\mathbf{01}$

Closure under Difference

Theorem

If L and M are regular languages then L – M is also regular

Proof

$$L - M = L \cap \overline{M}$$

Since M is a regular language, $\overline{M}$ is also a regular language by closure under complementation. Since L and $\overline{M}$ are regular, the $L \cap \overline{M}$ is also regular by closure under intersection.

so, L – M is a regular Language.

Homomorphism

A string homomorphism is a function on strings that works by **substituting a particular string for each symbol**.

Example: Let h(0) = ab

h(1) = Є

is a homomorphism, where replace all 0's by ab and replace all 1's by Є

Let w=0011

h(0011) = h(0)h(0)h(1)h(1)

h(w) = abab

Homomorphism

Theorem

If L is a regular language over an alphabet ∑, then h is a homomorphism function on ∑ then h (L) is also regular.

Proof

To prove: $L[h(E)] = h[L(E)]$

Basis

If $E = \varepsilon$ or ϕ, then

$H(E) = \varepsilon$ or ϕ

$\therefore L[h(E)] = L(E)$

If E is ε or ϕ, then L(E) contains either no strings or a string with no symbols.

Thus,

$H[l(E)] = l(E) = L[h(E)]$

If $E = a$, then

$L(E) = \{a\}$

$h[(E)] = \{\{h(a)\} = L[h(E)]$ for $E = \{a\}$

Homomorphism

Induction

There are 3 cases in induction part → Union, Concatenation and closure.

Let $E = F + G$

$L[h(E)] = L[h(F + G)]$

$= L[h(F) \cup h(G)]$

$= L[h(F)] \cup L[h(G)]$ →1

$H[L(E)] = h[L(F + G)]$

$= h[L(f) \cup L(G)]$

$= h[L(f)] \cup h[L(G)]$ →2

Therefore by induction, from (1) and (2)

$L[h(f)] = h[L(f)]$ and

$L[h(G)] = h[L(G)]$

Similarly concatenation and closure is also true,

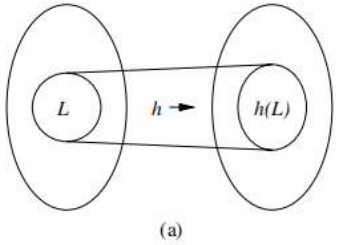
$\therefore L[h(R)] = h[L(R)] \Rightarrow$ Hence proved

Inverse homorphism

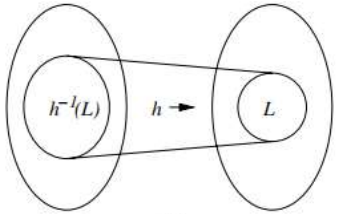
$h^{-1}(L)$ is the set of strings, w in $\sum^*$ such that h(w) is in L.

Theorem

If h is a homomorphism from alphabet, $\sum$ to alphabet T and L is regular language over T, then $h^{-1}(L)$ is also a regular language.



(a)



(b)

Figure 4.5: A homomorphism applied in the forward and inverse direction

Example: Suppose h is a homomorphism from the alphabet {0, 1, 2} to the alphabet {a, b} defined by h(0) = a, h(1) = ab, h(2) = ba.

(i) h (0120) = h (0)h(1)h(2)h(0)

= aabbaa

(ii) h(21120) = h(2)h(1)h(1)h(2)h(0)

= baababbaa

(iii) L[01*2]

H(L) = h(01*2)

= L[h(01*2)]

= L[h(0)h(1*)h(2)]

= L[a(ab)*ba]

(v) $h^{-1}(L) \Rightarrow h^{-1}$(ababa) where L = ababa

$= h^{-1}(a)h^{-1}(ba)h^{-1}(ba)$

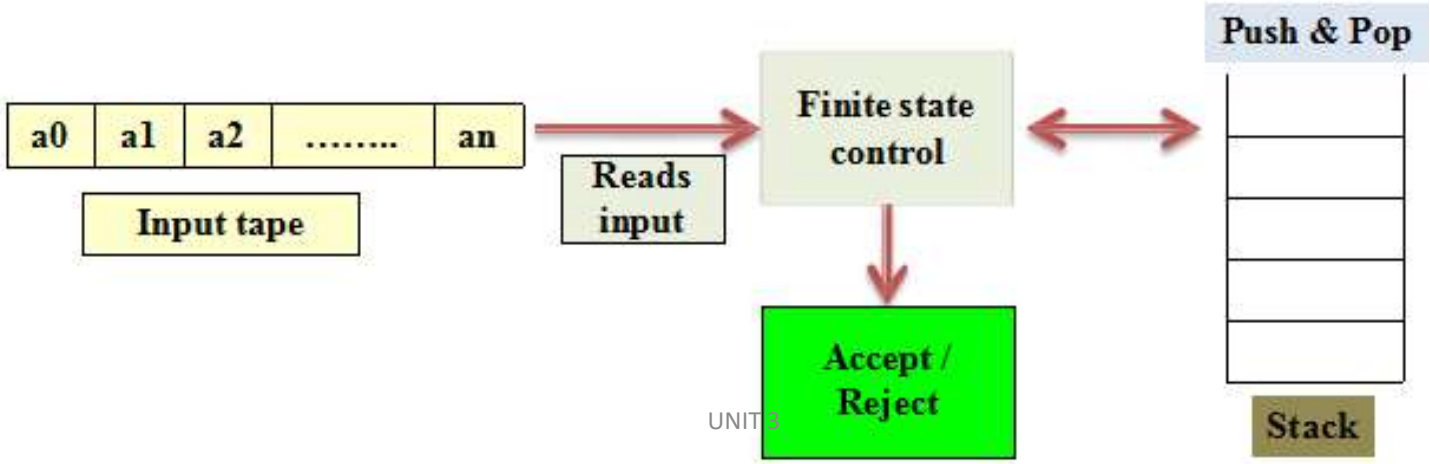
= 022.

Pushdown Automata (PDA)

- **Pushdown Automata** is a **Finite Automata** with extra memory called **stack** which helps **Pushdown Automata** to recognize **Context Free Languages.**
- The stack in the PDA provides **memory to store intermediate states** and results which increases the **capability of automation**.
- Due to the availability of stack, PDA becomes **more powerful than a finite automata.**
- The important feature of PDA is stack, which is **used to remember the input**. **Finite Automata associated with a stack is known as PDA.**

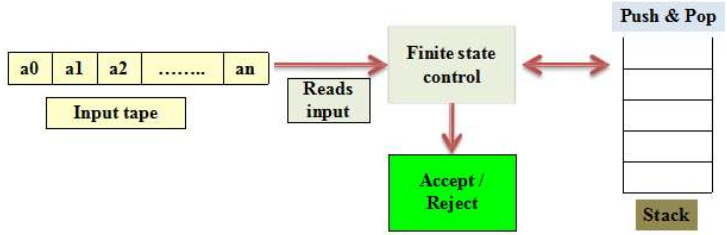
How does a pushdown automata work?

- A pushdown automaton reads a given input string from left to right.
- In each step, it chooses a **transition** by indexing a table by **input symbol, current state, and the symbol at the top of the stack**.
- A pushdown automaton can also manipulate the stack, as part of performing a transition.



PDA Components

- **Input tape**: The input tape is divided in **many cells or symbols.** The input head is read-only and may only move from **left to right**, one symbol at a time.
- **Finite control**: The finite control has some **pointer** which points the **current symbol which is to be read.**
- **Stack**: The stack is a structure in which we can push and remove the items from one end only. It has an **infinite size.** In PDA, the stack is used to **store the items temporarily.**
- The stack head **scans the top symbol of the stack.**
- A stack does two operations –
  - **Push** – a new symbol is added at the top.
  - **Pop** – the top symbol is read and removed.



Why do we need Pushdown Automata?

- **Finite Automata** can be used to accept **only regular languages**.
- **Finite Automata** can remember a **finite amount of information**, but a **PDA** can remember an **infinite amount of information**.
- Pushdown Automata is simply an **NFA augmented with an "external stack memory".**
- Pushdown Automata to recognize **Context Free Languages.**
- They can use the top of the stack to decide which transition to take.
- They can manipulate the stack as part of performing transition.

Formal Definition of Pushdown Automata

A PDA can be formally described as a 7-tuple **$(Q, \sum, \Gamma, \delta, q_0, Z_0, F)$**

- **Q** is the **finite number of states**
- $\sum$ is the **finite set of input alphabets**
- Γ is **stack symbols**
- **δ** is the **transition function**: $Q \times (\sum \cup \{\varepsilon\}) \times \Gamma \dashrightarrow Q \times \Gamma^*$
- **q_0** is the **start state** ($q_0 \in Q$)
- Z_0 is the **initial stack top symbol** ($Z_0 \in \Gamma$)
- **F** is a **set of accepting states or final states** ($F \in Q$)

δ is the transition function

old state *input symb.* *Stack top* *new state(s)* *new Stack top(s)*

$\delta : Q \times \sum \times \Gamma \Rightarrow Q \times \Gamma$

$$\delta(q,a,X) = \{(p,Y), \ldots\}$$

1. **state transition from q to p**
2. **a** is the **next input symbol**
3. **X** is the **current stack *top* symbol**
4. **Y** is the **replacement for X**;
   it is in Γ^* (a string of stack symbols)
   i. Set **Y = ε** for: Pop(X)
   ii. If **Y=X**: stack top is unchanged
   iii. If **$Y=Z_1Z_2\ldots Z_k$:** X is popped and is replaced by Y in reverse order (i.e., Z_1 will be the new stack top)

	Y = ?	Action
i)	Y=ε	Pop(X)
ii)	Y=X	Pop(X) Push(X)
iii)	$Y=Z_1Z_2..Z_k$	Pop(X) Push(Z_k) Push(Z_{k-1}) … Push(Z_2) Push(Z_1)

Conventions of Δ transition

$$\Delta(q,w,z_0)=(p,\alpha)$$

$\delta(q_0,a,z_0) = (q_1,az_0)$

$\delta(q_1,a,a) = (q_1,aa)$

$\delta(q_1,a,a) = (q_1,aa)$

$\delta(q_1,b,a) = (q_2,\varepsilon)$

$\delta(q_2,b,a) = (q_2,\varepsilon)$

$\delta(q_2,b,a) = (q_2,\varepsilon)$

$\delta(q_2,\Delta,z_0) = (q_3,\Delta z_0)$

- q ➏ represents state or current state.
- W ➏ input symbol.
- Z_0 ➏ start symbol for stack or top of stack.
- P ➏ state after push / pop operator.
- α ➏ stack elements after performing push or pop operations.

Instantaneous Descriptions (ID)

- ID is an informal notation of **how a PDA computes an input string** and **make a decision that string is accepted or rejected.**
- A ID is a triple (q, w, α), where:
  1. q is the current state.
  2. w is the remaining input.
  3. α is the stack contents, top at the left.

If $\delta(q,a, X)=\{(p, A)\}$ is a transition, then the following are also true:

- $(q, a, X) \vdash (p,\varepsilon,A)$
- $(q, aw, XB) \vdash (p,w,AB)$

|--- sign is called a "turnstile notation" and represents one move

|---* sign represents a sequence of moves

Example: Instantaneous Descriptions (ID)

PDA to accept $\{0^n1^n \mid n \geq 1\}$

$(q, \mathbf{0}00111, Z_0) \vdash (q, 00111, 0Z_0)$
$\vdash (q, 0111, 00Z_0)$
$\vdash (q, 111, 000Z_0)$
$\vdash (p, 11, 00Z_0)$
$\vdash (p, 1, 0Z_0)$
$\vdash (p, \epsilon, Z_0)$
$\vdash (f, \epsilon, Z_0)$

Thus, $(q, 000111, Z_0) \vdash^* (f, \epsilon, Z_0)$.

000111 is Accepted by PDA

$(q, 0001111, Z_0) \vdash (q, 001111, 0Z_0)$
$\vdash (q, 01111, 00Z_0)$
$\vdash (q, 1111, 000Z_0)$
$\vdash (p, 111, 00Z_0)$
$\vdash (p, 11, 0Z_0)$
$\vdash (p, 1, Z_0)$
$\vdash (f, 1, Z_0)$

Note the last ID has no move.

0001111 is not accepted, because the input is not completely consumed.

Languages of PDA (PDA Acceptance)

- A language can be accepted by Pushdown automata using two approaches:
- Acceptance by Final State:
  - The PDA is said to accept its input by the final state if it enters any final state in zero or more moves after reading the entire input.
- Acceptance by Empty Stack:
  - On reading the input string from the initial configuration for some PDA, the stack of PDA gets empty.

Languages of PDA (PDA Acceptance)

- *PDAs that accept by **final state**:*
  - For a PDA P, the language accepted by P, denoted by **L(P)** by *final state*, is:
    - **$L(P)=\{w \mid (q_0,w,Z_0) \vdash^* (q,\varepsilon, A)\}$**, s.t., $q \in F$
- *PDAs that accept by **empty stack**:*
  - For a PDA P, the language accepted by P, denoted by **N(P)** by *empty stack*, is:
    - **$N(P)=\{w \mid (q_0,w,Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$**, for any $q \in Q$.

Example Problems - PDA

Example1:

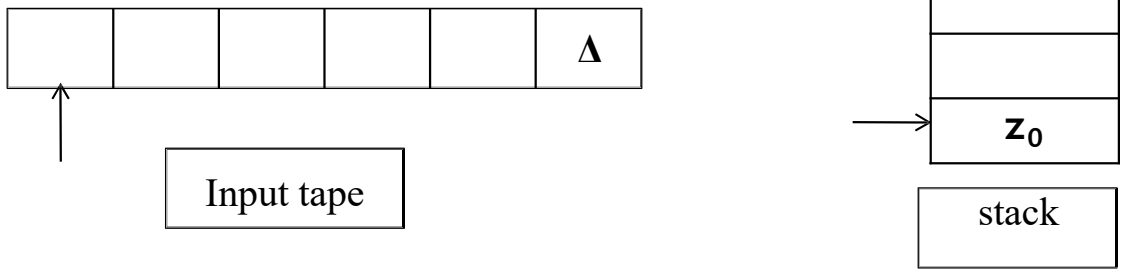
Design a PDA of a language $L=\{a^n b^n, n>=1\}$

Solution:

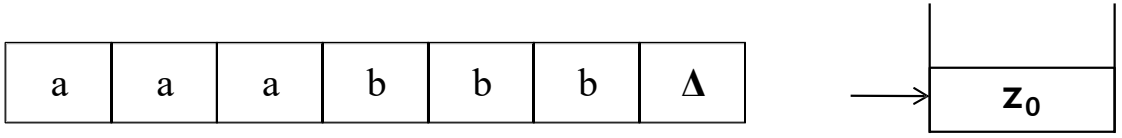
Let us consider n=3

∴ the string w ="aaabbb"

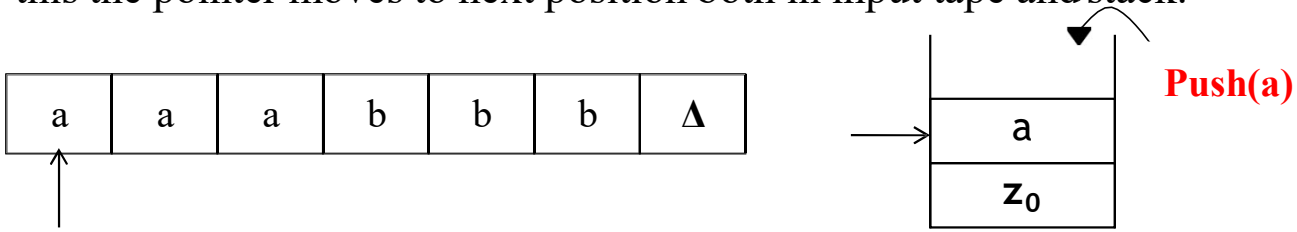
Now, the **input tape and stack** is prepared for doing operations with **pointer in input tape along with Δ as last element** and **z_0 as stack symbol with pointer**.



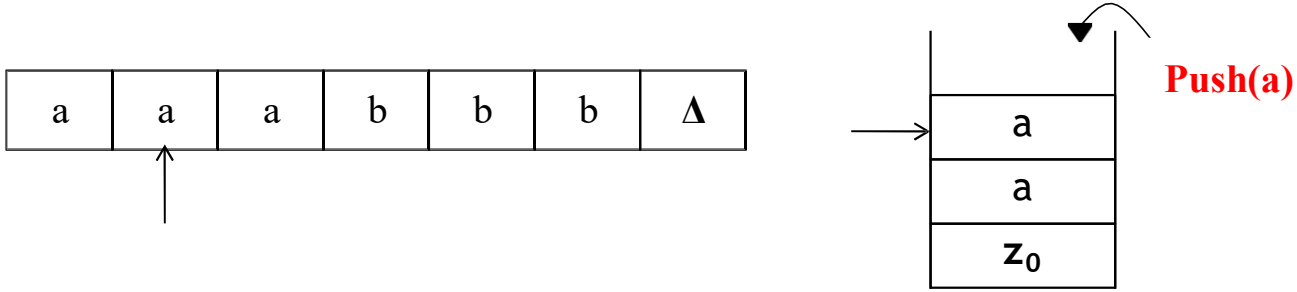
Step1: Insert the input in to the input tape ends with Δ and starts to read the input.



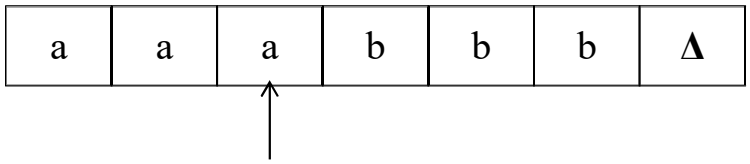
Step2: Read the input from tape, if it is 'a' then perform **push(a)** into the stack. After this the pointer moves to next position both in input tape and stack.

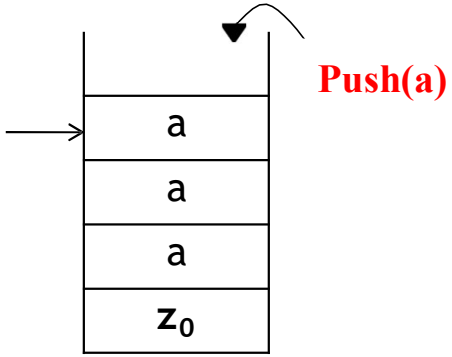


Step3: Move the pointer and read the next input from tape, if it is 'a' then perform **push(a)** into the stack. The pointer moves to top of the stack.

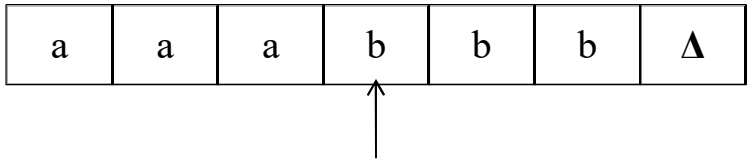


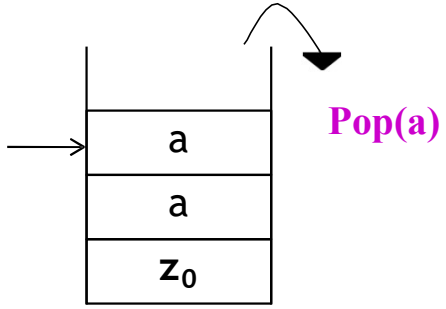
Step4: Move the pointer and read the next input, if it is ‘a’ then perform push(a) into the stack. The pointer moves to top of the stack.



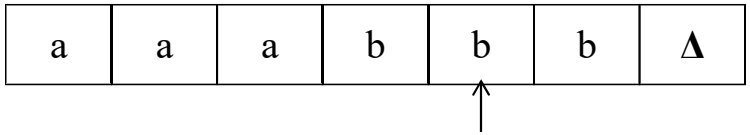


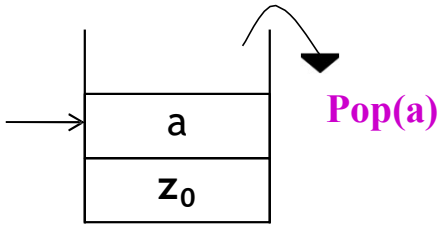
Step5: Move the pointer and read the next input, if it is ‘a’ then perform **push(a)** else **pop(a)** from the stack. The pointer moves to top of the stack.



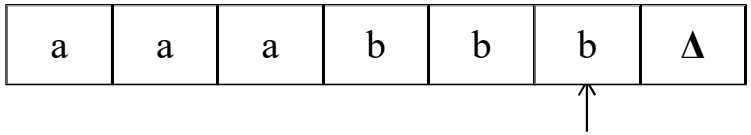


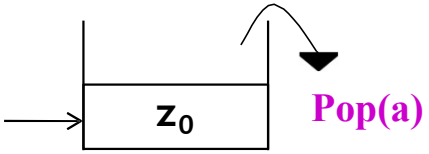
Step6: Move the pointer and read the next input, if it is ‘b’ then perform pop(a) from the stack. The pointer moves to top of the stack.





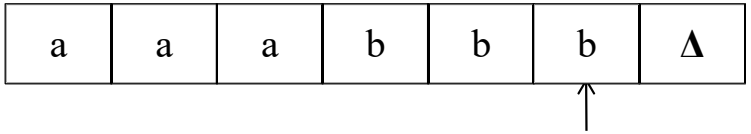
Step7: Move the pointer and read the next input, if it is ‘b’ then perform **pop(a)** from the stack. The pointer moves to top of the stack.

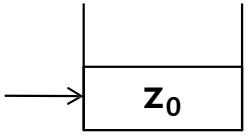




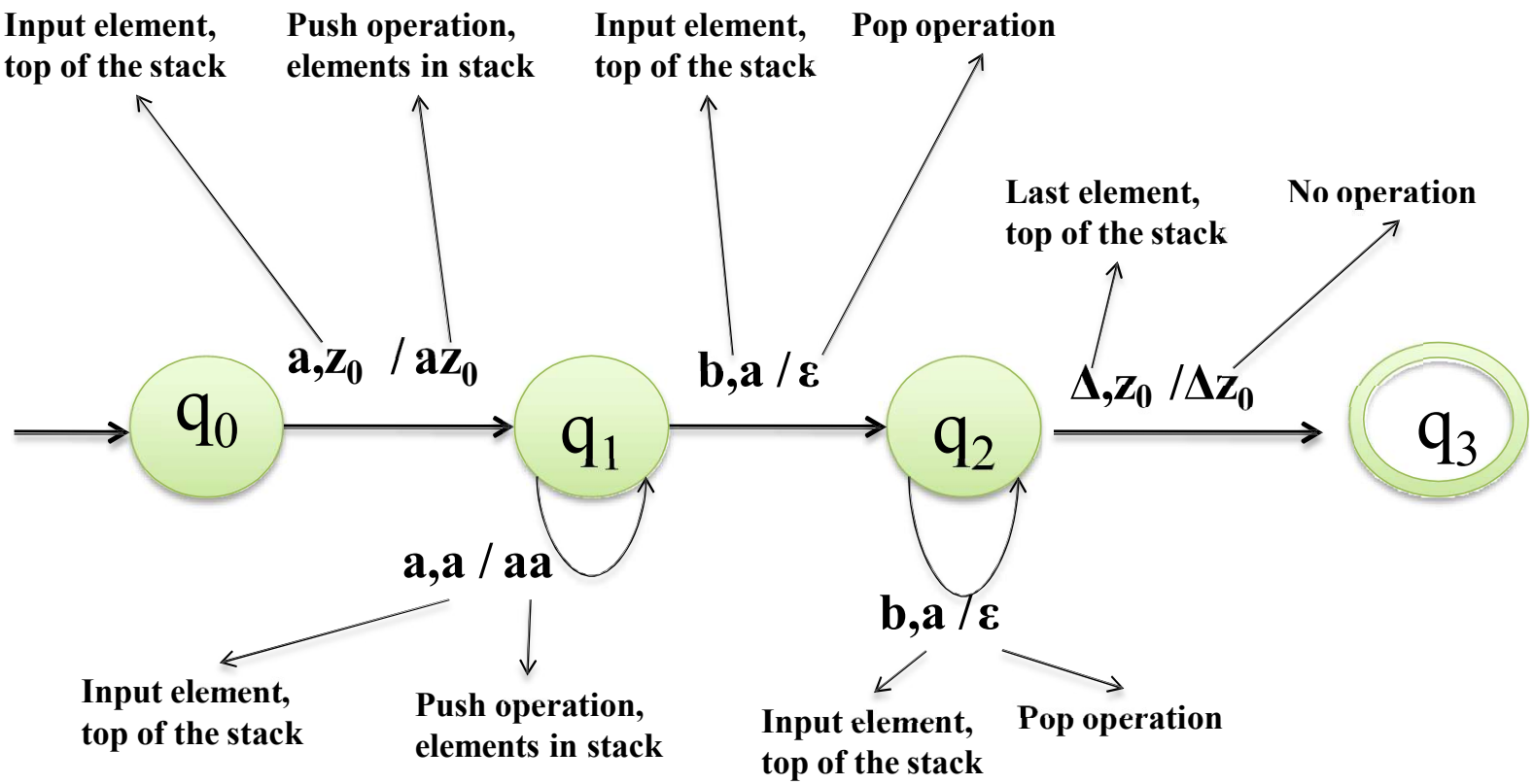
Step8: Move the pointer and read the next input, now the pointer points “**Δ” in input tape and z_0** in the stack.

This state remains and indicates that the PDA goes to an accepting state.

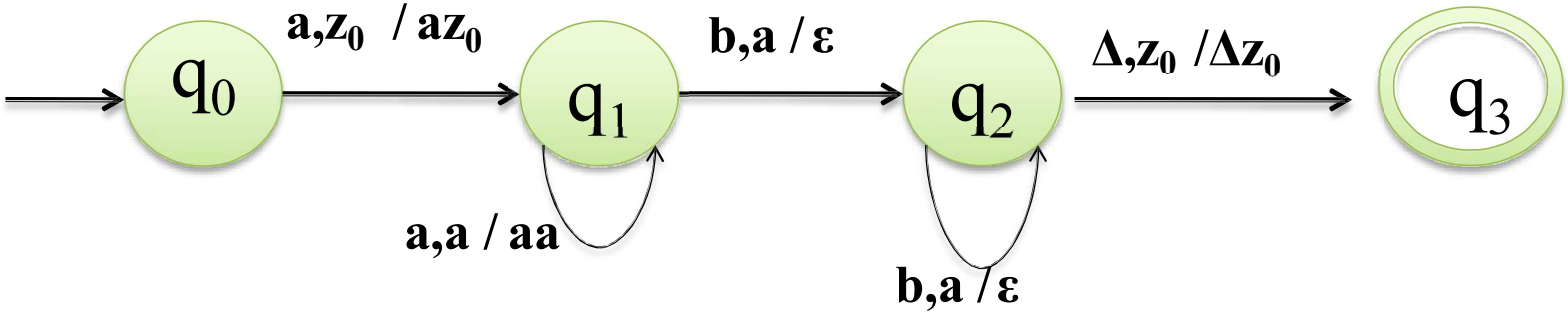




Step9: Transition Diagram



Step9: Transition Diagram



Step 10:Convention of Δ transition: $\Delta(q,w,z_0)=(p,\alpha)$

$\delta(q_0,a,z_0) = (q_1,az_0)$

$\delta(q_1,a,a) = (q_1,aa)$

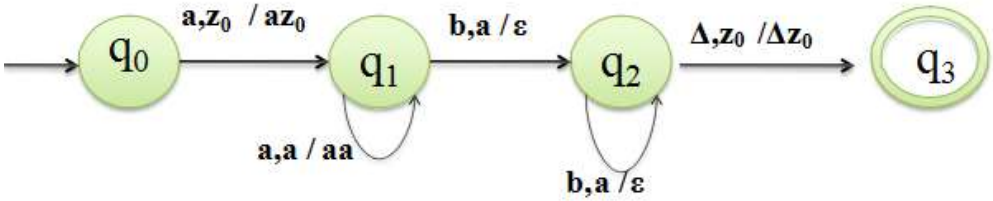
$\delta(q_1,a,a) = (q_1,aa)$

$\delta(q_1,b,a) = (q_2,\varepsilon)$

$\delta(q_2,b,a) = (q_2,\varepsilon)$

$\delta(q_2,b,a) = (q_2,\varepsilon)$

$\delta(q_2,\Delta,z_0) = (q_3,\Delta z_0)$



The given string reaches the final states and hence the string "w" is accepted.

Step 11:Instantaneous description in PDA

- It is represented as (State, String, Stack Element) $(q_0,aaabbb\Delta,z_0)$

$(q_0,aaabbb\Delta,z_0)$ $\vdash (q_1,aabbb\Delta,az_0)$

$\vdash (q_1,abbb\Delta,aaz_0)$

$\vdash (q_1,bbb\Delta,aaaz_0)$

$\vdash (q_2,bb\Delta,aaz_0)$

$\vdash (q_2,b\Delta,az_0)$

$\vdash (q_2,\Delta,z_0)$

$\vdash (q_3,\Delta z_0)$

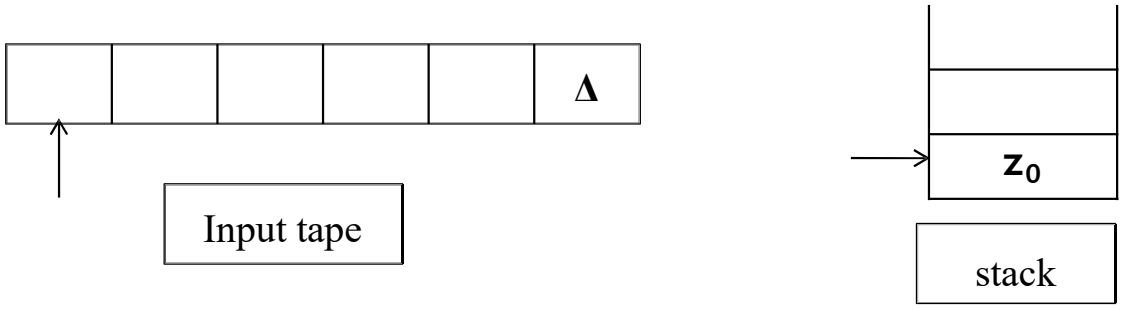
It reaches the final state "q_3", hence the string w="aaabbb" accepted by PDA

Example2:
Design a PDA of a language **L={0^n1^{2n} , n>=1}**
Solution:
Let us consider n=2

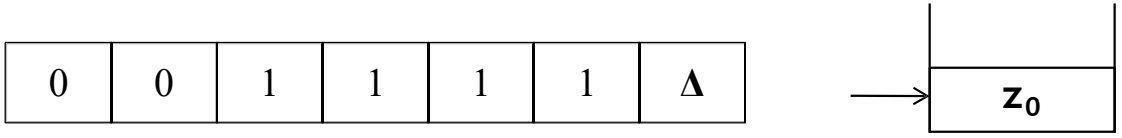
0^21^{2*2} = 0^21^4 = 001111

The string w =“001111”

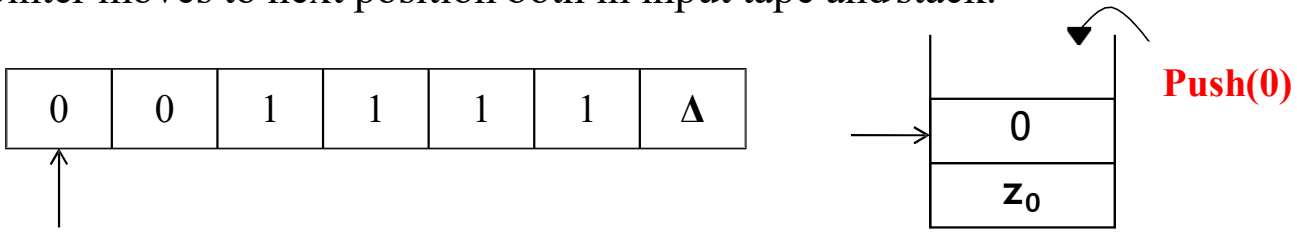
Now, the **input tape and stack** is prepared for doing operations with **pointer in input tape along with Δ as last element** and **z_0 as stack symbol with pointer**.



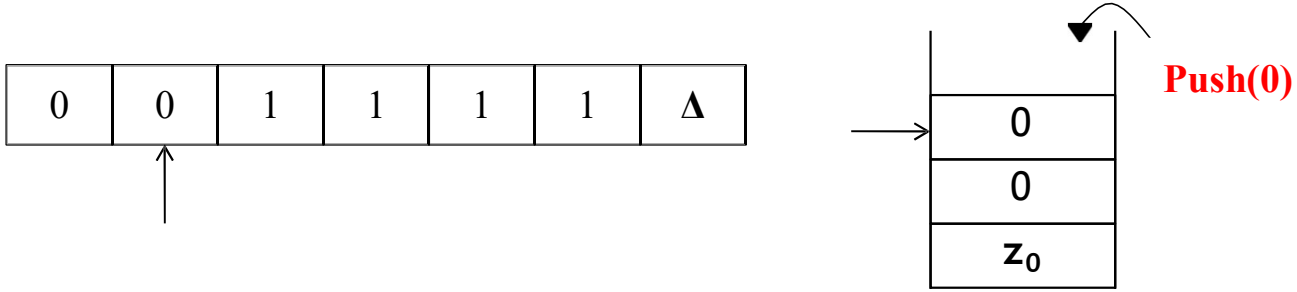
Step1: Insert the input in to the input tape ends with **Δ** and starts to read the input.



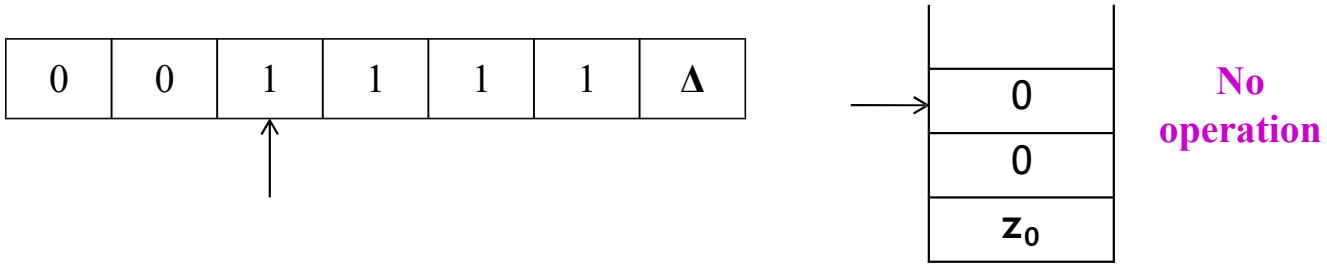
Step2: Read the input from tape, if it is '0' then perform **push(0)** into the stack. After this the pointer moves to next position both in input tape and stack.



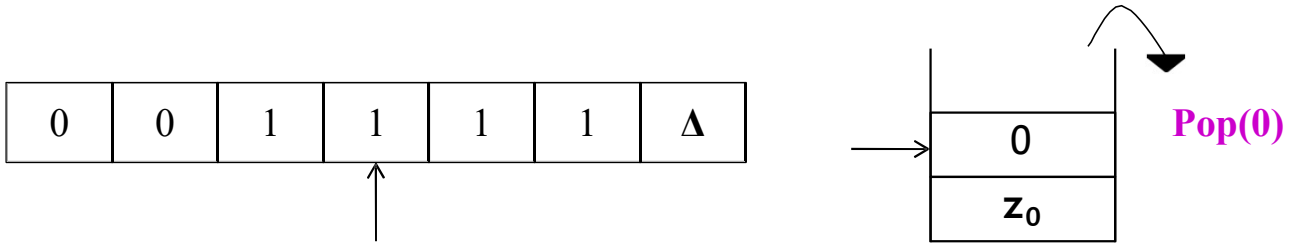
Step3: Move the pointer and read the next input from tape, if it is '0' then perform **push(0)** into the stack. The pointer moves to top of the stack.



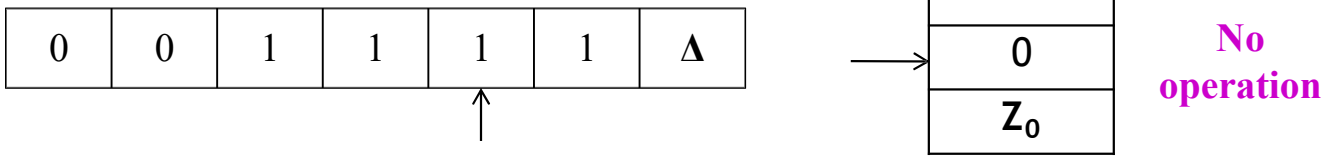
Step4: Move the pointer and read the next input, if it is '1' then no operation (**push / pop**) is performed in the stack. The pointer moves to top of the stack.



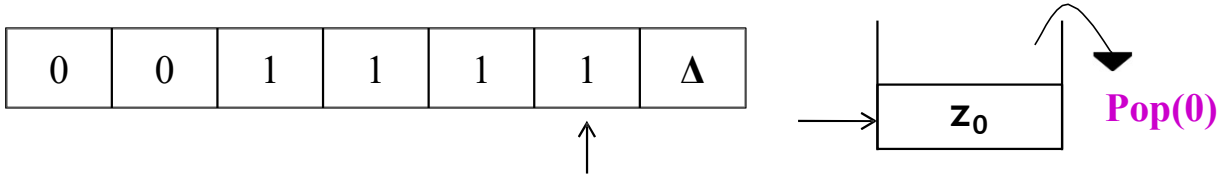
Step5: Move the pointer and read the next input, if it is '1' then perform **pop(0)** from the stack. The pointer moves to top of the stack.



Step6: Move the pointer and read the next input, if it is '1' then no operation (**push / pop**) is performed in the stack. The pointer moves to top of the stack.

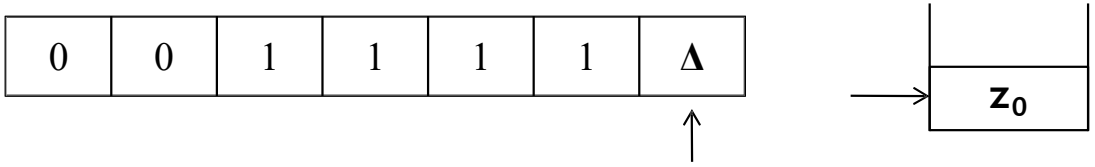


Step7: Move the pointer and read the next input, if it is '1' then perform **pop(0)** from the stack. The pointer moves to top of the stack.

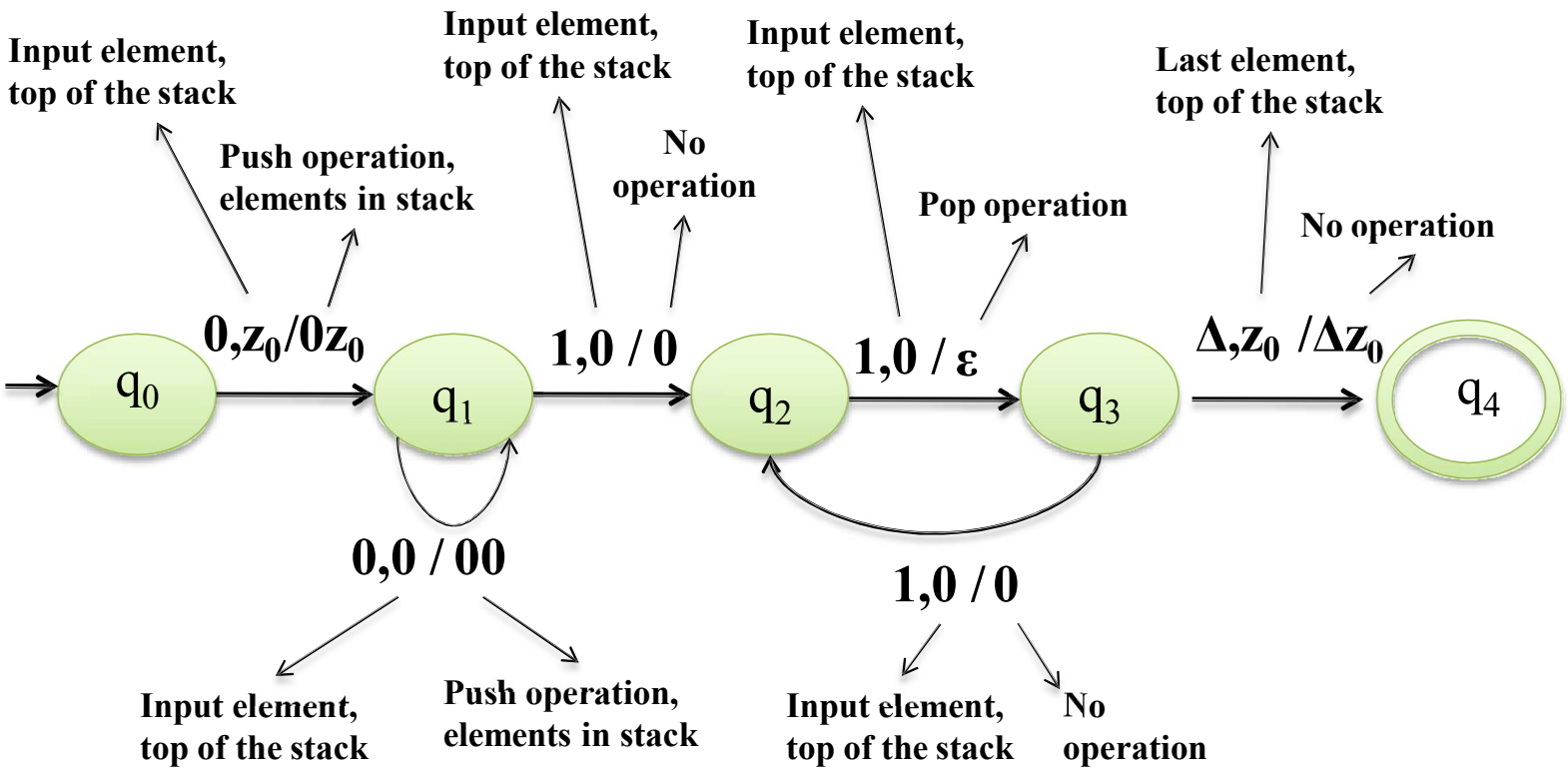


Step8: Move the pointer and read the next input, now the pointer points "**Δ" in input tape and Z_0** in the stack.

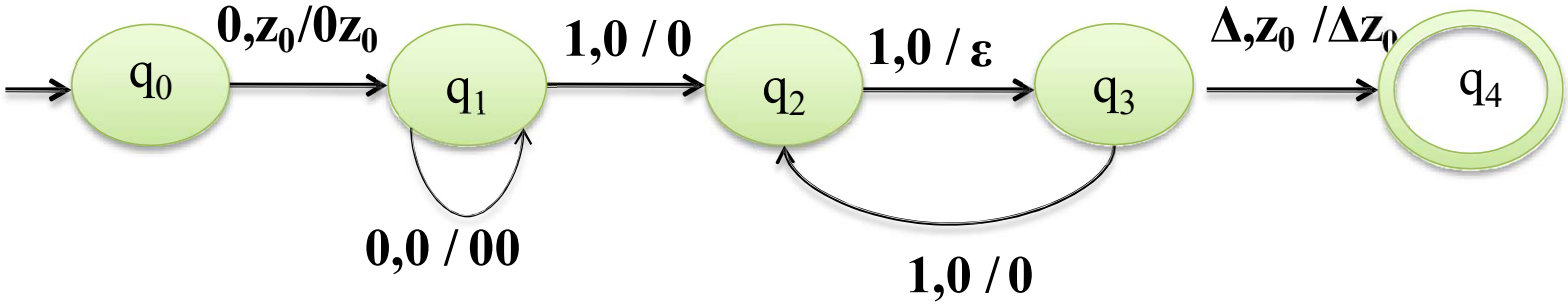
This state remains and indicates that the PDA goes to an accepting state.

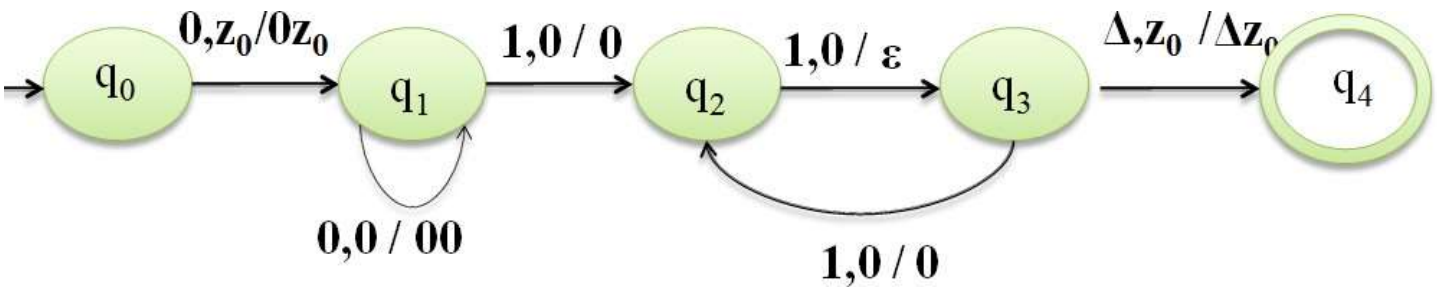


Step9: Transition Diagram



Step9: Transition Diagram





Step 10:Convention of Δ transition: $\Delta(q,w,z_0)=(p,\alpha)$

$\delta(q_0,0,z_0) = (q_1,0z_0)$

$\delta(q_1,0,0) = (q_1,00)$

$\delta(q_1,1,0) = (q_2,00)$

$\delta(q_2,1,0) = (q_3,\varepsilon)$

$\delta(q_3,1,0) = (q_2,0)$

$\delta(q_2,1,0) = (q_3,\varepsilon)$

$\delta(q_3,\Delta,z_0) = (q_4,\Delta z_0)$

The given string reaches the final states and hence the string "w" is accepted.

Step 11:Instantaneous description in PDA

- It is represented as (State, String, Stack Element)

$(q_0,001111\Delta,z_0)$

$(q_0,001111\Delta,z_0) \vdash (q_1,01111\Delta,0z_0)$

$\vdash (q_1,1111\Delta,00z_0)$

$\vdash (q_2,111\Delta,00z_0)$

$\vdash (q_3,11\Delta,0z_0)$

$\vdash (q_2,1\Delta,0z_0)$

$\vdash (q_3,\Delta,z_0)$

$\vdash (q_4,\Delta z_0)$

It reaches the final state "q_4", hence the string w="001111" accepted by PDA

$\vdash$ **Turnstile notation one move**

$\vdash^*$ **Sequence of moves**

HOME WORK

Design a PDA of a language $\mathbf{L=\{w\ w^R\ \ ,w\epsilon(0+1)^*\}}$

Design a PDA of a language $L=\{0^n 1^n,\ \ n>=1\}$

Design a PDA of a language $L=\{a^n b^m a^n,\ \ m, n>=1\}$

Design a PDA of a language $\mathbf{L=\{w\ Cw^R\ \ ,w\epsilon(0+1)^*\}}$

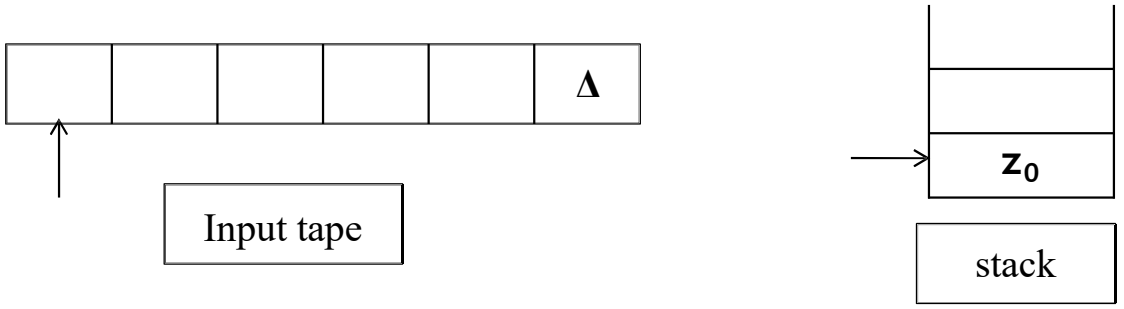
Example 3:

Design a PDA of a language L={well formed parenthesis}

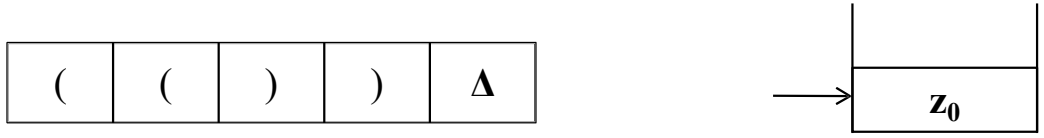
Solution:

Let us consider the string w ="(( ))"

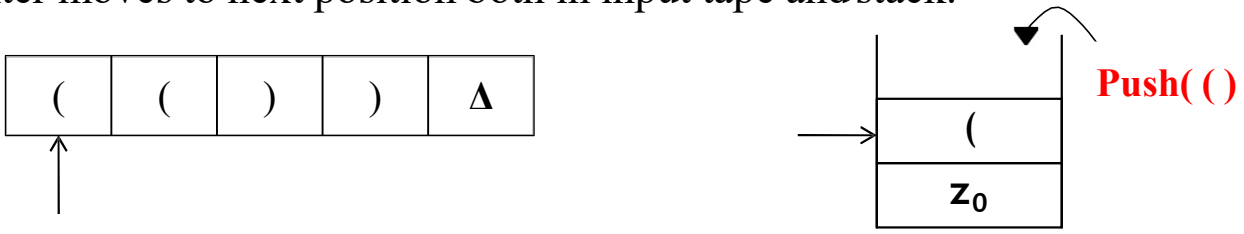
Now, the **input tape and stack** is prepared for doing operations with **pointer in input tape along with Δ as last element** and z_0 **as stack symbol with pointer**.



Step1: Insert the input in to the input tape ends with **Δ** and starts to read the input.



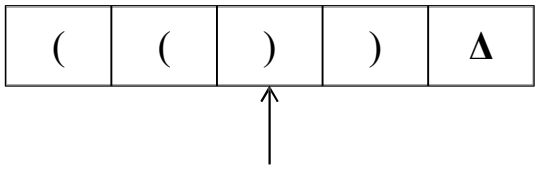
Step2: Read the input from tape, if it is '(' then perform **push(( )** into the stack. After this the pointer moves to next position both in input tape and stack.

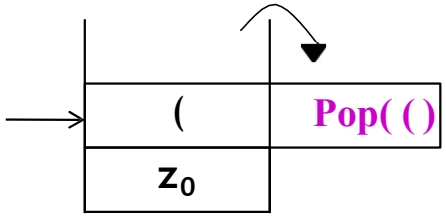


Step3: Move the pointer and read the next input from tape, if it is '(' then perform **push(( )** into the stack. The pointer moves to top of the stack.

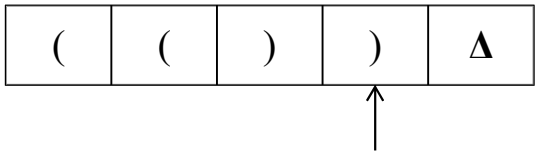


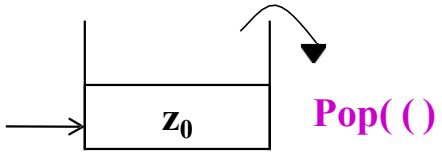
Step4: : Move the pointer and read the next input from tape, if it is ‘)’ then perform **pop(( )** from the stack. The pointer moves to top of the stack.





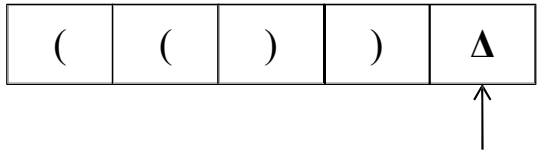
Step5: Move the pointer and read the next input from tape, if it is ‘)’ then perform **pop(( )** from the stack. The pointer moves to top of the stack.

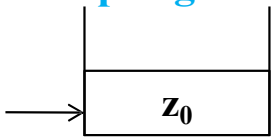




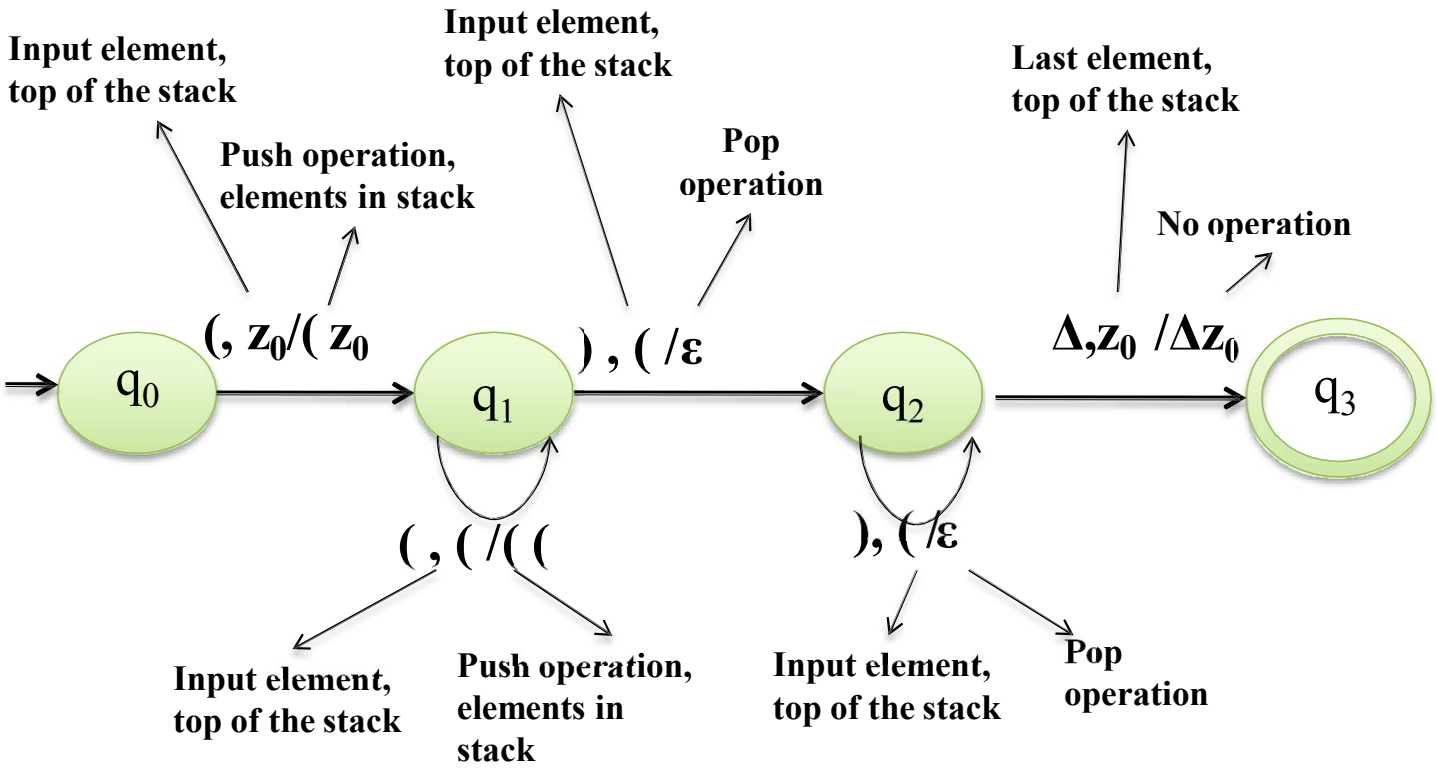
Step6: Move the pointer and read the next input, now the pointer points “**Δ” in input tape and z_0** in the stack.

This state remains and indicates that the PDA goes to an acceptingstate.

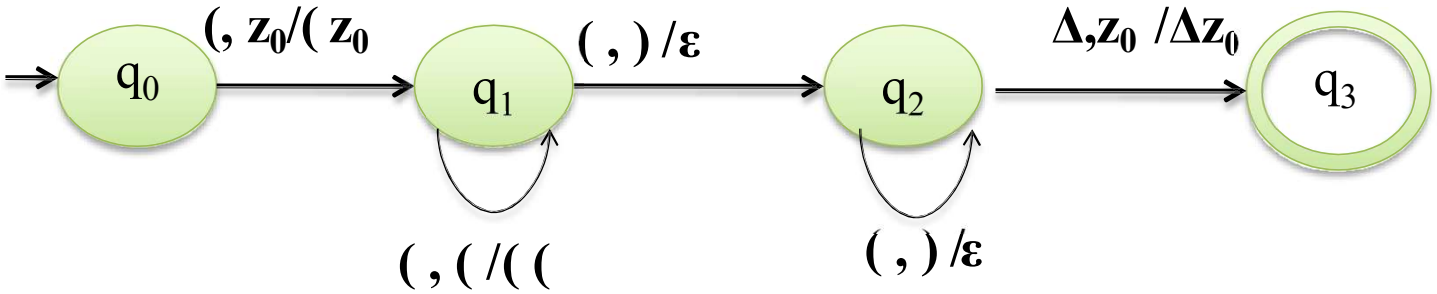


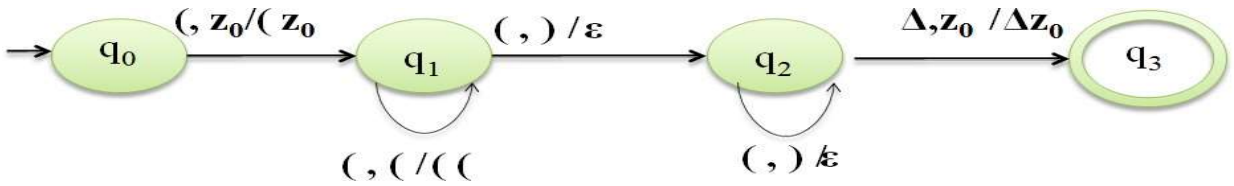


Step7: Transition Diagram



Step7: Transition Diagram





Step 8:Convention of Δ transition: $\Delta(q,w,z_0)=(p,\alpha)$

$\delta(q_0,(,z_0) = (q_1,(z_0)$

$\delta(q_1,(,)) = (q_1,(( )$

$\delta(q_1,(,)) = (q_2, \varepsilon)$

$\delta(q_2,(,)) = (q_2,\varepsilon)$

$\delta(q_2,\Delta,z_0) = (q_3,\Delta z_0)$

The given string reaches the final states and hence the string "w" is accepted.

Step 9:Instantaneous description in PDA

- It is represented as (State, String, Stack Element)

$(q_0,(( ))\Delta,z_0)$

$(q_0, (( ))\Delta,z_0) \vdash (q_1, ( )) \Delta, (z_0)$

$\vdash (q_2,)) \Delta,((z_0)$

$\vdash (q_2,)\Delta,(z_0)$

$\vdash (q_2,\Delta,z_0)$

$\vdash (q_3,\Delta z_0)$

It reaches the final state "q_3", hence the string w=" (( ))" accepted by PDA

Example 4:

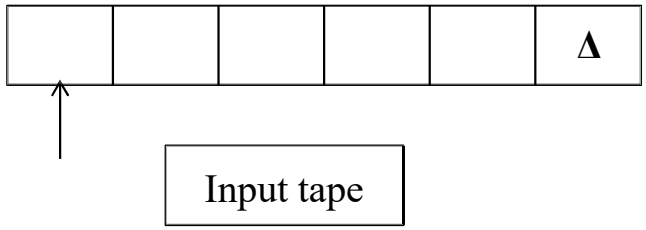
Design a PDA of a language **L={w w^R , w∈(0+1)*}**

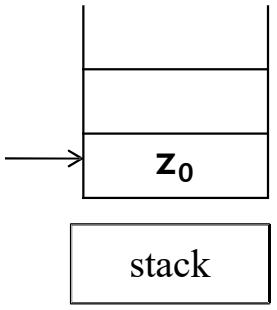
Solution:

Let the string **w**=0101 , then $\mathbf{w^R}$ = 1010

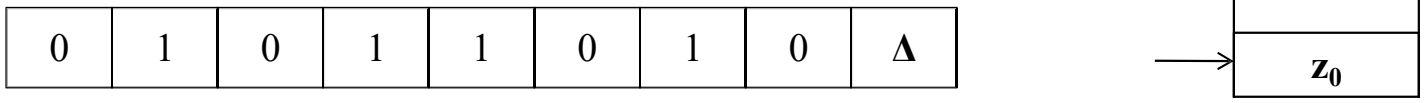
Therefore the string $\mathbf{ww^R}$ =01011010

Now, the **input tape and stack** is prepared for doing operations with **pointer in input tape along with Δ as last element** and z_0 **as stack symbol with pointer**.

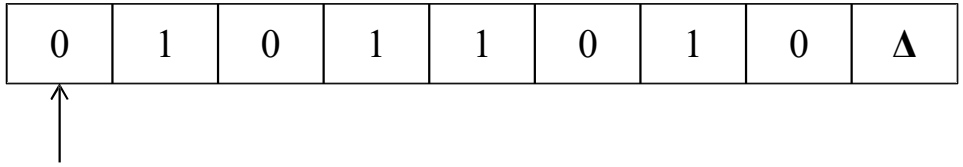


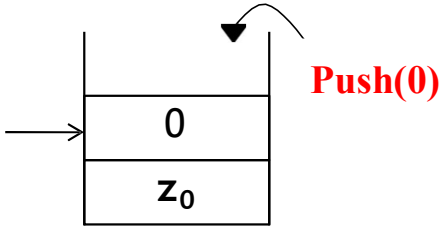


Step1: Insert the input in to the input tape ends with **Δ** and starts to read the input.

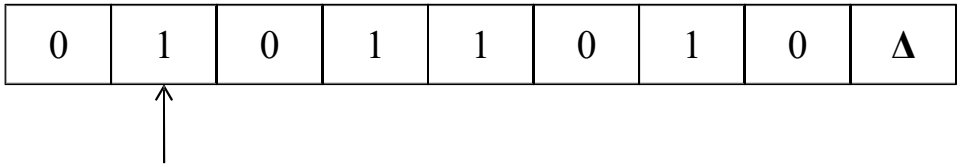


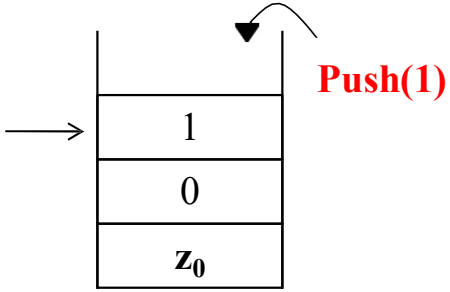
Step2: Read the input from tape, if it is '0' then perform **push(0)** into the stack. After this the pointer moves to next position both in input tape and stack.



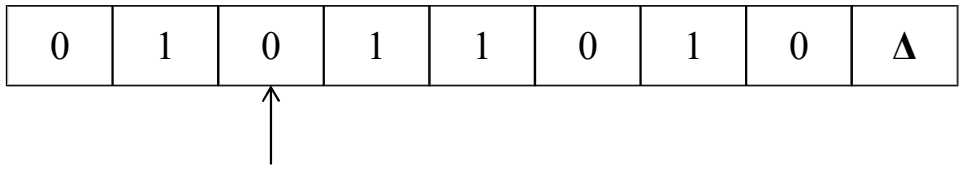


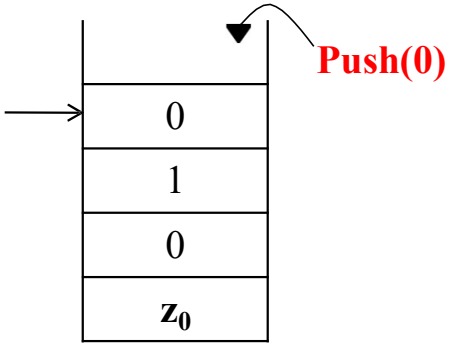
Step3: Move the pointer and read the next input from tape, if it is '1' then perform **push(1)** into the stack. The pointer moves to top of the stack.



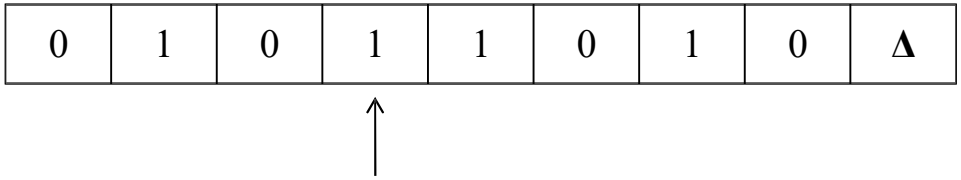


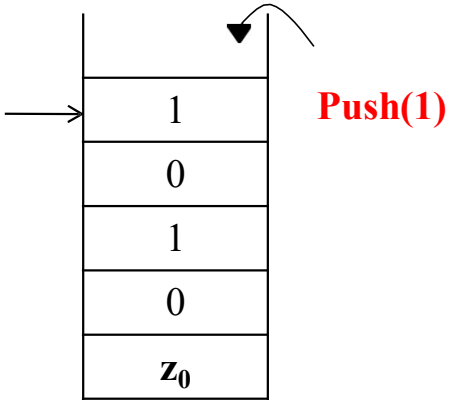
Step4: Move the pointer and read the next input, if it is '0' then perform **push(0)** into the stack. The pointer moves to top of the stack.



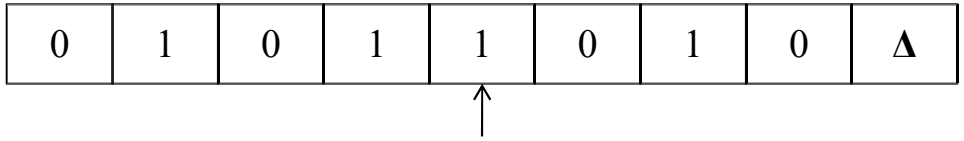


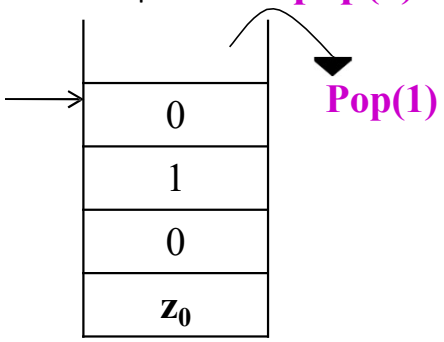
Step5: Move the pointer and read the next input from tape, if it is '1' then perform **push(1)** into the stack. The pointer moves to top of the stack.



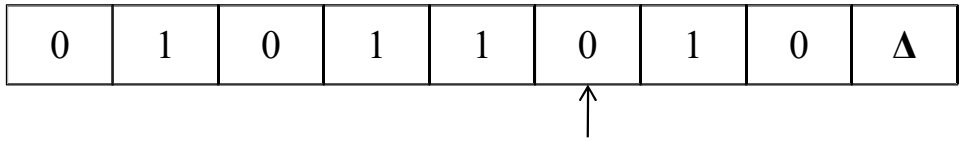


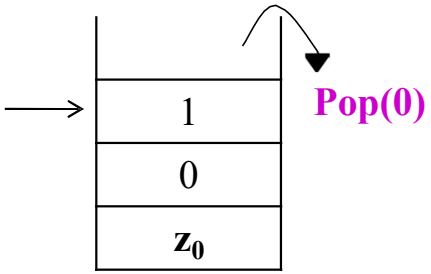
Step6: Move the pointer and read the next input, if it is '1' then perform **pop(1)** from the stack. The pointer moves to top of the stack.



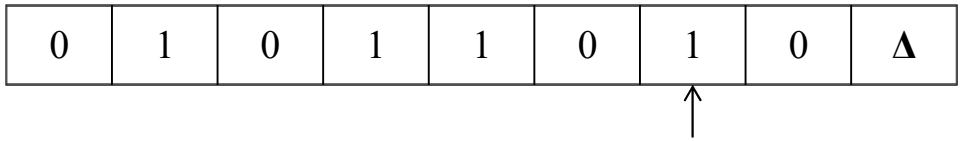


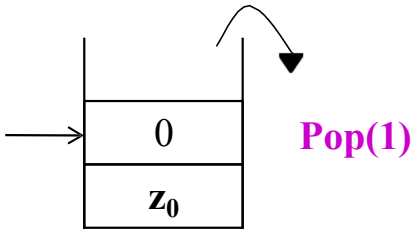
Step7: Move the pointer and read the next input, if it is '0' then perform **pop(0)** from the stack. The pointer moves to top of the stack.



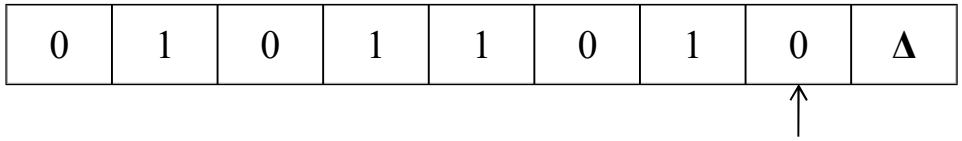


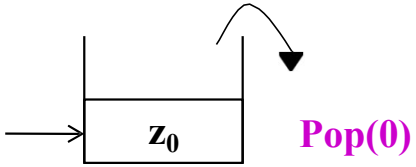
Step8: Move the pointer and read the next input, if it is '1' then perform **pop(1)** from the stack. The pointer moves to top of the stack.





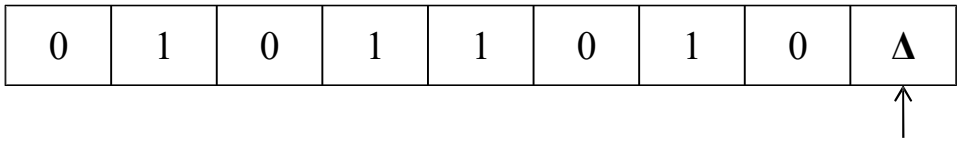
Step9: Move the pointer and read the next input, if it is '0' then perform **pop(0)** from the stack. The pointer moves to top of the stack.

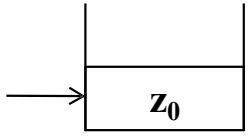




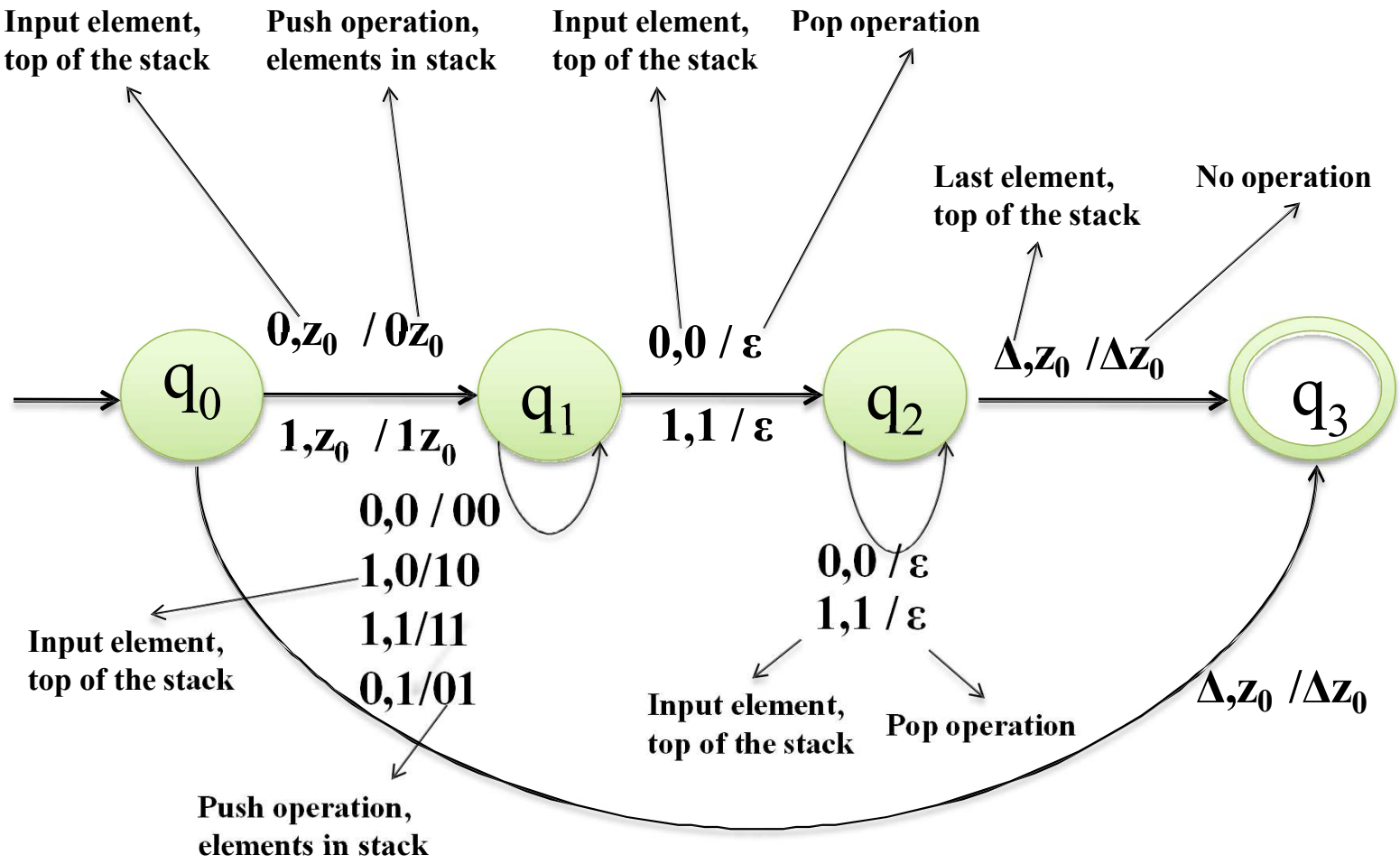
Step10: Move the pointer and read the next input, now the pointer points "**Δ" in input tape and z_0** in the stack.

This state remains and indicates that the PDA goes to an accepting state.



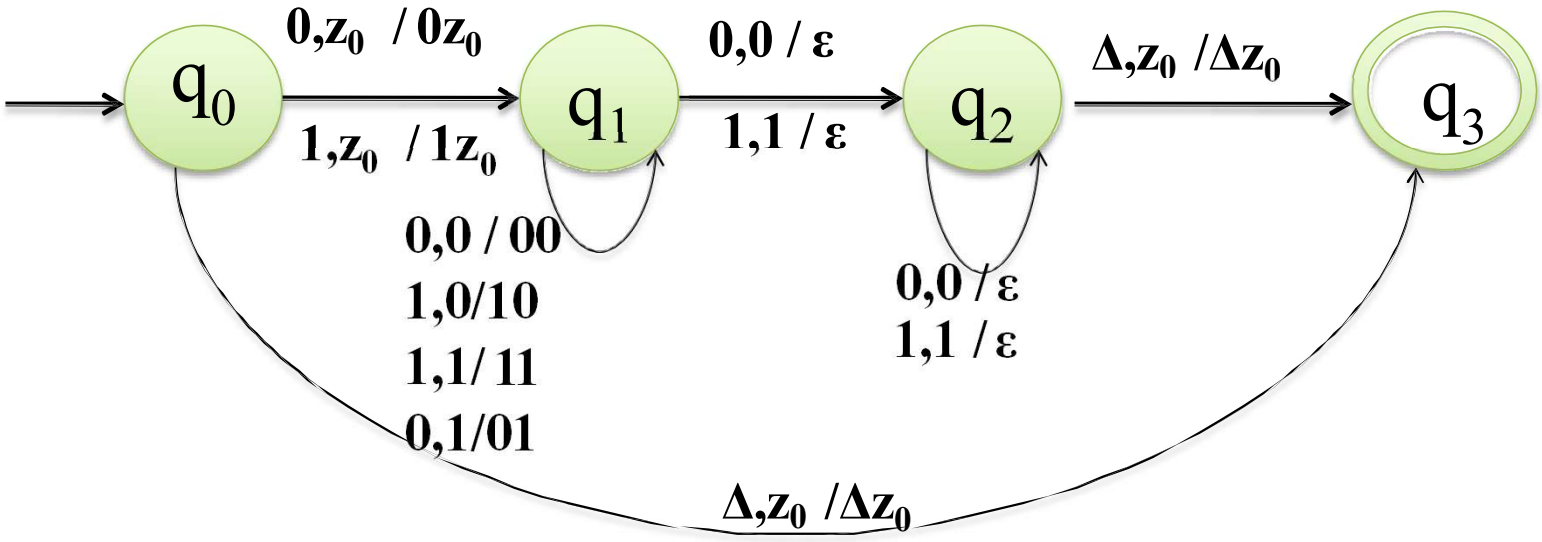


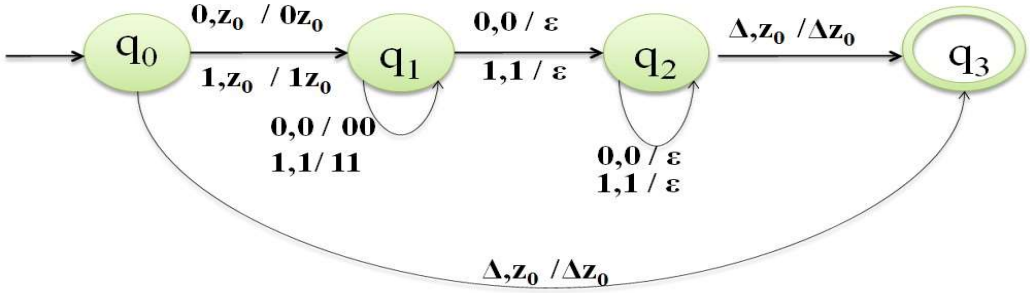
Step11: Transition Diagram



Step11: Transition Diagram

$L=\{w\, w^R , w(0+1)^*\}$





Step 12:Convention of Δ transition: $\Delta(q,w,z_0)=(p,\alpha)$ w=01011010

$\delta(q_0,0,z_0) = (q_1,0z_0)$

$\delta(q_1,1,0) = (q_1,10)$

$\delta(q_1,0,1) = (q_1,01)$

$\delta(q_1,1,0) = (q_1,10)$

$\delta(q_1,1,1) = (q_2,\varepsilon)$

$\delta(q_1,0,0) = (q_2,\varepsilon)$

$\delta(q_2,1,1) = (q_2,\varepsilon)$

$\delta(q_2,0,0) = (q_2,\varepsilon)$

$\delta(q_2,\Delta,z_0) = (q_3,\Delta z_0)$

The given string reaches the final states and hence the string "w" is accepted.

Step 13:Instantaneous description in PDA

- It is represented as (State, String, Stack Element)

$(q_0,01011010\Delta,z_0)$

$(\mathbf{q_0}, \mathbf{01011010}\Delta,z_0) \vdash (q_1,1011010\Delta,0z_0)$

$\vdash (q_1,011010\Delta,10z_0)$

$\vdash (q_1,11010\Delta,010z_0)$

$\vdash (q_1,1010\Delta,1010z_0)$

$\vdash (q_2,010\Delta,010z_0)$

$\vdash (q_2,10\Delta,10z_0)$

$\vdash (q_2,0\Delta,0z_0)$

$\vdash (q_2,\Delta,z_0)$

$\vdash (q_3,\Delta z_0)$

It reaches the final state "q_3", hence the string w="010110101" accepted by PDA

Applications of PDA

- **Compiler design**(parser design for syntactic analysis).
- **Online Transaction process system.**
- **Tower of Hanoi** (Recursive Solution).
- Handheld PDA supports **barcode scanning and data conveying.**
- **Predictive parsing- Bottom up LR parsing**