4. Push Down Automata

A Push Down Automata (PDA) is a way to implement a context free Grammar in a similar way we design finite Automata for regular grammar.

(OR)

→ The mathematical representation of CFL is called as PDA.

→ It is more powerful than FSM.

→ FSM has very limited memory but PDA has more memory.

→ PDA = Finite State Machine + Stack

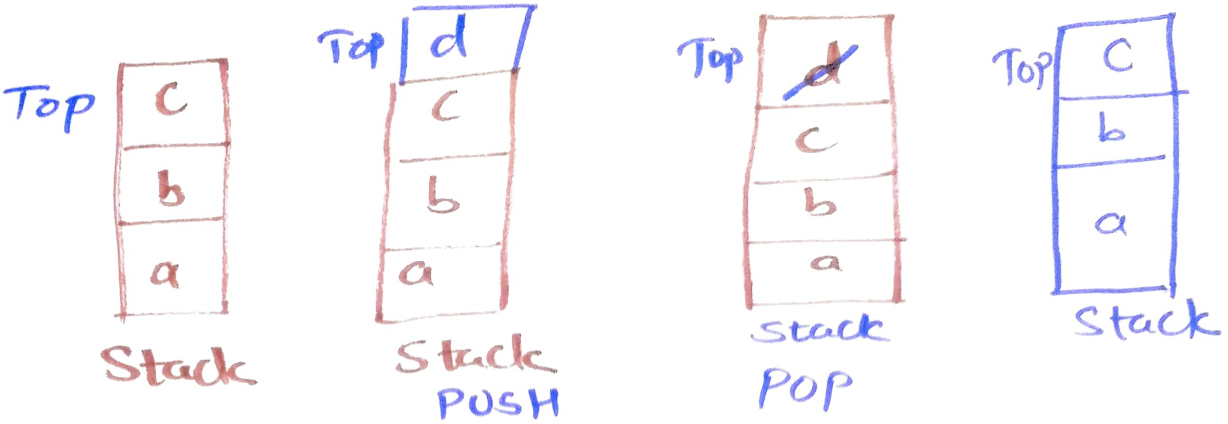
⇒ A Stack is used for storing all items temporarily. which is external storage to PDA.

→ A stack is a way we arrange elements one on top of another.

A stack does two basic operations:

PUSH :- A new element is added at the top of the stack

POP :- The top element of the stack is read & removed.



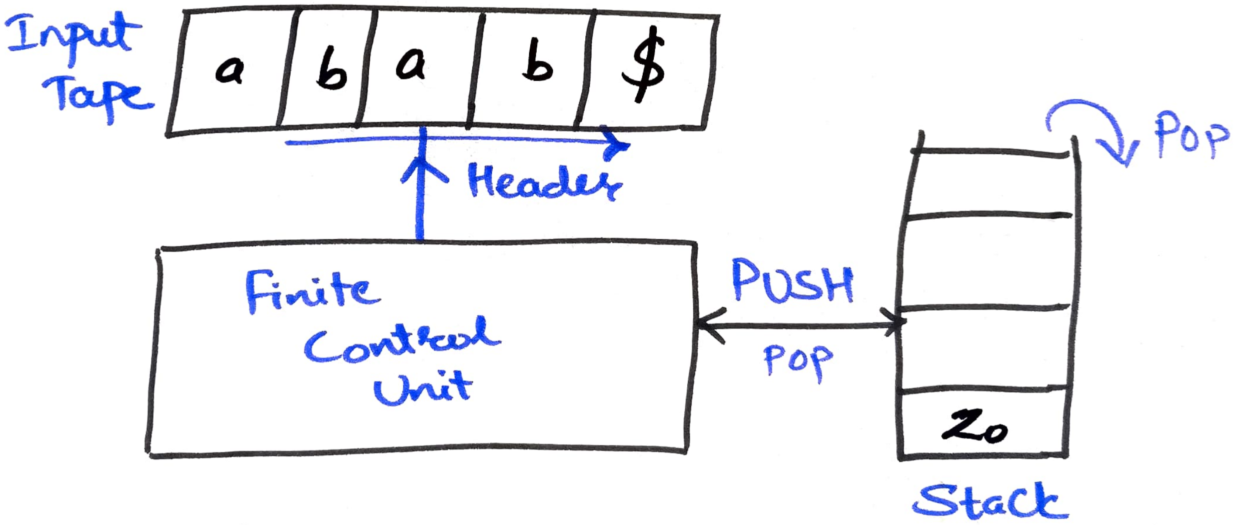
ex :- Pile of books

* A PDA has 3 Components (Block diagram)

1> Inpute tape [Tape header)

2> Finite Control Unit [FCU]

3> Stack with ~~a~~ infinite size.



-> No change in basic functionality of FA & PDA except stack element.

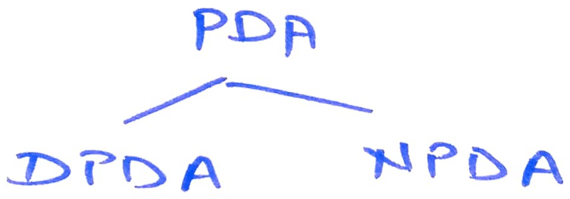
→ With the addition of stack PDA has got more accepting power Than FA but no change in the Computation power of FA & PDA.

→ Z_0 is the stack symbol & the purpose of Z_0 is only to see that the stack is empty (or) non-empty.

⇒ PDA can accept every language which is accepted by FA. & it also accept some of

⇒ the languages which are not accepted by FA.

⇒ The language which is accepted by PDA is CFL.



* PDA (Formal Definition)

PDA is defined by 7 tuples as shown below:-

$$P = \{Q, \Sigma, \Gamma, z_0, \delta, q_0, F\}$$

where,

$Q \Rightarrow$ Finite set of states

$\Sigma \Rightarrow$ Input symbols/alphabets

(toe) $\Gamma \Rightarrow$ Set of all stack symbols.

$z_0 \Rightarrow$ Start symbol from Stack

$\delta \Rightarrow$ Transition function

$$\boxed{\delta : Q \times \Sigma \times \Gamma \Rightarrow Q \times \Gamma^*}$$

$\underline{(\Sigma \cup \epsilon)}$

is a transition function.

$q_0 \Rightarrow$ Initial state

$F \Rightarrow$ The set of final states.

$$\delta(q, a, X) = (q, \underline{a\underline{b}})$$

X → Stack symbol from Γ

a → Input symbol (Σ)

q → State in Q.

q → New State

ab → New Topmost Symbol

$$\delta : \underset{\text{Current State}}{Q} \times \underbrace{(\Sigma \cup \epsilon)}_{\text{Input symbol including } \epsilon} \times \underbrace{\Gamma}_{\text{Stack symbol (Topmost)}} \Rightarrow \underline{Q \times \Gamma^*}$$

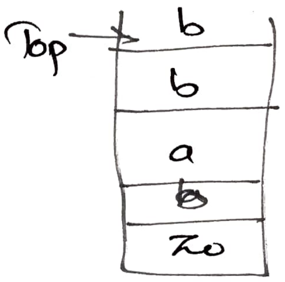
↳ Next state could be any state belonging to Q.

→ It can perform push, pop, or No-operation on stack.

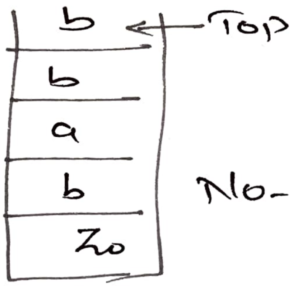
① Read input with No-operation on stack:-

$$\delta(\underset{\text{Current State}}{q_1}, \underset{\text{Current I/P Symbol}}{a}, \underset{\text{Topmost Stack Symbol}}{b}) = (q_2, b)$$

↳ Stack symbol b is replaced with b i.e. No stack operation (Read input)



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

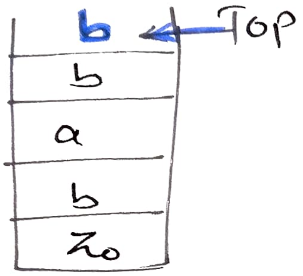


No-operation

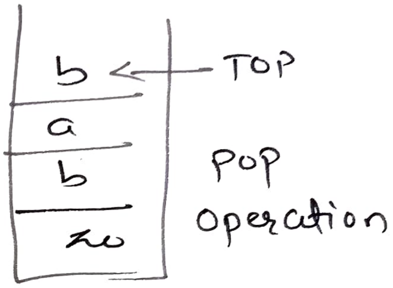
② POP operation:-

$$\delta(q_1, a, \underline{b}) = (q_2, \underline{\epsilon})$$

The above transition will erase the stack symbol. (Topmost). Replacing b with ϵ for erasing b from stack.



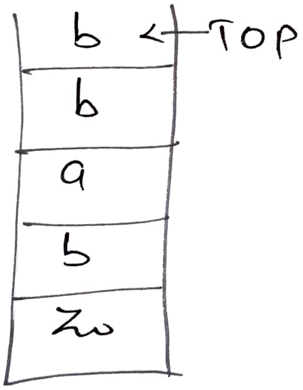
$$\delta(q_1, a, b) = \delta(q_2, \epsilon)$$



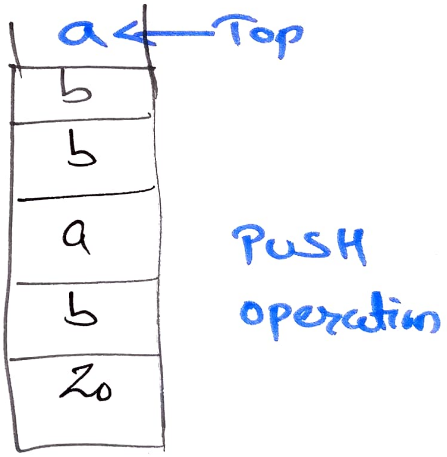
③ PUSH operation:-

$$\delta(q_1, a, \underline{b}) = (q_2, \underline{ab})$$

The above transition will perform push operation It will push 'a' onto the stack. Replacing 'b' with ab to push 'a' on to the stalk



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


a
Top
b
b
a
b
Zo
PUSH
operation