# Chapter # 8
# CFG = PDA

Dr. Shaukat Ali
Department of Computer Science
University of Peshawar

# Converting CFG into PDA.

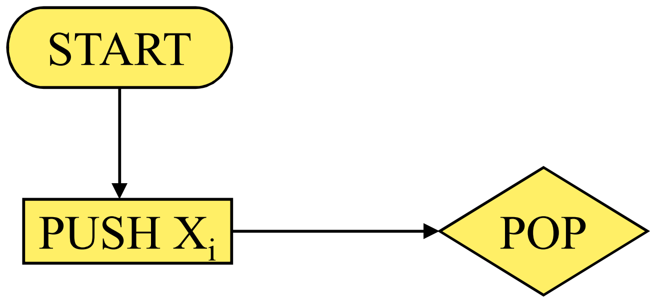
- If we are given a CFG in CNF as follows:

$$
\begin{array}{lll}
X_1 & \rightarrow & X_2X_3 \\
X_2 & \rightarrow & X_4X_5 \\
 & ---- & \\
X_3 & \rightarrow & a \\
X_4 & \rightarrow & a \\
X_5 & \rightarrow & b
\end{array}
$$

- Where the start symbol $S = X_1$ and other nonterminals are $X_2$, $X_3$, ----.
- We can use the following algorithm to construct PDA.

# Algorithm for converting CFG into PDA.

- If $X_i$ is the start symbol then convert in to the following PDA.

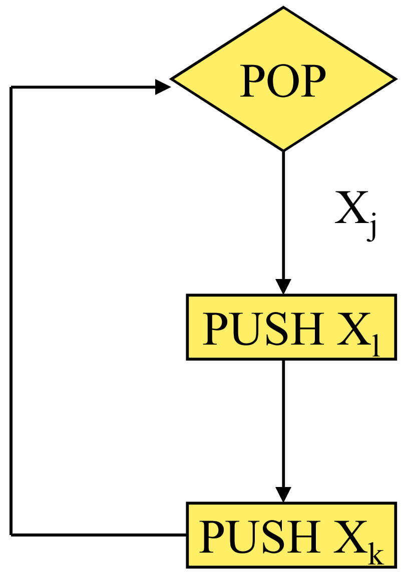



- For each production of the form

$$X_j \rightarrow X_k X_l$$

we include the circuit from the POP back to itself.

# Algorithm for converting CFG into PDA.

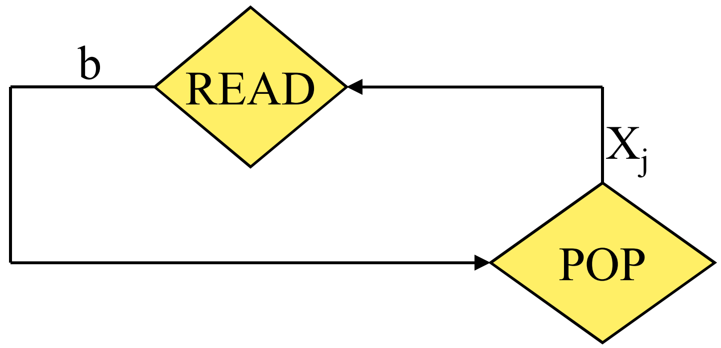



- For all production of the form

$$X_j \rightarrow b$$

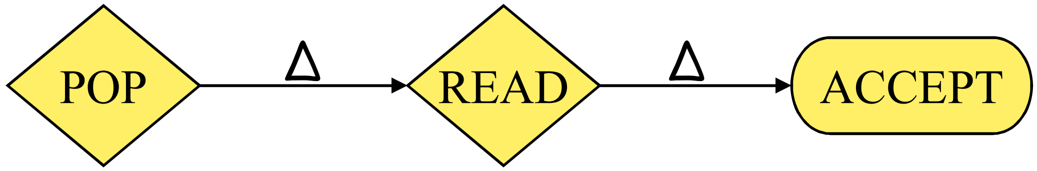
we include this circuit.

# Algorithm for converting CFG into PDA.



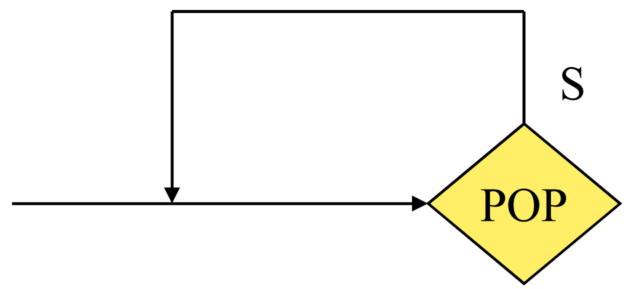

- When the stack is empty, which means that we have converted our last nonterminal to a terminal and the terminals have matched the INPUT TAPE, we should include the following circuit.

# Algorithm for converting CFG into PDA.




- By using this algorithm, we can assure that all words generated by the CFG will be accepted by the resultant PDA machine.
- That is all for a grammar that is in CNF. But there are context-free languages that cannot be put into CNF.
- In this case we can convert all productions into one of the two forms acceptable by CNF, while the word Λ must still be included.
- To include this word, we need to add another circuit to the PDA, a simple loop at the POP.

# Algorithm for converting CFG into PDA.




- This kills the nonterminal S without replacing with anything and the next time we enter the POP, we get a blank and proceed to accept the word.

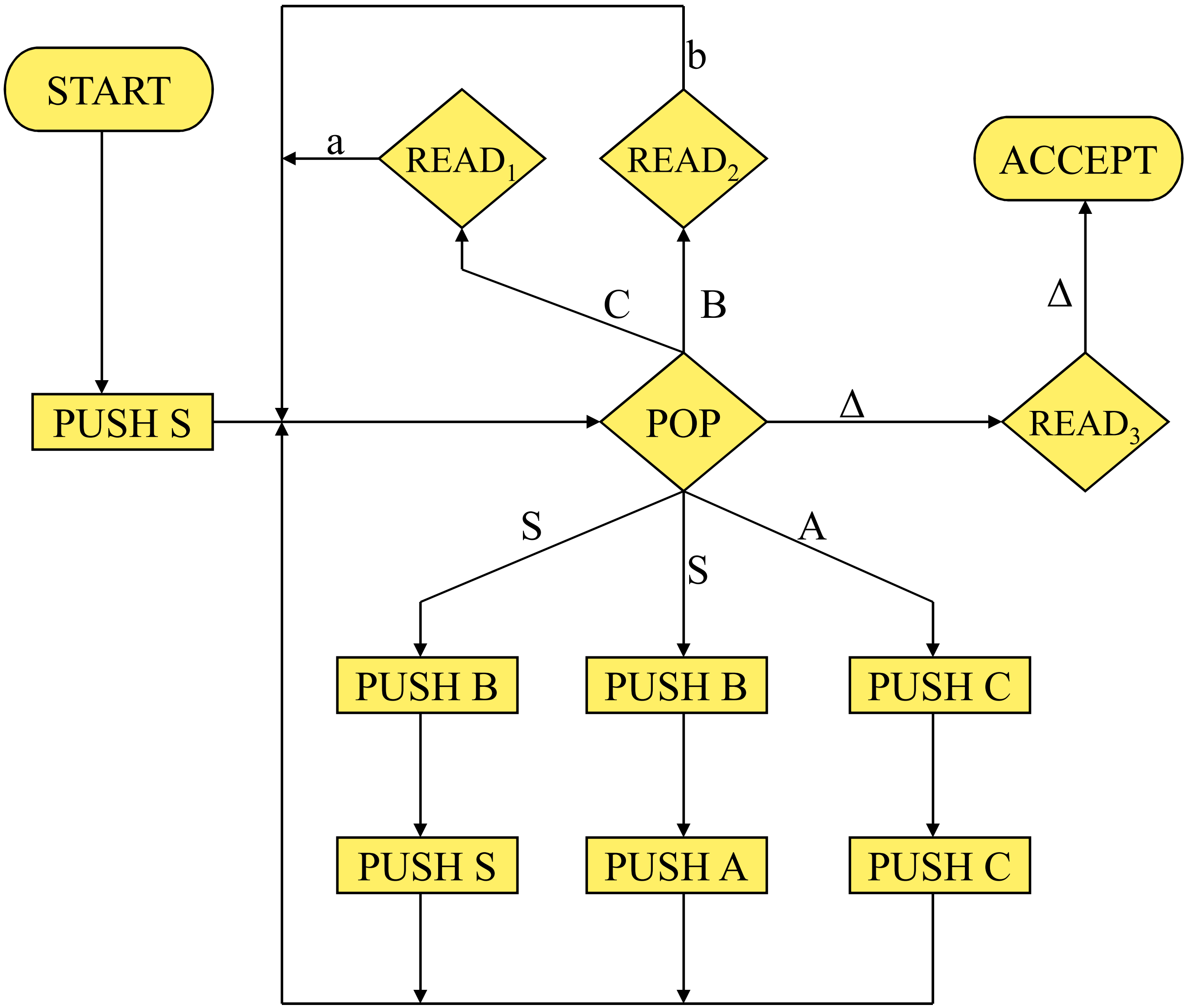
# Example.

- Consider the following grammar which is in CNF.

$$S \rightarrow SB$$
$$S \rightarrow AB$$
$$A \rightarrow CC$$
$$B \rightarrow b$$
$$C \rightarrow a$$

convert this grammar into equivalent PDA.

# Resulting PDA.



# Left-most derivation.

- Lets consider an example aab and derive it by using left-most derivation using the grammar.

| Working string generation. | | | Productions used. | | | |
|---|---|---|---|---|---|---|
| S | ==> | AB | S | → | AB | Step 1 |
| | ==> | CCB | A | → | CC | Step 2 |
| | ==> | aCB | C | → | a | Step 3 |
| | ==> | aaB | C | → | a | Step 4 |
| | ==> | aab | B | → | b | Step 5 |

# Left-most derivation.

- Now if we simulate the same string aab by using the resultant PDA, we will see this derivation is also left-most.
- At each step we will nonterminals on the STACK as that we have in working string generation in the left-most derivation.
- It means that if we construct PDA for a CFG by using the above algorithm, the derivation will be left most derivation.

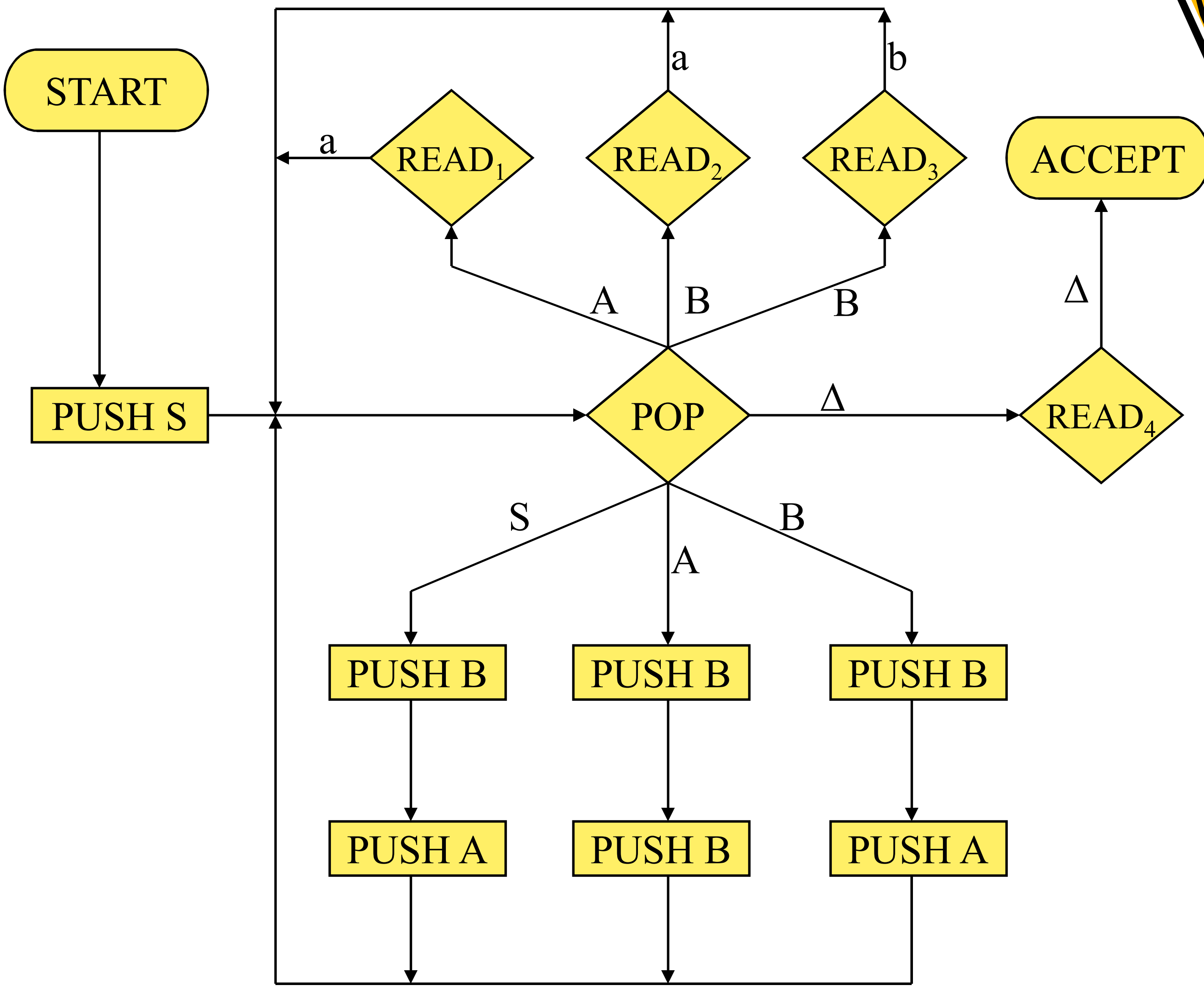
# Example.

- Consider the the following CFG, which is in CNF.

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow BB \\
B &\rightarrow AB \\
A &\rightarrow a \\
B &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Construct PDA for this grammar.

# Resulting PDA.



# Exercise.

- Try deriving the string baaab by using both left-most derivation and simulating by using the resultant PDA.

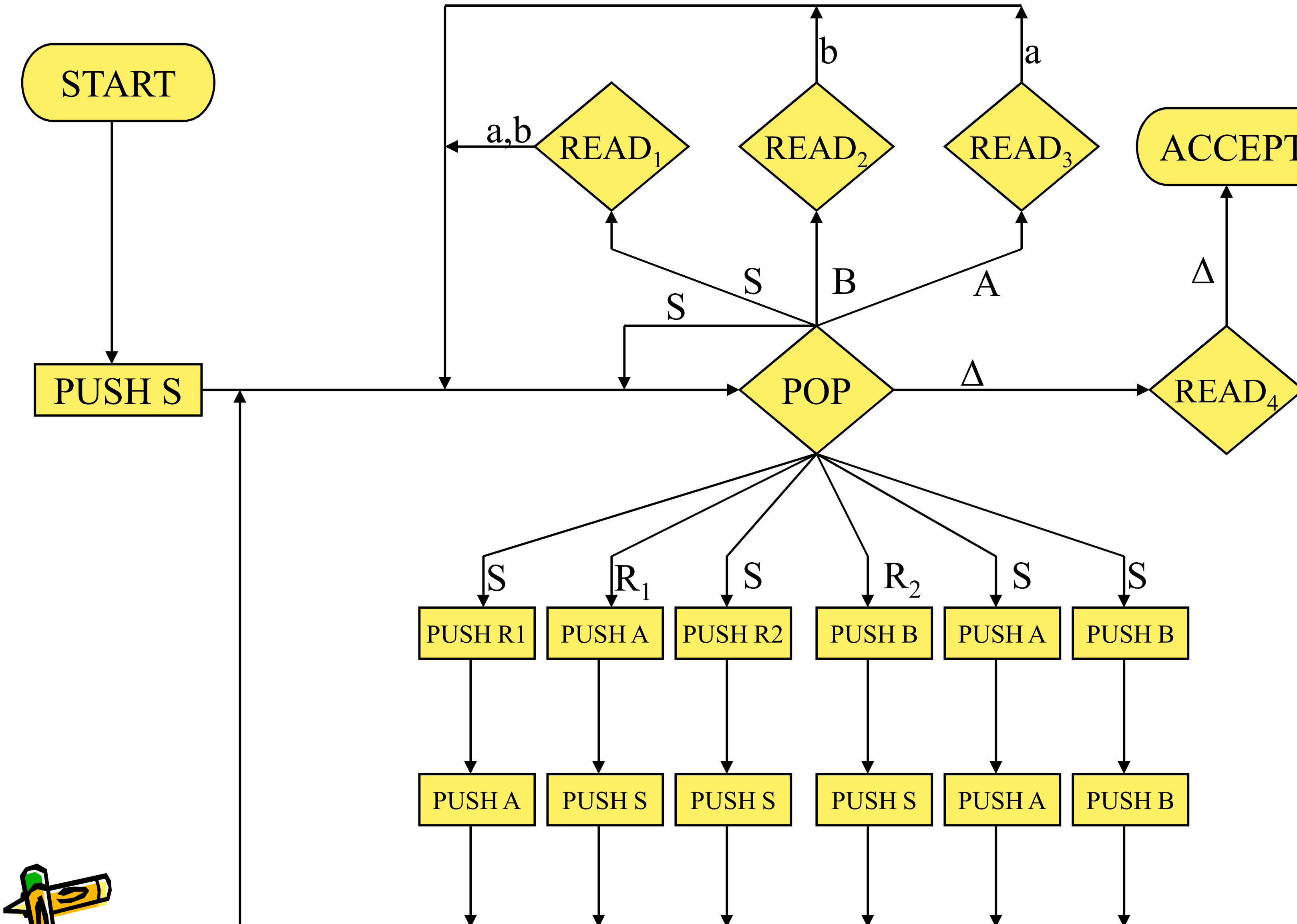
# Example.

- Consider the following CFG.

| | | |
|---|---|---|
| $S \rightarrow AR_1$ | | $A \rightarrow a$ |
| $R_1 \rightarrow SA$ | | $B \rightarrow b$ |
| $S \rightarrow BR_2$ | | $S \rightarrow \Lambda$ |
| $R_2 \rightarrow SB$ | | |
| $S \rightarrow AA$ | | |
| $S \rightarrow BB$ | | |
| $S \rightarrow a$ | | |
| $S \rightarrow b$ | | |

- Construct PDA.

# Resulting PDA.



# Example.

- Let us convert the CFG

  $S \rightarrow bA \mid aB$

  $A \rightarrow bAA \mid aS \mid a$

  $B \rightarrow aBB \mid bS \mid b$

  Construct PDA for this grammar.
- As this grammar is not in CNF, therefore:
  1. Convert this grammar into CNF.
  2. Construct PDA for the CNF.

# Step 1: Convert grammar into CNF.

- As this grammar has only two terminal symbols a and b. Therefore we consider two new nonterminals X and Y.
- First convert the production rules in to the standard production forms. The grammar becomes

$$S \rightarrow YA$$
$$S \rightarrow XB$$
$$A \rightarrow YAA$$
$$A \rightarrow XS$$
$$A \rightarrow a$$
$$B \rightarrow XBB$$
$$B \rightarrow YS$$
$$B \rightarrow b$$
$$X \rightarrow a$$
$$Y \rightarrow b$$

# Step 1: Convert grammar into CNF.

- Now convert these productions into the CNF.
- If a production rule has exactly two nonterminals or a terminal symbol on the RHS, ignore them and consider all the others.
- After conversion the grammar in CNF becomes.

$S \rightarrow YA \mid XB$

$A \rightarrow YR_1 \mid XS \mid a$

$R_1 \rightarrow AA$

$B \rightarrow XR_2 \mid YS \mid b$

$R_2 \rightarrow BB$

$X \rightarrow a$

$Y \rightarrow b$

# Step 2: Convert CNF into PDA.

- Convert into PDA in class room.

# Convert the following into PDA.

1. (i) $S \rightarrow aSbb \mid abb$
   (ii) $S \rightarrow SS \mid a \mid b$

2. (i) $S \rightarrow XaaX$
   (ii) $X \rightarrow aX \mid bX \mid \Lambda$

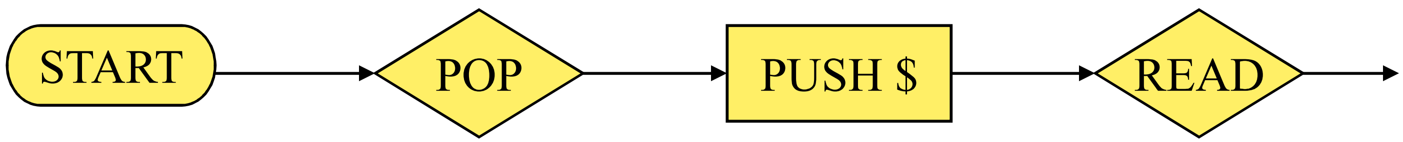
3. (i) $S \rightarrow XY$
   (ii) $X \rightarrow aX \mid bX \mid a$
   (iii) $Y \rightarrow Ya \mid Yb \mid a$

# Building a CFG for every PDA.

- A PDA is in conversion form, if it meets all the following conditions.
  - There is only one ACCEPT state.
  - There are no REJECT state.
  - Every READ state is followed immediately by a POP, that is, every edge leading out of any READ state goes directly into a POP state.
  - No two POP exist in a row on the same path without a READ or HERE states between them whether or not there are any intervening PUSH states. (POPs must be separated by READ states).
  - Every edge has only one label (no multiple labels).

# Building a CFG for every PDA.

- Even before we get to START, a "bottom of STACK" symbol $, is placed on the STACK. The STACK is never popped beneath this symbol. Right before entering ACCEPT this symbol is popped and left out.
- The PDA must begin with the sequence.



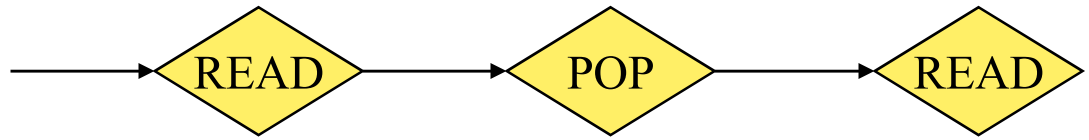

- The entire input string must be read before the machine can accept the word.

# Building a CFG for every PDA.

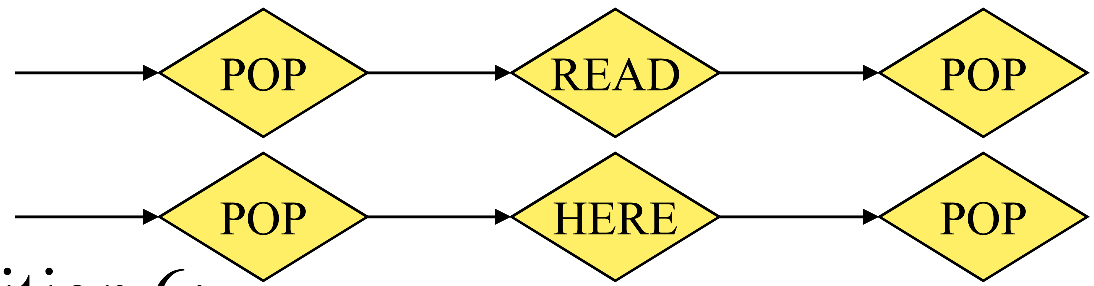
- Condition 1:
  - Condition 1 is easy to accommodate. If we have a PDA with several ACCEPT states. Let us simplify erase all but one of them and have all the edges that formerly went into the others feed into the one remaining.
- Condition 2:
  - Condition 2 is easy because we are dealing with nondeterministic machines.
  - If we are at a state with no edge labeled with the character we have just read or popped, we simply crash.
  - For an input string to be accepted, there must be a safe path to ACCEPT, the absence of such a path is termed as REJECT.
  - Therefore, we can erase all REJECT states and the edges leading to them without effecting the language accepted by the PDA.

# Building a CFG for every PDA.

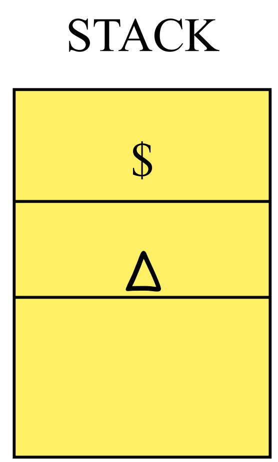
- Condition 3:




- Condition 4:




- Condition 6:

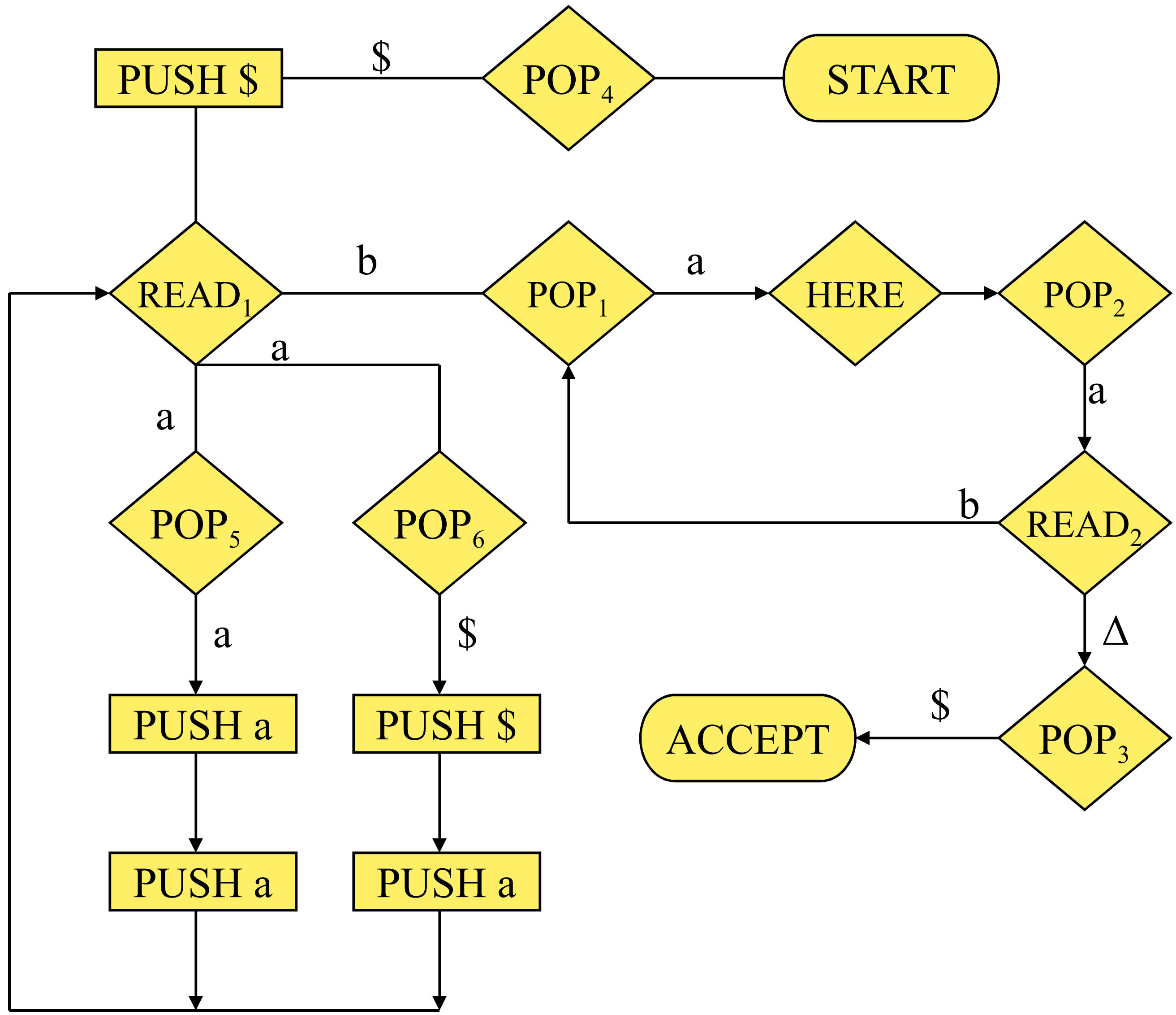


# Example.

- PDA in the conversion form. The PDA we use is one that accepts the language.

$$[a^{2n}b^{n}] = [\ aab,\ aaaabb,\ aaaaaabbb, --- \ ]$$

# Example.


PUSH $
$
POP4
START
READ1
b
POP1
a
HERE
POP2
a
a
a
POP5
POP6
b
READ2
a
$
Δ
PUSH a
PUSH $
ACCEPT
$
POP3
PUSH a
PUSH a

- End of Chapter # 8