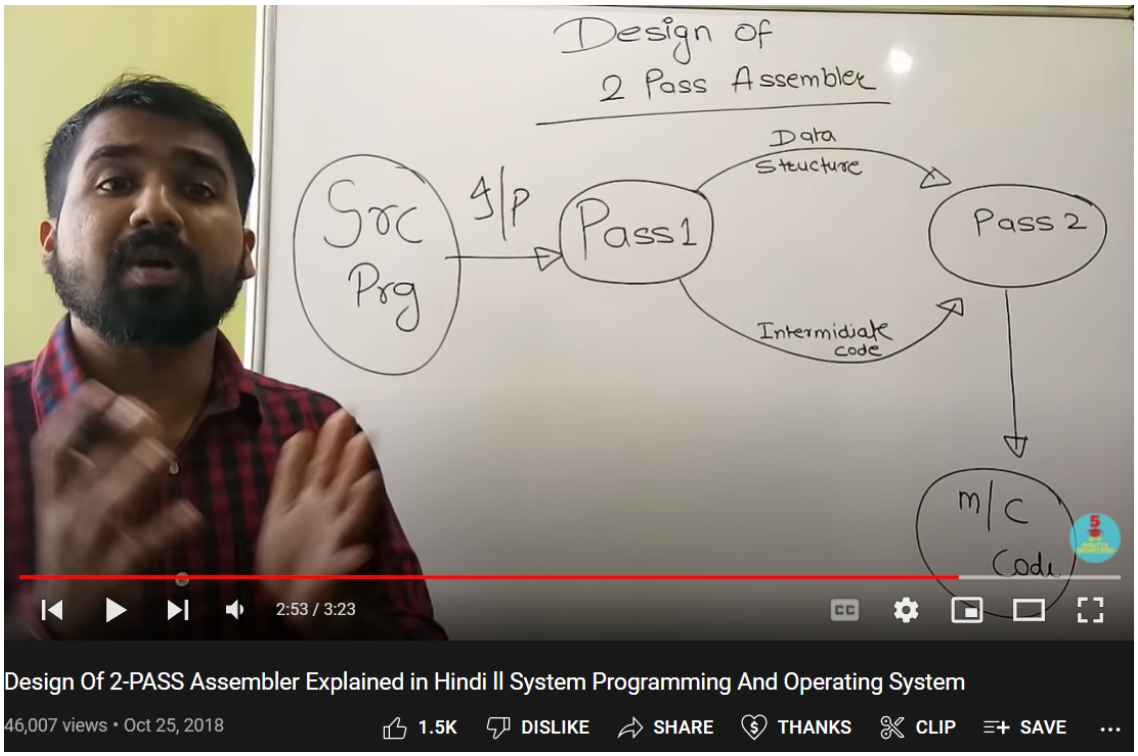
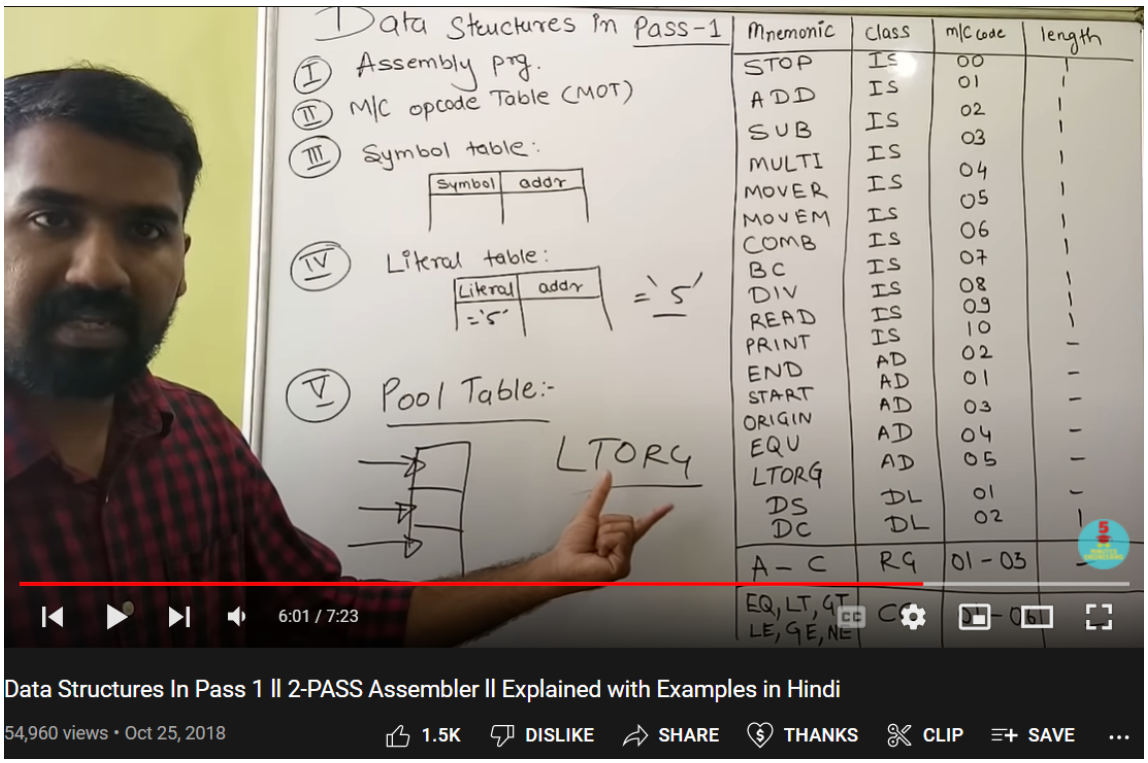
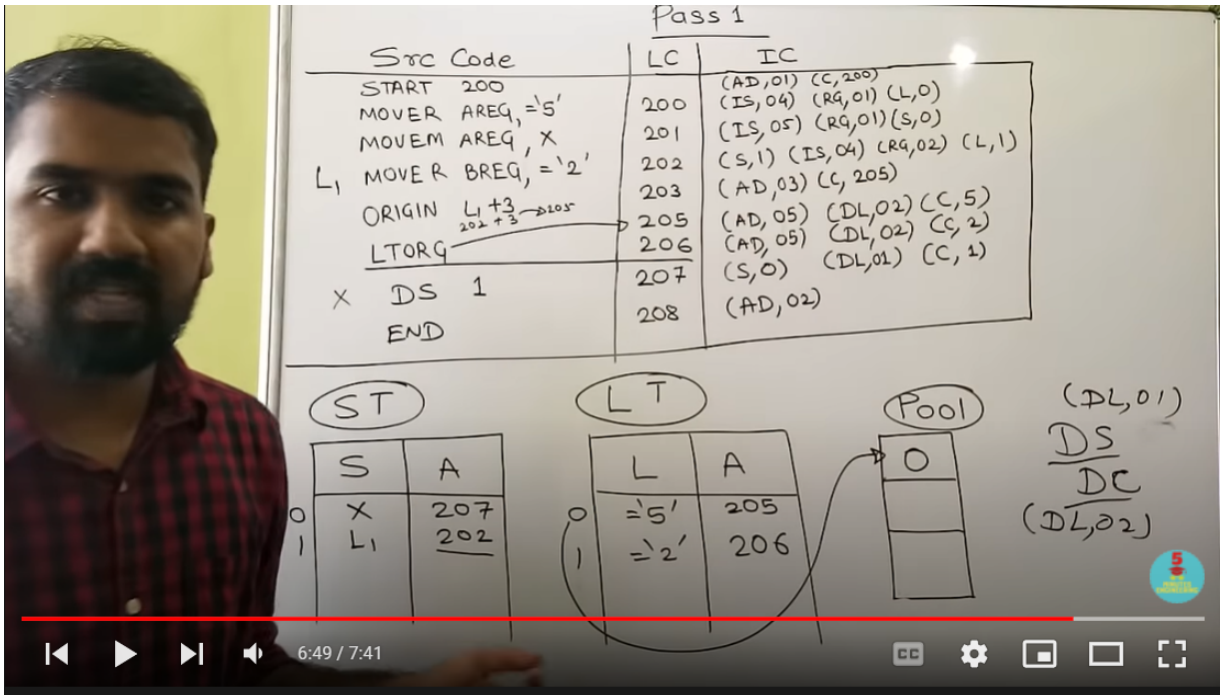
SPCC



Design Of 2-PASS Assembler Explained in Hindi ll System Programming And Operating S…



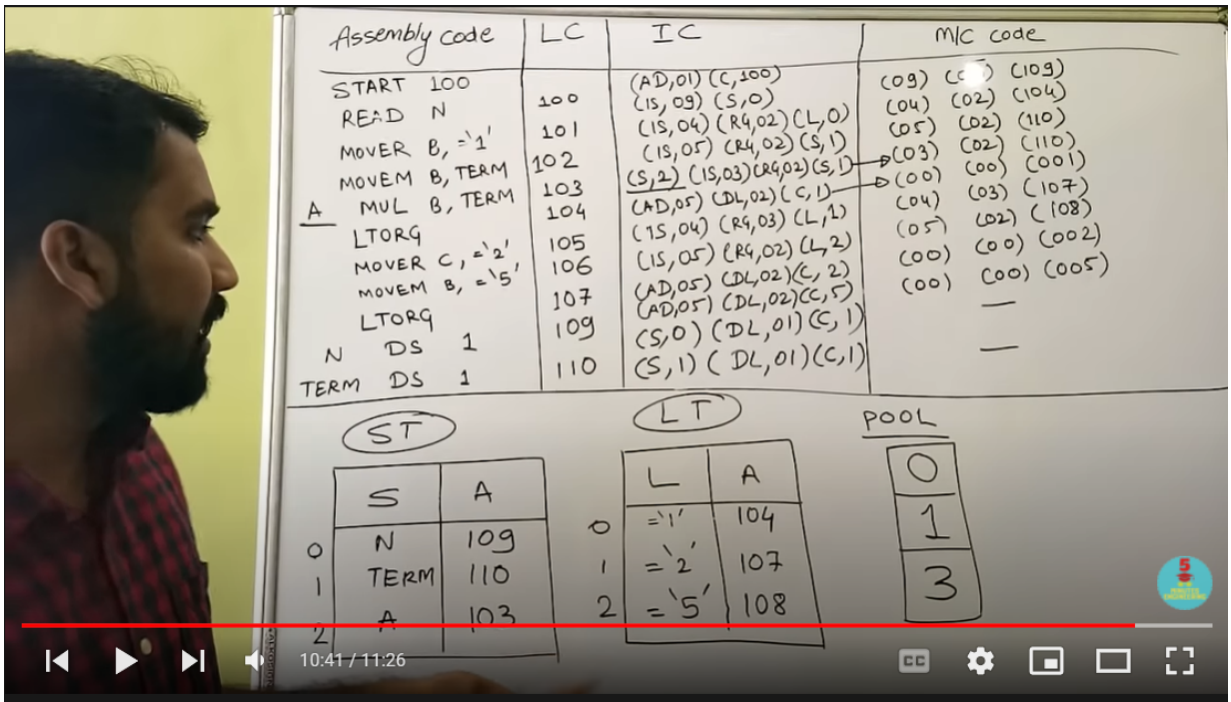
Data Structures In Pass 1 ll 2-PASS Assembler ll Explained with Examples in Hindi



PASS-1 Of 2-PASS Assembler Explained with Solved Example in Hindi ll Part-2 ll SPOS

43,802 views • Oct 25, 2018 1.6K DISLIKE SHARE THANKS CLIP SAVE ...

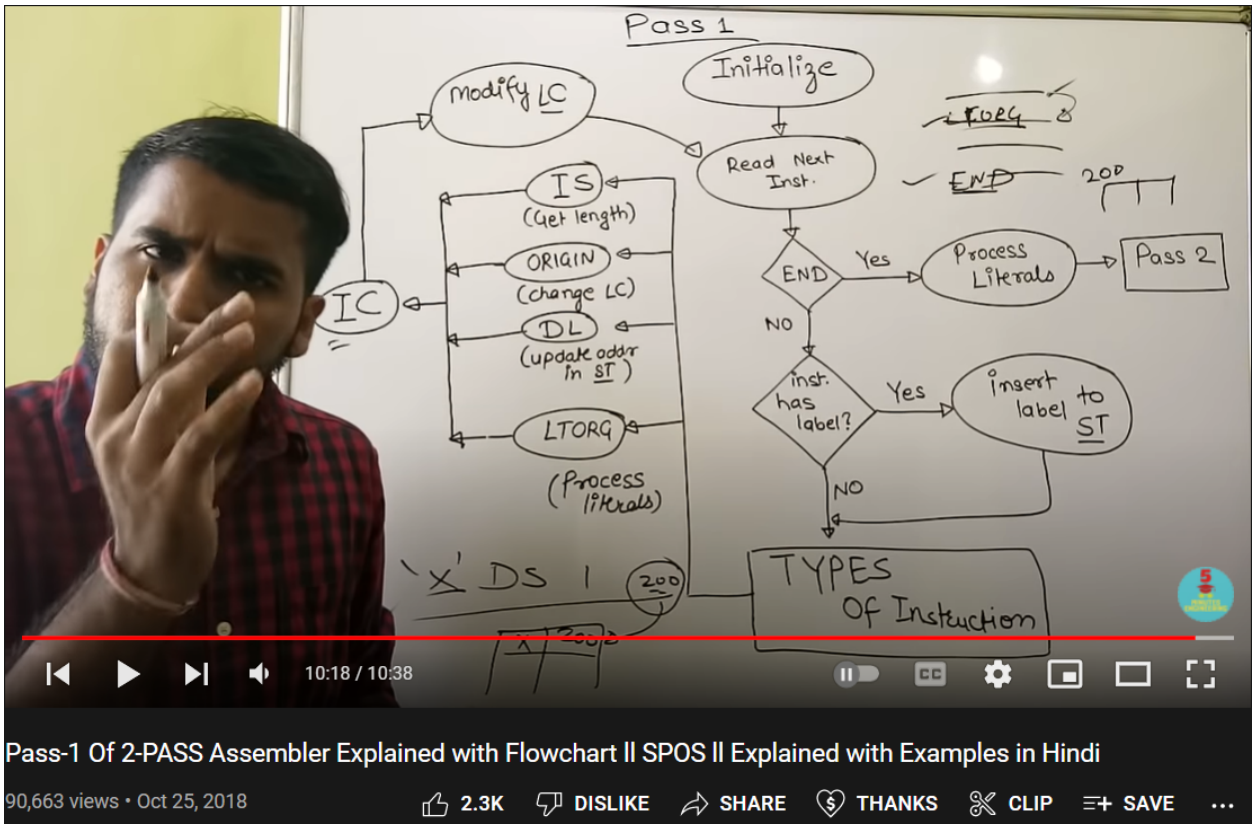
PASS-1 Of 2-PASS Assembler Explained with Solved Example in Hindi ll Part-2 ll SPOS



PASS-2 Of 2-PASS Assembler Explained with Solved Example in Hindi ll SPOS

45,242 views • Oct 25, 2018 1K DISLIKE SHARE THANKS CLIP SAVE ...

PASS-2 Of 2-PASS Assembler Explained with Solved Example in Hindi ll SPOS



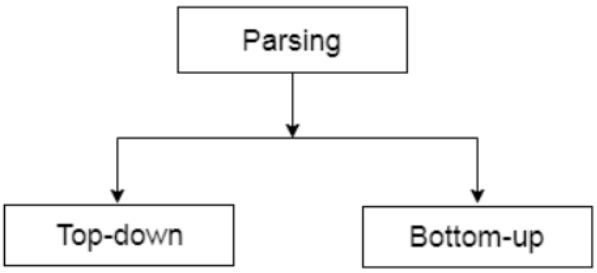
Pass-1 Of 2-PASS Assembler Explained with Flowchart ll SPOS ll Explained with Exampl…

Phaser:

Parser is a compiler that is used to break the data into smaller elements coming from lexical analysis phase.

A parser takes input in the form of sequence of tokens and produces output in the form of parse tree.

Parsing is of two types: top down parsing and bottom up parsing.



Top down paring

- The top down parsing is known as recursive parsing or predictive parsing.
- Bottom up parsing is used to construct a parse tree for an input string.
- In the top down parsing, the parsing starts from the start symbol and transform it into the input symbol.

Bottom up parsing

- Bottom up parsing is also known as shift-reduce parsing.
- Bottom up parsing is used to construct a parse tree for an input string.
- In the bottom up parsing, the parsing starts with the input symbol and construct the parse tree up to the start symbol by tracing out the rightmost derivations of string in reverse.

LR Parser

LR parsing is one type of bottom up parsing. It is used to parse the large class of grammars.

In the LR parsing, "L" stands for left-to-right scanning of the input.

"R" stands for constructing a right most derivation in reverse.

"K" is the number of input symbols of the look ahead used to make number of parsing decision.

LR parsing is divided into four parts: LR (0) parsing, SLR parsing, CLR parsing and LALR parsing.

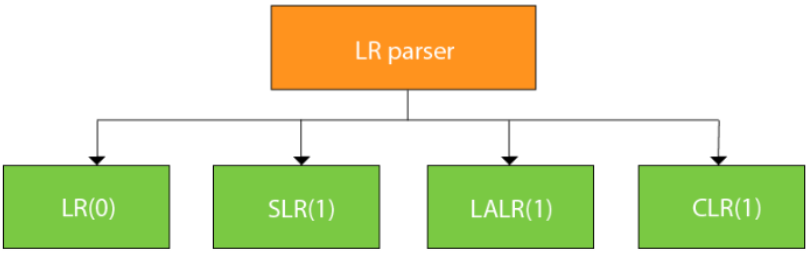


Fig: Types of LR parser

LR algorithm:

The LR algorithm requires stack, input, output and parsing table. In all type of LR parsing, input, output and stack are same but parsing table is different.

Construction of LL(1) Parsing Table

Difficulty Level : Medium • Last Updated : 03 Jun, 2021

Prerequisite – Classification of top-down parsers, FIRST Set, FOLLOW Set

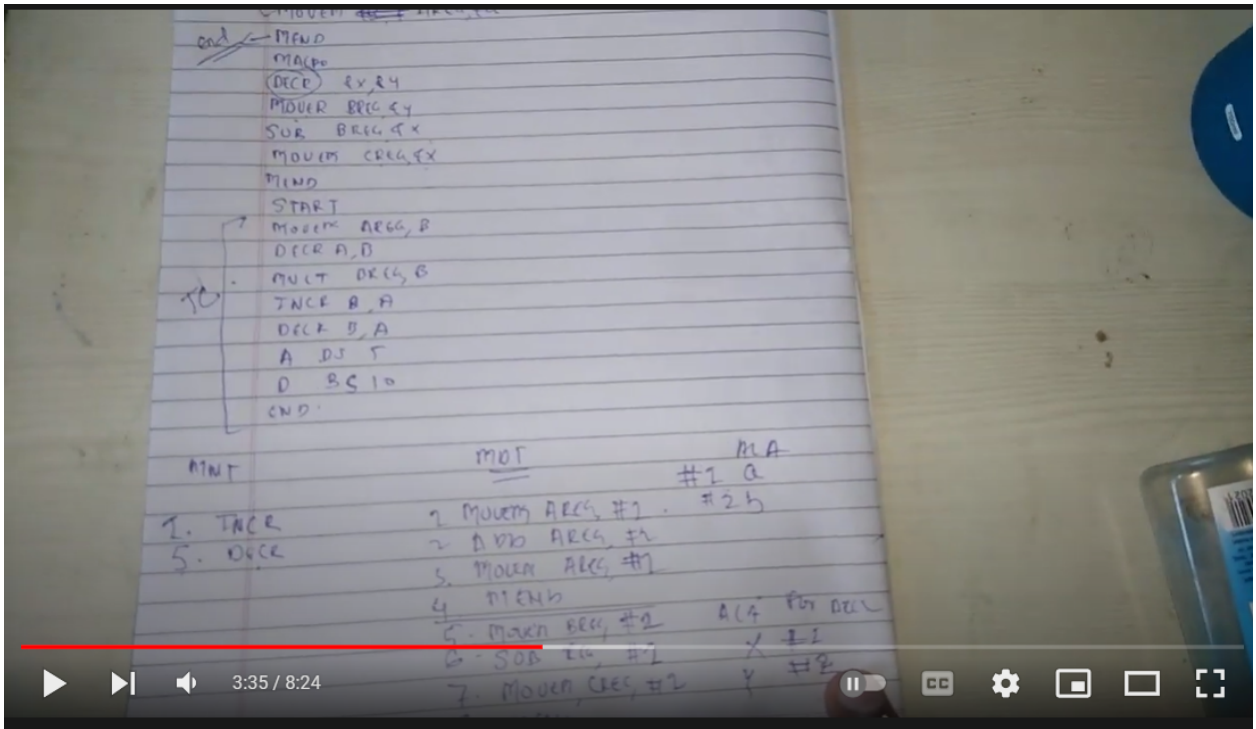
A top-down parser builds the parse tree from the top down, starting with the start non-terminal. There are two types of Top-Down Parsers:

1. Top-Down Parser with Backtracking
2. Top-Down Parsers without Backtracking

LL(1) Parsing:

Here the 1st **L** represents that the scanning of the Input will be done from Left to Right manner and the second **L** shows that in this parsing technique we are going to use Left most Derivation Tree. And finally, the **1** represents the number of look-ahead, which means how many symbols are you going to see when you want to make a decision.

LL(1) parsing is a top-down parsing method in the syntax analysis phase of compiler design. Required components for LL(1) parsing are input string, a stack, parsing table for given grammar, and parser. Here, we discuss a parser that determines that given string can be generated from a given grammar(or parsing table) or not.



Macro-Processor.MNT,MDT,ALA,IC,EXPANDED.

1,312 views • Feb 27, 2020 35 DISLIKE SHARE SAVE ...

Specify the problem

- In Pass-I the macro definitions are searched and stored in the macro definition table and the entry is made in macro name table
- In Pass-II the macro calls are identified and the arguments are placed in the appropriate place and the macro calls are replaced by macro definitions.

Specification of databases:-

Pass 1:-

- The input macro source program.
- The output macro source program to be used by Pass2.
- Macro-Definition Table (MDT), to store the body of macro def^{ns}.
- Macro-Definition Table Counter (MDTC), to mark next available entry MDT.
- Macro- Name Table (MNT), used to store names of macros.
- Macro Name Table counter (MNTC), used to indicate the next available entry in MNT.
- Argument List Array (ALA), used to substitute index markers for dummy arguments before storing a macro-def^{ns}.

Specification of databases:-

- **Pass 2:-**
- The copy of the input from Pass1.
- The output expanded source to be given to assembler.
- MDT, created by Pass1.
- MNT, created by Pass1.
- Macro-Definition Table Pointer (MDTP), used to indicate the next line of text to be used during macro-expansion.
- Argument List Array (ALA), used to substitute macro-call arguments for the index markers in the stored macro-def^{ns}

	MACROS		PROCEDURE
1	The corresponding machine code is written every time a macro is called in a program.	1	The Corresponding m/c code is written only once in memory
2	Program takes up more memory space.	2	Program takes up comparatively less memory space.
3	No transfer of program counter.	3	Transferring of program counter is required.
4	No overhead of using stack for transferring control.	4	Overhead of using stack for transferring control.
5	Execution is fast	5	Execution is comparatively slow.
6	Assembly time is more.	6	Assembly time is comparatively less.
7	More advantageous to the programs when repeated group of instruction is too short.	7	More advantageous to the programs when repeated group of instructions is quite large.

AD = assembler directive
IS = imperative statements
DL= DS & DC = declaration statements & declaration counters
RG = registers
CC= comparison condition