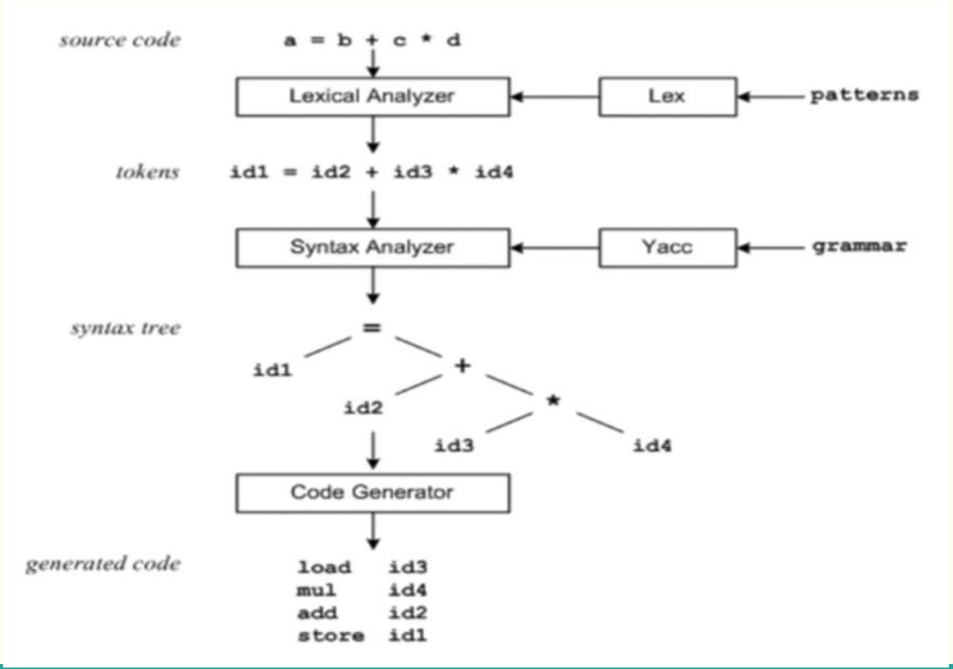
Project Step 2: Parser/Syntax analyzer

Programming Languages

Due: April 12, 2026, 23:59

Project Description

- Part 1 was on lexical analysis and you've written a lex file, etc.
- 2nd part will be on parsing and you will write a yacc file and the rest.



Project 2: Parser with **yacc**

- PART A: Design of your own language (update/enhancement)
- **PART B: Syntax Analyzer / Parser (yacc)**
- **PART C: Example Programs**

PART A: Design of the language

- In Part 1, you've already given a name and designed your language and you have your grammar in BNF form.
- Continue from the new starter code provided in github classroom. **Leave your lex repo as it is.** You can copy your lex file and any other file from the lext part into the new project. And then work on it.
- First, make any enhancements to your grammar by adding the features you want to add. See the next slide.
- Your readme should contain the new grammar.

Enhancements

- In this part, you will add the functionality to your language by implementing features below. Your grade will depend on the ones you successfully implemented.
- <u>**Features and their scores for grading:**</u>
  - Simple PL with statement-by-statement execution: **8 pts**
  - Comments: **2 pts**
  - Conditional statements (at least one of if-else, switch, etc.): **5 pts**
  - Loops (at least one of for, while, do-while, etc.): **5 pts**
  - Ability to write functions: **5 pts**
  - Exception handling: **5 pts**
- 30pts in total
- *Note: if your original language design doesn't have some of these features you need to either add it or a equivalent feature. For example, instead of loops you may rely on recursion, then you need to demonstrate that as well.*

PART B: Syntax Analyzer

- Here you will write a yacc file newlang.y which will produce an executable of the compiler of your language once compiled via gcc.
- Once you have the compiler, you should be able to run your code in your language
- For example if you have a code like below (your syntax can be different of course)
  - `a=false;`
  - `b=true;`
  - `if(a | b) then c = true;`
  - `print(c);`
- Your output will be something like
  - `true`

PART C: Examples

- You will write one or more example program(s) in your language.
- You are supposed to demonstrate all the constructs in your language with this program.
    - For example, if you can have loops, write a loop
    - If you can write functions, write one.
    - Etc.
- We should be able to understand your syntax and play with the code and still get correct results.

Submission

1. Submit your github repo link by the deadline.
2. Work on the yacc repo. (Everyone must have a GitHub account and commit separately so that we can see who wrote where, etc.) If your nickname is not your name, write your name to your GitHub profile so that we can understand who is who.
   At the end, your repo must contain all files including .l, .y, readme, makefile and your code files.
3. Template starter code:
   https://github.com/akdenizcse/pl-yacc-starter-kit

Contents of your submission

- Your new repo will contain:
  - A project report (an `README.md` file) that includes:
    - Project group members
    - Name of your programming language
    - (Updated?) Grammar in BNF form
    - Explaining ot the syntax of your language
    - Any design decisions you make
  - Your lex and yacc files (`plname.l, plname.y`)
  - And your example program(s) (`exampleprog1.yourextension, ...`)

References

- The starter code is based on this repo: https://github.com/jengelsma/yacc-tutorial
- The related video can be very helpful:

  https://www.youtube.com/watch?v=__-wUHG2rfM

- Another useful video:

  https://www.youtube.com/watch?v=8EO5Y7KhoeU