

Due Mar 30 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** pdf




You are to create an Abstract Syntax Tree using YACC for our Cminus language.

This lab can easily take 30 hours. START EARLY



<http://epaperpress.com/lexandyacc/>  [\(http://epaperpress.com/lexandyacc/\)](http://epaperpress.com/lexandyacc/) (has an example that creates an ast. Click on calculator on the left)

Background: from our previous lab we know we can check a program for syntax correctness. With an Abstract Syntax Tree (AST), we can store the structure of the input program in a data structure (called an Intermediate Representation - IR), which allows us to perform multiple passes on the source information to improve the performance of the code. A final pass over the AST allows us to generate target (assembly language). Our ultimate goal is to generate MIPS code and run that code on a MIPS simulator called SPIM.


Our task is to program YACC such that the result at the end of parsing (after yyparse()), is to have a pointer to a data structure which, when properly traversed, represent our parsed program. For us to do this, we must add additional information to our YACC program which constructs an AST during the shift/reduce processes used in YACC. Basically, at the end of each production rule, we will need to add Semantic action to: create a node, plumb the newly created node and to ensure that the information is attached to the yylval companion stack. The AST node will have different types to tell us what that node is representing.

The objective is after yyparse() completes, your main() program needs to print out the AST. The print out should be such that the output formatting indicates the structure of the program your read in. My input file [HERE](#)  generated the output file [HERE](#).  

Things that you will likely need to do

- 1) Create a separate AST.c file which will contain all your Abstract Syntax Tree code. You can use what I have created [HERE](#) 
- 2) Add semantic actions to every production rule in your Cminus.y submission from the previous LAB. You may use mine as as a hint or starting point. Please note that the supplied code may or may not be a completely correct CMINUS parser -- [HERE](#). 
- 3) You DO NOT have to check for variable declarations being done prior to the use of the NAME. The next LAB will be the inclusion of SYMBOL table.
- 4) Utilize an AST routine you write to print on the AST to help debug your sematic actions.

Deliverables

- 1) A copy of your YACC file, well documented.
- 2) A copy of your Abstract Syntax tree code well documented
- 3) The output of your parse for the following INPUT given [HERE](#). 

Note: We may ask you to demonstrate your code.