# EXP 4 STUDY OF YACC TOOL

YACC is known as Yet Another Compiler Compiler. It is used to produce the source code of the syntactic analyzer of the language produced by LALR (1) grammar. The input of YACC is the rule or grammar, and the output is a C program. If we have a file translate.y that consists of YACC specification, then the UNIX system command is:

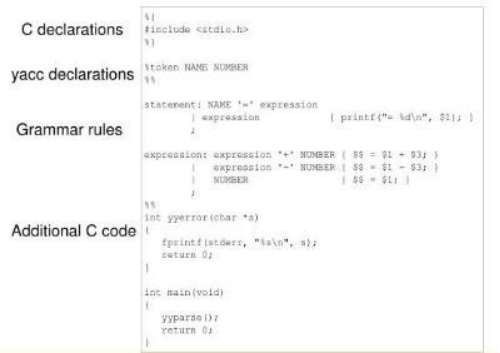
## YACC translate.y

This command converts the file translate.y into a C file y.tab.c. It represents an LALR parser prepared in C with some other user’s prepared C routines. By compiling y.tab.c along with the ly library, we will get the desired object program a.out that performs the operation defined by the original YACC program.

The construction of translation using YACC is illustrated in the figure below:

|  |
| --- |
|  |
|  |  |

## STRUCTURE OF YACC:



|  |
| --- |
|  |
|  |  |

**Declarations Part:** This part of YACC has two sections; both are optional. The first section has ordinary C declarations, which is delimited by %{ and %}. Any temporary variable used by the second and third sections will be kept in this part. Declaration of grammar tokens also comes in the declaration part. This part defined the tokens that can be used in the later parts of a YACC specification.

|  |
| --- |
|  |
|  |  |

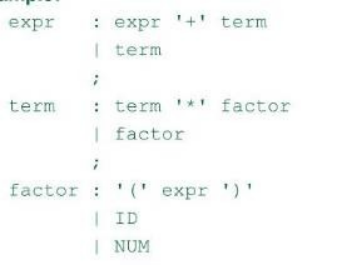
**Translation Rule Part:** After the first %% pair in the YACC specification part, we place the translation rules. Every rule has a grammar production and the associated semantic action.

A set of productions:

<head> => <body>1 | <body>2 | ….. | <body>n would be written in YACC as

|  |  |  |
| --- | --- | --- |
| <head> :  | | <body>1  <body>2  ….. | {<semantic action>1}  {<semantic action>2} |
| | | <body>n | {<semantic action>n} |
|  |  |  |
| ; |  |  |

Grammar rule section



In a YACC production, an unquoted string of letters and digits that are not considered tokens is treated as non-terminals.