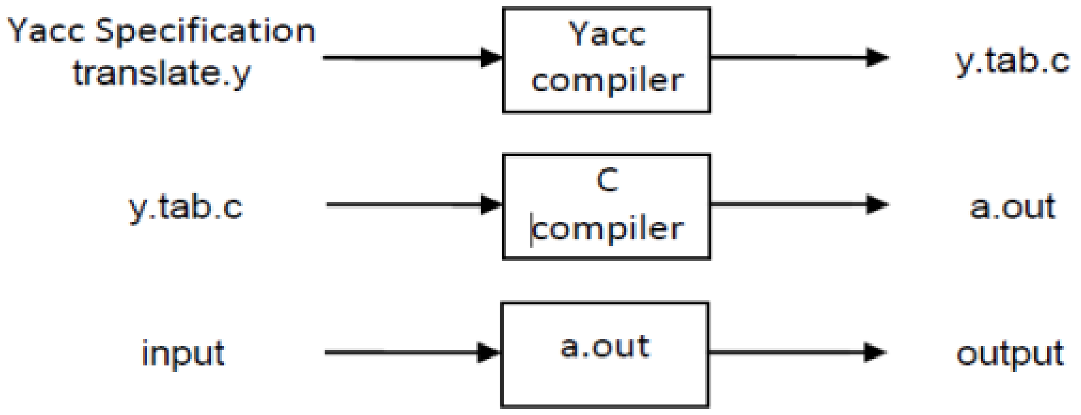
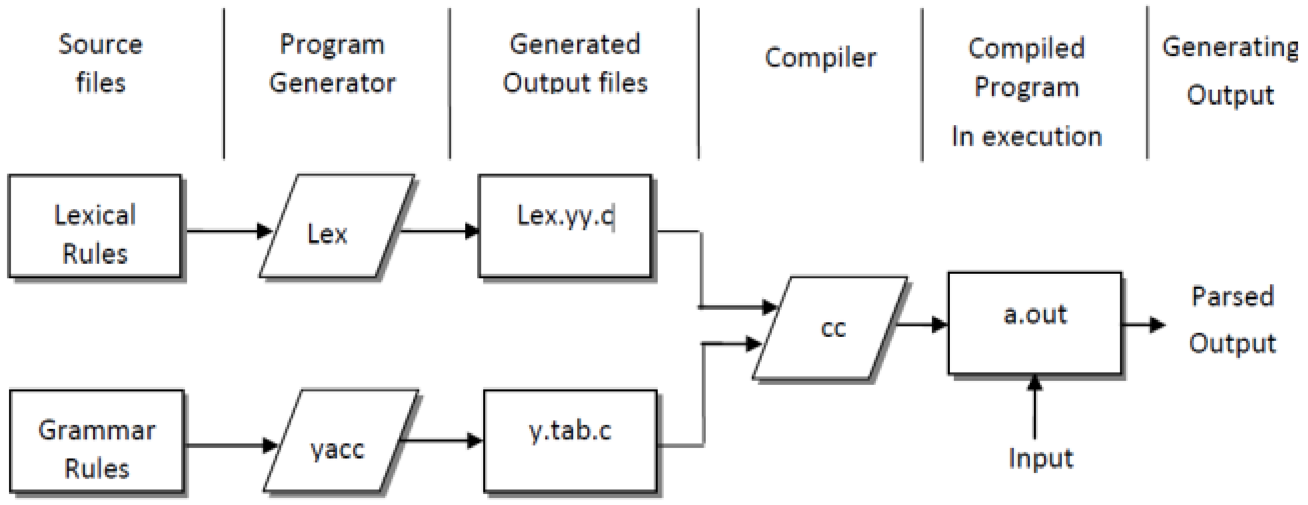
YACC

- YACC is a Yet Another Compiler Compiler.
- Yacc is a computer program for the Unix operating system.
- It is a LALR parser generator, generating a parser, the part of a compiler that tries to make syntactic sense of the source code, specifically a LALR parser, based on an analytic grammar written in a notation similar to BNF.
- Yacc itself used to be available as the default parser generator on most Unix systems.
- The input to Yacc is a grammar with snippets of C code (called "actions") attached to its rules.
- Its output is a shift-reduce parser in C that executes the C snippets associated with each rule as soon as the rule is recognized. Typical actions involve the construction of parse trees.

declarations
%%
translation rules
%%
supporting C routine



Design of a syntax analyzer for a sample language:



YACC (Yet Another Compiler Compiler).

- Automatically generate a parser for a context free grammar (LALR parser)
  Allows syntax direct translation by writing grammar productions and semantic action
  LALR(1) is more powerful than LL(1).
- Work with lex. YACC calls yylex to get the next token.
  YACC and lex must agree on the values for each token.
- Like lex, YACC pre-dated c++, need workaround for some constructs when using c++ (will give an example).
- yyparse() returns 0 if the program is grammatically correct, non-zero otherwise.
- YACC automatically builds a parser for the grammar (LALR parser).

Program to recognize a valid variable (identifier) which starts with a letter followed by any number of letters or digits.

LEX

```
%{
        #include"y.tab.h"
        extern   yylval;
%}
%%
[0-9]+    {yylval=atoi(yytext);  return  DIGIT;}
[a-zA-Z]+        {return LETTER;}
[\t]     ;
\n      return 0;
.       {return yytext[0];}
%%
```

YACC

```
%{
        #include<stdio.h>
%}
%token           LETTER DIGIT
%%
variable:        LETTER|LETTER rest
;
rest:  LETTER rest
        |DIGIT rest
        |LETTER
        |DIGIT
        ;
%%
main()
{
        yyparse();
        printf("The string is a valid variable\n");
}
int yyerror(char *s)
{
        printf("this is not a valid variable\n");
        exit(0);
}
```

OUTPUT

```
$lex p4b.l
$yacc –d p4b.y
$cc lex.yy.c y.tab.c –ll
$./a.out
input34
The string is a valid variable
$./a.out
89file
This is not a valid variable
```