Lex program

```
%%
while                          return WHILE;
[A-Za-z] ([A-Za-z] | [0-9])*       return ID;
[0-9]+                         { return NUM; }
[ \t ]                         ;
\n                             yyterminate();
.                              return yytext[0];
%%
```

Yacc program

```
%token ID NUM WHILE
%right '='
%left '+' '-'
%left '*' '/'
%left MINUS

%%

S : WHILE{ L1();} '(' E ')' {Lcond(); } E ';' { End();}

E : V '=' { push (); } E{codegen_assign();}
  | E '+' { push (); } E{codegen();}
  | E '-' { push (); } E{codegen();}
  | E '*' { push (); } E{codegen();}
  | E '/' { push (); } E{codegen();}
  |'(' E ')'
  | '-' { push (); } E{codegen_umin();} %prec MINUS
  | V
  |NUM{push();}
  ;
V : ID{push();}
```

```
  ;
%%

#include"lex.yy.c"
#include<stdio.h>
char st[100][10];
int top=0;
char temp[3]="t0";
main()
{
printf("enter expression\n");
yyparse();
}

push()
{
strcpy(st[++top],yytext);
}

codegen()
{
printf("%s = %s %s %s\n",temp,st[top-2],st[top-1],st[top]); //t0=a*b
top-=2;
strcpy(st[top],temp);
temp[1]++;
}

codegen_umin()
{
printf("%s = -%s\n",temp,st[top]);//t0=-a
top--;
strcpy(st[top],temp);
temp[1]++; //to generate new temporary variable
}
```

```
codegen_assign()
{
printf("%s = %s\n",st[top-2],st[top]); //c = t0
top-=2;
}

L1()
{
printf("\nL1: \n");
}
Lcond()
{
printf("%s = not %s\n", temp,st[top]);
printf("if %s goto End \n",temp);
temp[1]++; //to generate new temporary variable
}
End()
{
printf("goto L1\n");
printf("End : end of while loop\n\n");
}
```

```
Enter the expression : while(c=788) v=7*u/8+uuu;

L1:
c = 788
t0 = not c
if t0 goto End
t1 = 7 * u
t2 = t1 / 8
t3 = t2 + uuu
v = t3
goto L1
End: end of while loop
```