Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

SET I

Continuous Assessment Test - 4

Question Paper

Degree & Branch		BE (CSE)		**Semester**	VI
Subject Code & Name		UCS1602 Compiler Design		**Regulation: 2018**	
Section		C	**Academic Year**	**2020-2021**	
Reg. No:		185001146	**Name**	Shivanirudh S G	
Date:	30.04.2021	Batch: I	**Time:** **8.15 am – 10.15 am**	**Max. marks : 50**	

Code for Optimized three address code generation

LEX FILE:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "y.tab.h"
%}
%option yylineno

id [a-z][a-z]*
num [0-9]+
add_op ("+"|"-")
mul_op ("*"|"/")

%%
{num} {yylval.int_val = atoi(yytext);return NUM;}
{id} {yylval.str = strdup(yytext);return ID;}
"=" {yylval.str = strdup(yytext);return AOP;}
{mul_op} {yylval.str = strdup(yytext);return MUL_OP;}
{add_op} {yylval.str = strdup(yytext);return ADD_OP;}
";" {return SEP;}
[+\-^*/,()] {return *yytext;}
[ \t\n]+ {;}

. {
char errmsg[100];
sprintf(errmsg, "Invalid Character: %s at line %d", yytext, yylineno);
strcat(errmsg, "\n");
yyerror(errmsg);
}
```

```
%%
```

YACC:

```
%{
#include<stdio.h>
int flag = 0;
int yylex(void);
int yyerror(char *);
int yywrap();
extern int yy_flex_debug;

int tmp = 0;
struct info{
char *var;
char *code;
int int_val;
};

typedef struct info node;

node *makeNode(){
node *n = (node*)calloc(1, sizeof(node));
n->int_val = 0;
n->var = (char*)calloc(50, sizeof(char));
n->code = (char*)calloc(5000, sizeof(char));
return n;
}
%}

%token NUM ID AOP
%token SEP
%token MUL_OP ADD_OP

%right MUL_OP
%left ADD_OP

%union{
int int_val;
char *str;
struct info *Node;
}

%type<str> ID ADD_OP MUL_OP
%type<int_val> NUM
%type<Node> program code
%type<Node> assn_stmts assn_stmt expr
%type<Node> E T F

%%
program : code
code : assn_stmts assn_stmt {
```

```
printf("%s\n%s", $1->code, $2->code);
}
;
assn_stmts : assn_stmt assn_stmts{
printf("%s%s", $1->code, $2->code);
}
| assn_stmt {
$$ = $1;
}
;
assn_stmt : ID AOP expr {
$$ = makeNode();
char tac[100];
printf("%s", $1);
printf("%-5s := %s\n", $$->var, $3->var);
printf("%s%s", $3->code, tac);
}
;
expr : E{
$$ = $1;
}
;

E : T MUL_OP E{
$$ = makeNode();
char tac[100];
printf("x%d", ++tmp);
printf("%-5s := %s %s %s\n", $$->var, $1->var, $2, $3->var);
printf("%s%s%s", $1->code, $3->code, tac);
}
| T{
$$ = $1;
}
| F{
$$ = $1;
}
;

T : T ADD_OP F{
$$ = makeNode();
char tac[100];
printf("x%d", ++tmp);
printf("%-5s := %s %s %s\n", $$->var, $1->var, $2, $3->var);
printf("%s%s%s", $1->code, $3->code, tac);
}
| F{
$$ = $1;
}
;

F : ID{
```

```
$$ = makeNode();
printf("%s", $1);
printf("");
}
| NUM{
$$ = makeNode();
$$->int_val = $1;
printf("%d", $1);
printf("");
}
;
%%
int yyerror(char* str){
printf("\n%s", str);
flag = 1;
return 0;
}

int yywrap(){
return 1;
}

int main(){
yy_flex_debug = 1;
printf("\nGiven code\n");
system("cat file.txt");
printf("\n-------------------------------------------------------------------------------------\n");
printf("\nThree Address Code\n");

yyparse();
return 0;
}
```

Output:

```
➜ ./a.out < file.txt

Given code
a = b*c/d+g
x = y+z-s+g*z/h
-------------------------------------------------------------------------------------

Three Address Code
bcdgx1      :=  +
x2      :=  /
x3      :=  *
a      :=
yzx4      :=  +
sx5      :=  -
gx6      :=  +
zhx7      :=  /
x8      :=  *
x      :=

syntax error%
```