VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2023-24

School Of Computer Science and Engineering (SCOPE)

Digital Assignment - I

Compiler Design Lab

Apurva Mishra: 22BCE2791

Date: 12 Novemeber 2024

Contents

1 Assignment 2
1.1 Answer 2
1.2 Output 4

1 Assignment

Question 1 : Write a C program to implement Code optimization technique.

1.1 Answer

Code 1: main.l

```
%{
#include <stdlib.h>
#include <stdio.h>
void yyerror(char *);
#include "y.tab.h"
%}
%%
[0-9]+ {
yylval = atoi(yytext);
return INTEGER;
}
[a-z] {
yylval = *yytext;
return VARIABLE;
}
("jmp") { return JMP; }
[-+/*;=] { return *yytext; }
[ \t\n]+ ;
. yyerror("invalid character");
%%
int yywrap(void) {
return 1;
}
```

Code 1: main.y

```
%token INTEGER VARIABLE JMP
%left '+' '-'
%left '*' '/'
%{
#include "core.h"
void yyerror(char *);
int yylex(void);
int sym[26];
%}
%%
program:
function { ; }
;
function:
function expr { ;}
| /* NULL */
;
```

```
expr:
| VARIABLE '=' INTEGER '+' INTEGER ';' { new_node($1, $3, $5); print_line();}
| VARIABLE '=' VARIABLE '+' INTEGER ';' { new_node($1, $3, $5); print_line();
| VARIABLE '=' INTEGER '+' VARIABLE ';' { new_node($1, $3, $5); print_line();
|  VARIABLE  '='  VARIABLE  '+'  VARIABLE  ';'  {  new_node($1,  $3,  $5);
print_line();}
| jump ';' { ; }
;
jump:
JMP INTEGER { ;}
;
%%
void yyerror(char *s) {
fprintf(stderr,"%s\n", s);
}
int main(void) {
yyparse();
return 0;
}
```

Code 1: core.h

```
#include <stdio.h>

struct eval {
    int var;
    int var1;
    int var2;
};

struct eval global[10];
int i = 0;

int skip_count = 0;

void new_node(int var, int var1, int var2) {
    global[i].var = var;
    global[i].var1 = var1;
    global[i].var2 = var2;

    i += 1;
}

void print_line() {
    if (skip_count > 0) {
        skip_count -= 1;
        return;
    }

    int k = i - 1;

    int prev = -1;
```

```
31      for (int j = 0; j < k; j++) {
32              if (global[j].var1 == global[k].var1 && global[j].var2 ==
global[k].var2) {
33              prev = global[j].var;
34              break;
35          }
36      }
37
38      if (prev != -1) {
39         printf("%c = %c; | Common Subexpressions Elimination\n", global[k].var,
prev);
40          return;
41      }
42
43      if (global[k].var1 >= 'a' && global[k].var2 >= 'a') {
44              printf("%c = %c + %c; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
45      } else if (global[k].var1 >= 'a' && global[k].var2 < 'a') {
46              printf("%c = %c + %d; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
47      } else if (global[k].var1 < 'a' && global[k].var2 >= 'a') {
48              printf("%c = %d + %c; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
49      } else {
50          printf("%c = %d; | Constant Folding\n", global[k].var, global[k].var1
+ global[k].var2);
51      }
52  }
```

Code 1: run.sh

```
1  #!/bin/bash
2
3  lex main.l
4  yacc -d main.y
5  gcc lex.yy.c y.tab.c -o main
6  ./main
```

1.2 Output

The given code applies following two optimisations:

- Common Subexpressions Elimination
- Constant Folding

```
da/ass6/ques2 via C v16.0.0-clang
❯ ./run.sh
conflicts: 2 shift/reduce, 1 reduce/reduce
main.y:18.5: warning: rule never reduced because of conflicts: expr: /* empty */
a = b + c;
d = b + c;
e = a + 2;
f = a + 2;
g = 2 + 5;
h = 2 + 5;
a = b + c; | No optimization
d = a; | Common Subexpressions Elimination
e = a + 2; | No optimization
f = e; | Common Subexpressions Elimination
g = 7; | Constant Folding
h = g; | Common Subexpressions Elimination
```