19CSE401 Compiler Design Lab

Anindita Das Badhan
CH.EN.U4CSE22180
4th Year CSE-B

Lab Exercise- 08

Aim: To write a program that implements the target code generation

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int label[20]; // For storing label numbers
int no = 0;    // Label counter

// Function declaration
int check_label(int k);

int main() {
    FILE *fp1, *fp2;
    char fname[100];
    char op[20], operand1[20], operand2[20], result[20];
    char ch;
    int i = 0;

    printf("\nEnter filename of the intermediate code: ");
    scanf("%s", fname);

    fp1 = fopen(fname, "r");
    fp2 = fopen("target.txt", "w");

    if (fp1 == NULL || fp2 == NULL) {
        printf("\nError opening file.\n");
        exit(1);
    }
```

```c
    // Read the intermediate code line by line
    while (fscanf(fp1, "%s", op) != EOF) {
        i++;

        if (check_label(i)) {
            fprintf(fp2, "\nLABEL#%d:\n", i);
        }

        if (strcmp(op, "print") == 0) {
            fscanf(fp1, "%s", result);
            fprintf(fp2, "\tOUT %s\n", result);
        }
        else if (strcmp(op, "goto") == 0) {
            fscanf(fp1, "%s %s", operand1, operand2);
            fprintf(fp2, "\tJMP %s, LABEL#%s\n", operand1, operand2);
            label[no++] = atoi(operand2);
        }
        else if (strcmp(op, "[]=") == 0) {
            fscanf(fp1, "%s %s %s", operand1, operand2, result);
            fprintf(fp2, "\tSTORE %s[%s], %s\n", operand1, operand2, result);
        }
        else if (strcmp(op, "uminus") == 0) {
            fscanf(fp1, "%s %s", operand1, result);
            fprintf(fp2, "\tLOAD -%s, R1\n", operand1);
            fprintf(fp2, "\tSTORE R1, %s\n", result);
        }
```

```
    else {
        // Handle arithmetic and relational operations using first character
        switch (op[0]) {
            case '*':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tLOAD %s, R1\n", operand2);
                fprintf(fp2, "\tMUL R1, R0\n");
                fprintf(fp2, "\tSTORE R0, %s\n", result);
                break;
            case '+':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tLOAD %s, R1\n", operand2);
                fprintf(fp2, "\tADD R1, R0\n");
                fprintf(fp2, "\tSTORE R0, %s\n", result);
                break;
            case '-':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tLOAD %s, R1\n", operand2);
                fprintf(fp2, "\tSUB R1, R0\n");
                fprintf(fp2, "\tSTORE R0, %s\n", result);
                break;
            case '/':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tLOAD %s, R1\n", operand2);
                fprintf(fp2, "\tDIV R1, R0\n");
                fprintf(fp2, "\tSTORE R0, %s\n", result);
                break;
            case '%':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tLOAD %s, R1\n", operand2);
                fprintf(fp2, "\tMOD R1, R0\n");
                fprintf(fp2, "\tSTORE R0, %s\n", result);
                break;
```

```
                break;
            case '=':
                fscanf(fp1, "%s %s", operand1, result);
                fprintf(fp2, "\tSTORE %s, %s\n", operand1, result);
                break;
            case '>':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tJGT R0, %s, LABEL#%s\n", operand2, result);
                label[no++] = atoi(result);
                break;
            case '<':
                fscanf(fp1, "%s %s %s", operand1, operand2, result);
                fprintf(fp2, "\tLOAD %s, R0\n", operand1);
                fprintf(fp2, "\tJLT R0, %s, LABEL#%s\n", operand2, result);
                label[no++] = atoi(result);
                break;
            default:
                // Unsupported operation
                fprintf(fp2, "\t; Unknown operation: %s\n", op);
        }
    }
}

fclose(fp1);
fclose(fp2);
```

```
    // Display the generated target code
    fp2 = fopen("target.txt", "r");
    if (fp2 == NULL) {
        printf("Error opening output file.\n");
        exit(1);
    }

    printf("\nGenerated Target Code:\n\n");
    while ((ch = fgetc(fp2)) != EOF) {
        putchar(ch);
    }

    fclose(fp2);
    return 0;
}

// Function to check if the line number matches a label
int check_label(int k) {
    for (int i = 0; i < no; i++) {
        if (k == label[i]) return 1;
    }
    return 0;
}
```

Input.txt

```
1 = a t1
2 = b t2
3 + t1 t2 t3
4 = t3 c
5 print c
6
```

Output:

```
asecomputerlab@linux:~/CDLAB180$ nano tarcodegen.c
asecomputerlab@linux:~/CDLAB180$ gcc tarcodegen.c -o tarcodegen
asecomputerlab@linux:~/CDLAB180$ ./tarcodegen

Enter filename of the intermediate code: input.txt

Generated Target Code:

        STORE a, t1
        STORE b, t2
        LOAD t1, R0
        LOAD t2, R1
        ADD R1, R0
        STORE R0, t3
        STORE t3, c
        OUT c
```

Result: Thus, the program to implement target code generation has been successfully executed