Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering
Session: 2024-2025

Compiler Design Lab

PRACTICAL No. 5

Name : Shreyas Sahare

Batch : A4

Roll No : 60

Topic: Three Address Code Generation

Aim: Write a program to generate three address code for the given language construct using SDTS.

- **(a)** Batch 1: if-then-else,
- **(b)** Batch 2: for loop
- **(c)** Batch 3: while loop
- **(d)** Batch 4: do while loop

Input: Example for if-then-else

```
if (a<5 && b>c)
{
        c= b+d
        d= i+j
}
else
{
         d= a+ b
        k= x+y

}
```

Output:

1) if (a<5) goto 3
2) Goto__
3) If b> c goto 5
4) goto 10
5) T1=b+d
6) c=T1
7) T2=i+j
8) d=T2
9) goto 12
10) T3=a+b
11) d=T3
12) T4=x+y
13) k=T4
14) END

Code:

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

// Structure to hold a TAC instruction
typedef struct {
    char code[100];
} TAC;

// Function to generate a new temporary variable
char* new_temp(int* temp_count) {
    (*temp_count)++;
    char* temp = (char*)malloc(10 * sizeof(char));
    sprintf(temp, "T%d", *temp_count);
    return temp;
}

// Function to generate TAC for a statement
void generate_statement_tac(char* statement, TAC* tac, int* tac_count, int* temp_count) {
    if (strchr(statement, '=')) {
        char lhs[50], rhs[50];
        sscanf(statement, "%[^=]=%s", lhs, rhs);
        char* temp = new_temp(temp_count);
        sprintf(tac[*tac_count].code, "%s = %s", temp, rhs);
        (*tac_count)++;
```

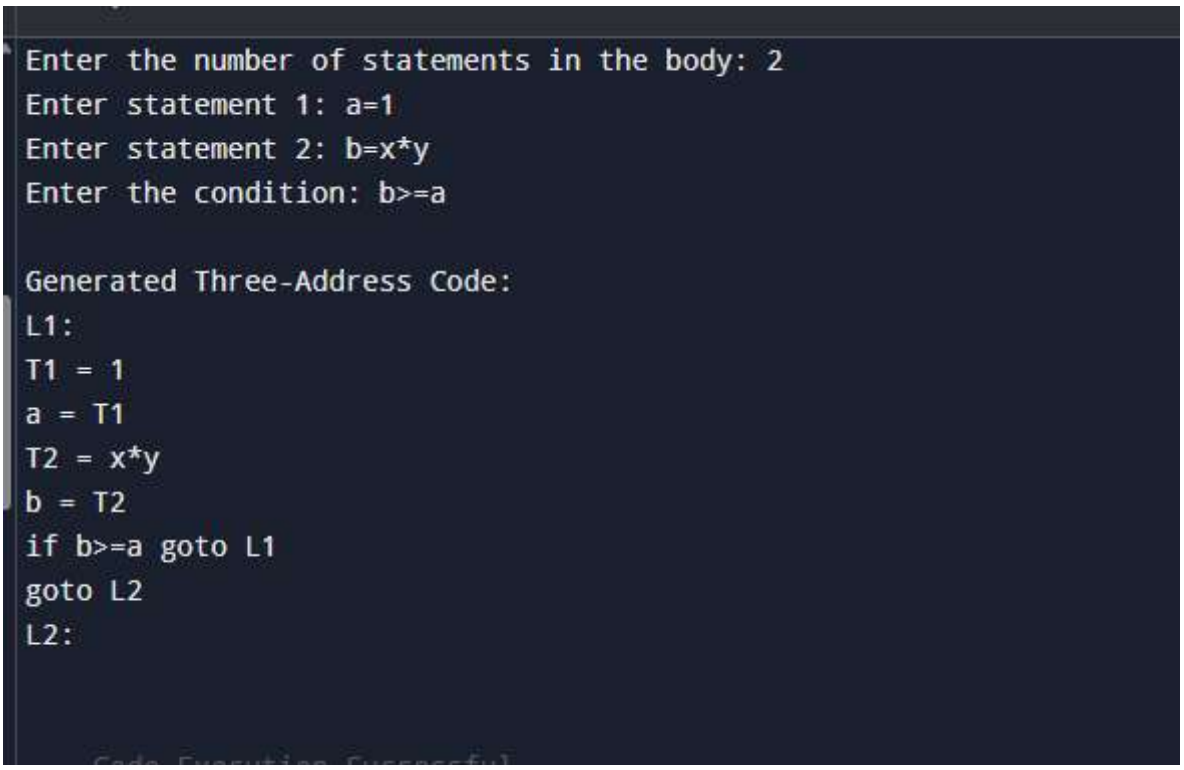
```
        sprintf(tac[*tac_count].code, "%s = %s", lhs, temp);
        (*tac_count)++;
        free(temp);
    } else {
        fprintf(stderr, "Unsupported statement type\n");
        exit(1);
    }
}

// Function to parse and generate TAC for a condition
void generate_condition_tac(char* condition, TAC* tac, int* tac_count, char* true_label, char* false_label) {
    char* cond1 = strtok(condition, "&|");
    char* cond2 = strtok(NULL, "&|");
    char* operator = strchr(condition, '&') ? "&&" : "||";

    char* cond_label1 = "L3";
    char* cond_label2 = "L4";

    if (cond2 == NULL) {
        // Single condition
        sprintf(tac[*tac_count].code, "if %s goto %s", cond1, true_label);
        (*tac_count)++;
        sprintf(tac[*tac_count].code, "goto %s", false_label);
        (*tac_count)++;
    } else {
        // Compound condition
        if (strcmp(operator, "&&") == 0) {
            // AND condition
            sprintf(tac[*tac_count].code, "if %s goto %s", cond1, cond_label1);
```

```
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "goto %s", false_label);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "%s:", cond_label1);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "if %s goto %s", cond2, true_label);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "goto %s", false_label);
            (*tac_count)++;
        } else if (strcmp(operator, "||") == 0) {
            // OR condition
            sprintf(tac[*tac_count].code, "if %s goto %s", cond1, true_label);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "goto %s", cond_label1);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "%s:", cond_label1);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "if %s goto %s", cond2, true_label);
            (*tac_count)++;
            sprintf(tac[*tac_count].code, "goto %s", false_label);
            (*tac_count)++;
        }
    }
}

// Function to generate TAC for the do-while loop
void generate_tac(char** body, int body_count, char* condition, TAC* tac, int* tac_count, int* temp_count) {
    char start_label[10] = "L1";
    char end_label[10] = "L2";
```

```
    // Generate TAC for the body
    sprintf(tac[*tac_count].code, "%s:", start_label);
    (*tac_count)++;
    for (int i = 0; i < body_count; i++) {
        generate_statement_tac(body[i], tac, tac_count, temp_count);
    }

    // Generate TAC for the condition
    generate_condition_tac(condition, tac, tac_count, start_label, end_label);
    sprintf(tac[*tac_count].code, "%s:", end_label);
    (*tac_count)++;
}

int main() {
    int body_count;
    printf("Enter the number of statements in the body: ");
    scanf("%d", &body_count);
    getchar(); // Consume newline

    char** body = (char**)malloc(body_count * sizeof(char*));
    for (int i = 0; i < body_count; i++) {
        body[i] = (char*)malloc(100 * sizeof(char));
        printf("Enter statement %d: ", i + 1);
        fgets(body[i], 100, stdin);
        body[i][strcspn(body[i], "\n")] = 0; // Remove newline
    }

    char condition[100];
    printf("Enter the condition: ");
```

```
    fgets(condition, 100, stdin);
    condition[strcspn(condition, "\n")] = 0; // Remove newline

    TAC tac[100];
    int tac_count = 0;
    int temp_count = 0;

    generate_tac(body, body_count, condition, tac, &tac_count, &temp_count);

    // Print the generated TAC
    printf("\nGenerated Three-Address Code:\n");
    for (int i = 0; i < tac_count; i++) {
        printf("%s\n", tac[i].code);
    }

    // Free allocated memory
    for (int i = 0; i < body_count; i++) {
        free(body[i]);
    }
    free(body);

    return 0;
}
```

Screen Shot :


Enter the number of statements in the body: 2
Enter statement 1: a=1
Enter statement 2: b=x*y
Enter the condition: b>=a
Generated Three-Address Code:
L1:
T1 = 1
a = T1
T2 = x*y
b = T2
if b>=a goto L1
goto L2
L2: