21BDS0340

Abhinav Dinesh Srivatsa

Compiler Design Lab

Assignment – V

Question 1

Aim: To write a C program that constructs a DAG from postfix notation.

Program:

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

typedef struct Node
{
    char data;
    struct Node *left;
    struct Node *right;
} Node;

Node *createNode(char data)
{
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

Node *buildDAG(char postfix[])
{
    Node *stack[100];
    int top = -1;

    for (int i = 0; postfix[i] != '\0'; i++)
    {
        char ch = postfix[i];

        if (isalnum(ch))
        {
            Node *newNode = createNode(ch);
            stack[++top] = newNode;
        }
        else
        {
            Node *newNode = createNode(ch);
```

```
            newNode->right = stack[top--];
            newNode->left = stack[top--];

            stack[++top] = newNode;
        }
    }

    return stack[top];
}

void printPostfix(Node *root)
{
    if (root == NULL)
    {
        return;
    }

    printPostfix(root->left);
    printPostfix(root->right);
    printf("%c ", root->data);
}

int main()
{
    char postfix[100];
    printf("Enter a postfix expression: ");
    scanf("%s", postfix);

    Node *root = buildDAG(postfix);

    printf("Postfix expression: ");
    printPostfix(root);
}
```

Output:

```
Enter a postfix expression: 45+23+*
Postfix expression: 4 5 + 2 3 + *
```

Question 2

Aim: To implement the backend of a compiler from postfix

Program:

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

#define MAX_EXPR_LENGTH 100
```

```c
typedef struct Quadruple
{
    char op;
    char arg1[10];
    char arg2[10];
    char result[10];
} Quadruple;

int isOperator(char ch)
{
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

void postfixToQuadruple(char postfix[], Quadruple quadruples[], int *numQuadruples)
{
    int top = -1;
    int tempCount = 1;

    for (int i = 0; postfix[i] != '\0'; i++)
    {
        char ch = postfix[i];

        if (isalnum(ch))
        {
            Quadruple q;
            q.op = '=';
            q.arg1[0] = ch;
            q.arg1[1] = '\0';
            q.arg2[0] = '\0';
            sprintf(q.result, "T%d", tempCount++);
            quadruples[(*numQuadruples)++] = q;
            top++;
        }
        else if (isOperator(ch))
        {
            Quadruple q;
            q.op = ch;
            sprintf(q.arg2, "%c", quadruples[top].result[1]);
            sprintf(q.arg1, "%c", quadruples[top - 1].result[1]);
            sprintf(q.result, "T%d", tempCount++);
            quadruples[(*numQuadruples)++] = q;
            top--;
        }
        else
        {
            Quadruple q;
            q.op = '=';
            sprintf(q.arg1, "%c", quadruples[top].result[1]);
            q.arg2[0] = '\0';
            sprintf(q.result, "T%d", tempCount++);
            quadruples[(*numQuadruples)++] = q;
```

```
            top--;
        }
    }
}

void printQuadruples(Quadruple quadruples[], int numQuadruples)
{
    printf("Quadruple Notation:\n");
    for (int i = 0; i < numQuadruples; i++)
    {
        printf("%c, %s, %s, %s\n", quadruples[i].op, quadruples[i].arg1,
quadruples[i].arg2, quadruples[i].result);
    }
}

void generateMachineCode(Quadruple quadruples[], int numQuadruples)
{
    printf("\n8086 Machine Code:\n");

    for (int i = 0; i < numQuadruples; i++)
    {
        Quadruple q = quadruples[i];

        if (q.op == '=')
        {
            printf("MOV %s, %s\n", q.result, q.arg1);
        }
        else
        {
            printf("MOV AX, %s\n", q.arg1);

            switch (q.op)
            {
            case '+':
                printf("ADD AX, %s\n", q.arg2);
                break;
            case '-':
                printf("SUB AX, %s\n", q.arg2);
                break;
            case '*':
                printf("MUL %s\n", q.arg2);
                break;
            case '/':
                printf("DIV %s\n", q.arg2);
                break;
            }

            printf("MOV %s, AX\n", q.result);
        }
    }
}
```

```c
int main()
{
    char postfix[MAX_EXPR_LENGTH];
    printf("Enter a postfix expression: ");
    fgets(postfix, sizeof(postfix), stdin);
    postfix[strcspn(postfix, "\n")] = '\0';

    Quadruple quadruples[MAX_EXPR_LENGTH];
    int numQuadruples = 0;

    postfixToQuadruple(postfix, quadruples, &numQuadruples);
    printQuadruples(quadruples, numQuadruples);
    generateMachineCode(quadruples, numQuadruples);

    return 0;
}
```

Output:

```
Enter a postfix expression: 45+34+*1-
Quadruple Notation:
=, 4, , T1
=, 5, , T2
+, 1, 2, T3
=, 3, , T4
=, 4, , T5
+, 2, 3, T6
*, 1, 2, T7
=, 1, , T8
-, 1, 2, T9

8086 Machine Code:
MOV T1, 4
MOV T2, 5
MOV AX, 1
ADD AX, 2
MOV T3, AX
MOV T4, 3
MOV T5, 4
MOV AX, 2
ADD AX, 3
MOV T6, AX
MOV AX, 1
MUL 2
MOV T7, AX
MOV T8, 1
MOV AX, 1
SUB AX, 2
MOV T9, AX
```