18. Write a C program to implement the back end of the compiler

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

char op[10];

char arg1[10];

char arg2[10];

char result[10];

} Quadruple;

Quadruple quad[100];

int quadIndex = 0;

void addQuadruple(char\* op, char\* arg1, char\* arg2, char\* result) {

strcpy(quad[quadIndex].op, op);

strcpy(quad[quadIndex].arg1, arg1);

strcpy(quad[quadIndex].arg2, arg2);

strcpy(quad[quadIndex].result, result);

quadIndex++;

}

void generateAssembly() {

printf("Generated Assembly Code:\n");

for (int i = 0; i < quadIndex; i++) {

if (strcmp(quad[i].op, "+") == 0) {

printf("MOV R0, %s\n", quad[i].arg1);

printf("ADD R0, %s\n", quad[i].arg2);

printf("MOV %s, R0\n", quad[i].result);

} else if (strcmp(quad[i].op, "\*") == 0) {

printf("MOV R0, %s\n", quad[i].arg1);

printf("MUL R0, %s\n", quad[i].arg2);

printf("MOV %s, R0\n", quad[i].result);

} else if (strcmp(quad[i].op, "=") == 0) {

printf("MOV %s, %s\n", quad[i].result, quad[i].arg1);

}

}

}

int main() {

addQuadruple("=", "b", "", "t1");

addQuadruple("=", "c", "", "t2");

addQuadruple("\*", "t2", "d", "t3");

addQuadruple("+", "t1", "t3", "t4");

addQuadruple("=", "t4", "", "a");

generateAssembly();

return 0;

}

