Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н. Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Космический факультет» КАФЕДРА «Прикладная математика, информатика и вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 9

по курсу «Системное программное обеспечение»

на тему: «Создание однопроходного макропроцессора для языка Ассемблера»

Студент \_\_К3-56Б\_\_  **\_\_\_\_\_\_\_\_\_\_** Несмеянов С. А.

(Группа) (Подпись, дата) (И.О.Фамилия)

#### Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Чернышов А. В.

(Подпись, дата) (И.О.Фамилия)

2024 г.

**Задание**

Написать на языке программирования Си программу простого макропроцессора для любого языка Ассемблера. Программа макропроцессора должна поддерживать следующие элементы:

* обнаруживать и запоминать макроопределения (текстовые строки между директивами macro и endm);
* запоминать имена макроопределений;
* распознавать формальные позиционные макропараметры;
* по именам макроопределений обнаруживать вызовы макрогенерации

в тексте программы на Ассемблере:

* распознавать фактические позиционные макропараметры;
* выполнять макроподстановку с учётом распознанных формальных и фактических макропараметров.

**Список используемых структур, команд и функций**

**Структура данных**

* **struct macro**: Основная структура, описывающая макрос, содержит:
  + num: количество параметров макроса.
  + name: имя макроса.
  + param: массив параметров макроса.
  + code: строка, хранящая текст макроса.

**Функции**

1. **replace**
   * Заменяет параметры макроса на переданные при его вызове:
     + strstr: Находит вхождение параметра, который нужно заменить.
     + strncpy: Копирует часть строки до найденного параметра.
     + strcat: Конкатенирует замененные параметры.
     + strcpy: Записывает результат в исходную строку.
2. **processFile**
   * Читает входной файл и создает выходной файл с подставленными макросами:
     + fopen: Открывает входной и выходной файлы.
     + fgets: Читает строки из файла.
     + strstr: Проверяет, содержит ли строка макрос или вызов макроса.
     + malloc: Выделяет память для нового макроса.
     + strncpy: Копирует имя макроса.
     + strtok: Разбивает параметры макроса.
     + strcat: Конкатенирует строки для создания тела макроса.
     + fprintf: Записывает конечный код в выходной файл.
     + fclose: Закрывает файлы.
3. **main**
   * Точка входа программы:
     + argv: Получает аргументы командной строки для названий входного и выходного файлов.
     + processFile: Вызывает функцию для обработки файлов с макросами

**Код программы**

**macro\_processor.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define size 32

struct macro;

typedef struct macro\* pMacro;

struct macro {

int num; // количество параметров макроса

char name[size]; // имя макроса

char param[size][size]; // параметры макроса

char code[size\*size]; // тело макроса

};

void replace(char\* str, char\* substr1, char\* substr2) {

char res[size\*size];

memset(res, '\0', size\*size);

char\* temp = strstr(str, substr1);

while (temp != NULL) {

char words[size\*size];

memset(words, '\0', size\*size);

strncpy(words, str, temp - str); // захватываем всё до макроса

strcat(res, words);

strcat(res, substr2);

strcat(res, temp + strlen(substr1)); // после макроса

strcpy(str, res);

temp = strstr(str, substr1);

}

}

void processFile(char\* file1, char\* file2) {

FILE\* in = fopen(file1, "r");

FILE\* out = fopen(file2, "w");

pMacro macros[size];

int num\_macro = 0; // количество макросов

char temp[size\*size];

while (fgets(temp, size\*size, in) != NULL) {

// Если это определение макроса

char\* str = strstr(temp, "macro");

if (str != NULL) {

macros[num\_macro] = (pMacro)malloc(sizeof(struct macro));

macros[num\_macro]->num = 0;

strncpy(macros[num\_macro]->name, temp, strlen(temp) - strlen(str) - 1);

char\* params = str + strlen("macro");

if (strcmp("\n", params) != 0) { // если есть параметры

char\* param = strtok(params, " ,\n");

for (int i = 0; param != NULL; i++) {

strcpy(macros[num\_macro]->param[i], param);

macros[num\_macro]->num++;

param = strtok(NULL, " ,\n");

}

}

memset(macros[num\_macro]->code, '\0', size\*size);

while (fgets(temp, size\*size, in) != NULL) {

if (strstr(temp, "endm") == NULL) {

strcat(macros[num\_macro]->code, temp);

} else {

num\_macro++;

break;

}

}

} else {

// Обработка вызовов макросов

int pr = 1;

for (int i = 0; i < num\_macro; i++) {

char\* name = strstr(temp, macros[i]->name);

if (name != NULL && strstr(temp, "macro") == NULL) {

char\* params = name + strlen(macros[i]->name);

char param\_actual[size][size];

if (strcmp("\n", params) != 0) { // если есть параметры

char\* param = strtok(params, " ,\n");

for (int j = 0; param != NULL; j++) {

strcpy(param\_actual[j], param);

param = strtok(NULL, " ,\n");

}

}

char new\_code[size\*size];

strcpy(new\_code, macros[i]->code);

for (int j = 0; j < macros[i]->num; j++) {

replace(new\_code, macros[i]->param[j], param\_actual[j]);

}

fprintf(out, "%s", new\_code);

pr = 0;

break;

}

}

if (pr == 1) {

fprintf(out, "%s", temp);

}

}

}

fclose(in);

fclose(out);

}

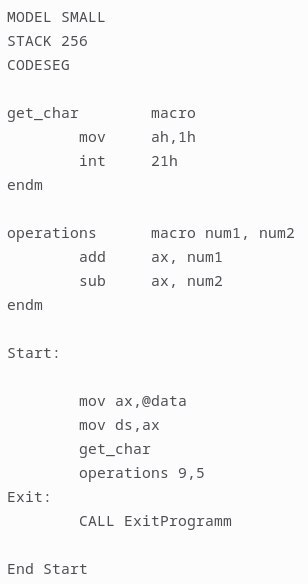
int main(int argc, char\* argv[]) {

processFile(argv[1], argv[2]);

return 0;

}

**input.asm**



**Результат работы программы**

**output.asm**

