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 **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

О т ч е т

по лабораторной работе № 5

Название: Программирование с использованием разноязыковых модулей

Дисциплина: Машинно-зависимые языки и основы компиляции

Студент гр. ИУ6-41Б 10.04.2023 Т. Е. Старжевский
(Подпись, дата) (И.О. Фамилия)

Преподаватель 15.04.2023 С. С. Данилюк
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Введение

Цель работы: изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.

Ход работы

Задание: Дана текст не более 255 символов. Удалить последовательности одинаковых символов, завершающихся символом «#».

1) Схема алгоритма показана на рисунке 1:

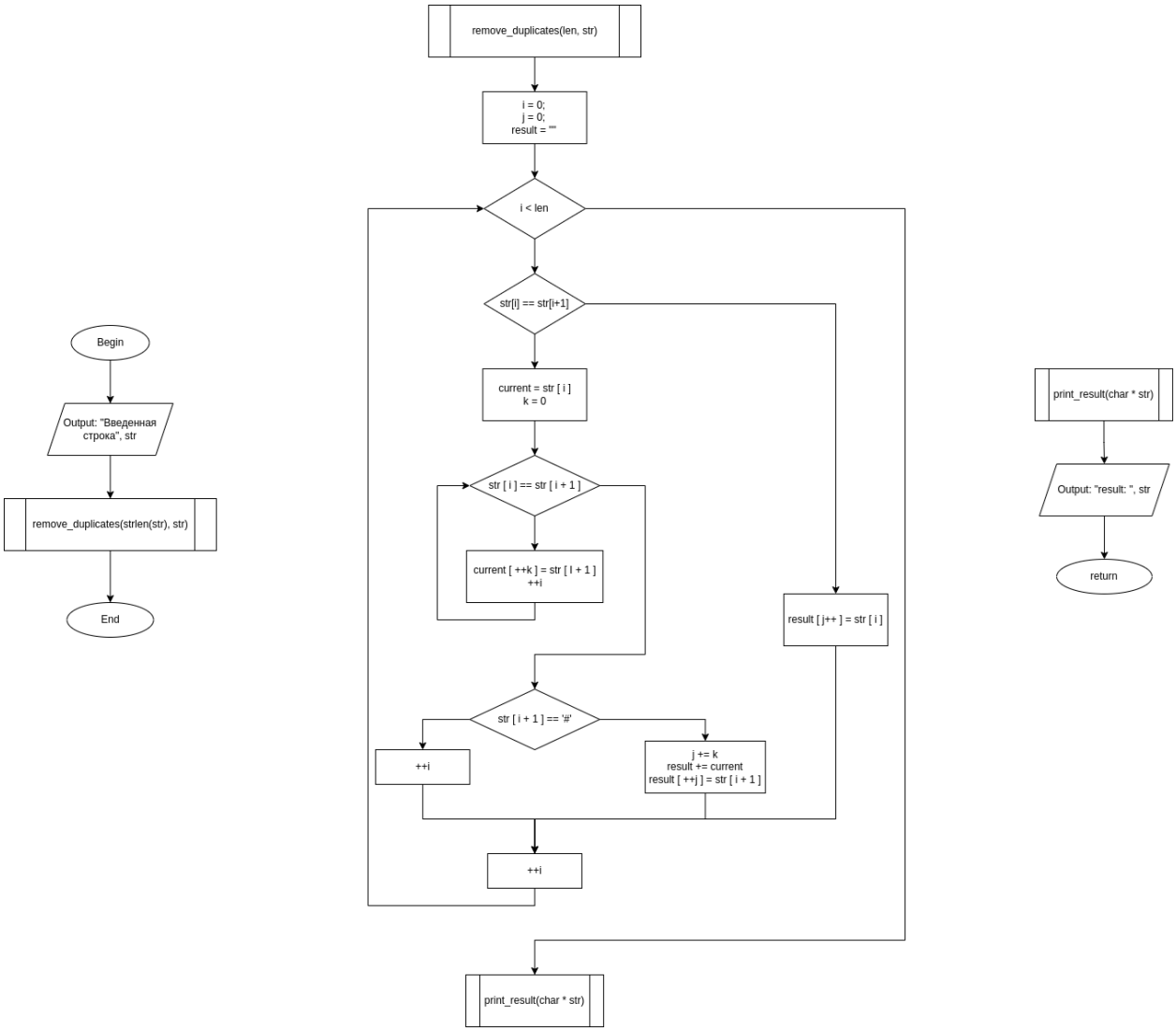

Рисунок 1 – схема алгоритма

2) Программа состоит из 3 модулей.

2.1) Код программы 1 модуля, содержащим старт программы и выход из нее **src/main.c**:

```
#include <iostream>
#include <stdio.h>
#include <cstring>

char* remove_duplicates_(char* str) {
size_t len = strlen(str);
char* result = (char*) malloc(len + 1);
size_t i = 0;
size_t j = 0;

while(i < len) {
if (str[i] != str[i+1]) {
result[j++] = str[i];
} else {
char current[256] = {str[i]};
size_t k = 0;

while(str[i] == str[i+1]){
current[++k] = str[i+1];
++i;
}

if (str[i+1] != '#'){
j += k;
strcat(result, current);
result[++j] = str[i+1];
} else {
++i;
}
}
++i;
}

return result;
}

extern "C" char * remove_duplicates(size_t size, char * str);

int main() {
char str[] = "abcd# abcdef aaa# bbbb# bbbb c# aaa bbbb abcdaaa# # kakkkakkkk#ak";
// char should_be[] = "abcd# abcdef bbbb c# aaa bbbb abcd kakkkaak";
printf("Введенная строка: %s\n", str);
```

```
remove_duplicates(strlen(str), str);

// printf("%s\n", result);
// if (strcmp(result, should_be)) {
// std::cout << "False\n";
// } else {
// std::cout << "True\n";
// }

// free(result);

return 0;
}
```

2.2) Код программы 2 модуля, содержащий логику программы на ассемблере **src/removeDuplicates.asm**:

```
%include "../lib64.asm"


%define STDIN 0
%define READ 0
%define STDOUT 1
%define WRITE 1
%define EXIT 60

section .bss
result resb 256

section .text
extern print_result
global remove_duplicates

remove_duplicates:
push rcx;
push rdx;
push rbx;

; rax = char* str удалим
; rsi = char* str
; rdi = strlen(str) удалим
mov rbx, rdi; rbx = strlen(str)
xor rcx, rcx; Обнуляем rcx, в качестве i
xor rax, rax; Обнуляем rax, в качестве j
; rdx временный
lea rdi, result; Результат помеащаем в rdi

while_i_lower_len:
```

```
cmp rcx, rbx
jge end_while; если i >= len то заканчиваем цикл
push rcx

mov dl, byte[rsi + 1]
cmp [rsi], dl
mov rcx, 0
je check_posled
mov rcx, 1
movsb
jmp next

check_posled:
mov dl, byte[rsi + rcx + 1]
cmp [rsi + rcx], dl
jne check_hash
inc rcx
jmp check_posled
check_hash:
cmp byte[rsi + rcx + 1], '#'
je skip_posled; если последовательность завершилась #, то скипаем ее
movsb; иначе копируем всю предыдущую последовательность
jmp next

skip_posled:
add rsi, rcx
add rsi, 2
jmp next

next:
pop rcx
inc rcx
jmp while_i_lower_len


end_while:
pop rcx;
pop rdx;
pop rbx;
; lea rax, result
lea rdi, result
call print_result
ret
```

2.3) Код программы 3 модуля, содержащий вывод результата программы на ассемблере, написанный на C **src/print_result.c**:

```
#include <iostream>
```

```
#include <stdio.h>
#include <cstring>

extern "C" void print_result(char * str) {
char should_be[] = "abcd# abcdef bbbb c# aaa bbbb abcd kakkkaak";
printf("Удаленные последовательности: \n");
printf("%s\n", str);
if (strcmp(str, should_be)) {
printf("False\n");
} else {
printf("True\n");
}
}
```

2.4) Makefile для сборки трех модулей:

```
CXX := g++
CXX_FLAGS := -Wall -Wextra -std=c++20 -ggdb

BIN := bin
SRC := src
INCLUDE := include
LIB := lib

LIBRARIES :=
EXECUTABLE := main
ASM_FILE := removeDuplicates
MODULES := print_result

all: $(BIN)/$(EXECUTABLE)

run: all
./$(BIN)/$(EXECUTABLE)

# $(BIN)/$(EXECUTABLE): $(SRC)/*.c $(BIN)/$(ASM_FILE).o
# $(CXX) $(CXX_FLAGS) -I$(INCLUDE) -L$(LIB) $^ -o $@ $(LIBRARIES) -fno-pie -no-pie

$(BIN)/$(EXECUTABLE): $(BIN)/$(EXECUTABLE).o $(BIN)/$(ASM_FILE).o $(BIN)/$(MODULES).o
$(CXX) $(CXX_FLAGS) -o $(BIN)/$(EXECUTABLE) $(BIN)/$(ASM_FILE).o
$(BIN)/$(EXECUTABLE).o $(BIN)/$(MODULES).o -fno-pie -no-pie


$(BIN)/$(EXECUTABLE).o: $(SRC)/$(EXECUTABLE).c
$(CXX) $(CXX_FLAGS) -c $(SRC)/$(EXECUTABLE).c -o $(BIN)/$(EXECUTABLE).o


$(BIN)/$(ASM_FILE).o: $(SRC)/$(ASM_FILE).asm
```

```
nasm -f elf64 -l $(BIN)/$(ASM_FILE).lst $(SRC)/$(ASM_FILE).asm -o $(BIN)/$(ASM_FILE).o


$(BIN)/$(MODULES).o: $(SRC)/$(MODULES).c
$(CXX) $(CXX_FLAGS) -c $(SRC)/$(MODULES).c -o $(BIN)/$(MODULES).o

clean:
-rm $(BIN)/*
```

Собрать и выполнить программу можно командой make run.

Результат работы программы представлен на рисунке 2:

```
timofey@timofey-ASUS:~/Projects/Study/BMSTU_Assembly/lab5$ make run
g++ -Wall -Wextra -std=c++20 -ggdb -c src/print_result.c -o bin/print_result.o
g++ -Wall -Wextra -std=c++20 -ggdb -o bin/main bin/removeDuplicates.o bin/main.o bin/print_result.o -fno-pie -no-pie
./bin/main
Введенная строка: abcd# abcdef aaa# bbbb# bbbb c# aaa bbbb abcdaaa#   # kakkkakkkk#ak
Удаленные последовательности:
abcd# abcdef    bbbb c# aaa bbbb abcd kakkkaak
True
```

Рисунок 2 - результат работы программы

3) Тестирование программы показаны на таблице 1:

Таблица 1 – Таблица тестирования

Исходные данные	Ожидаемый результат	Полученный результат
"abcd# abcdef aaa# bbbb# bbbb c#"	*abcd# abcdef bbbb c#*	*abcd# abcdef bbbb c#*
"abcd# abcdef aaa# bbbb# bbbb c# aaa bbbb abcdaaa#"	*abcd# abcdef bbbb c# aaa bbbb abcd*	*abcd# abcdef bbbb c# aaa bbbb abcd*
"abcd# abcdef aaa# bbbb# bbbb c# aaa bbbb abcdaaa# # kakkkakkkk#ak"	*abcd# abcdef bbbb c# aaa bbbb abcd kakkkaak*	*abcd# abcdef bbbb c# aaa bbbb abcd kakkkaak*
""	""	""
"a"	"a"	"a"

Вывод: Изучил конвенцию о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере и наоборот.

Контрольные вопросы:

1 .Что такое «конвенция о связи»? В чем заключается конвенция «register»?

Конвенция о связи – набор правил, определяющих, как между собой взаимодействует главный модуль и вызываемый. Конвенция о связи говорит, как передать параметры в процедуру, в каком порядке, какой модуль отвечает за выделение памяти.

Так, конвенция register предполагает, что параметры передаются через регистры EAX, ECX, EDX, остальные параметры передаются через стек в порядке, обратном их объявлению в функции. Очисткой стека занимается вызываемая процедура.

2. Что такое «пролог» и «эпилог»? Где располагается область локальных данных?

Пролог и эпилог функции – служебные части описания функции, гарантирующие корректную работу функции и стека. В прологе функции копируется текущее значение регистра ESP в регистр EBP, что гарантирует корректный доступ к параметрам и локальным при любых изменениях на стеке. Таким образом, параметры функции располагаются в области EBP+4*N байт, где N – номер параметра, а локальные переменные в области EBP-4*N, где N – номер параметра.

3. Как связана структура данных стека в момент передачи управления и текст программы и подпрограмм?

В большинстве конвенций о связях контроль над выделением памяти (push) на стек и очисткой памяти (add esp, <number>) занимается вызывающая программа.

4. С какой целью применяют разноязыковые модули в одном проекте?

Применение ассемблерных модулей в проекте C++ может ускорить выполнение программы за счет оптимизации необходимого участка кода.