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)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 5

Дисциплина: МЗЯиОК

Название: Связь разноязыковых модулей

Студент	ИУ6-43Б		В.К. Залыгин
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель работы

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

Задание

Дан текст не более 255 символов. Слова отделяются друг от друга пробелами. Поменять местами пары слов с указанными номерами.

Схема алгоритма

Схема алгоритма представлена на рисунках 1 и 2.

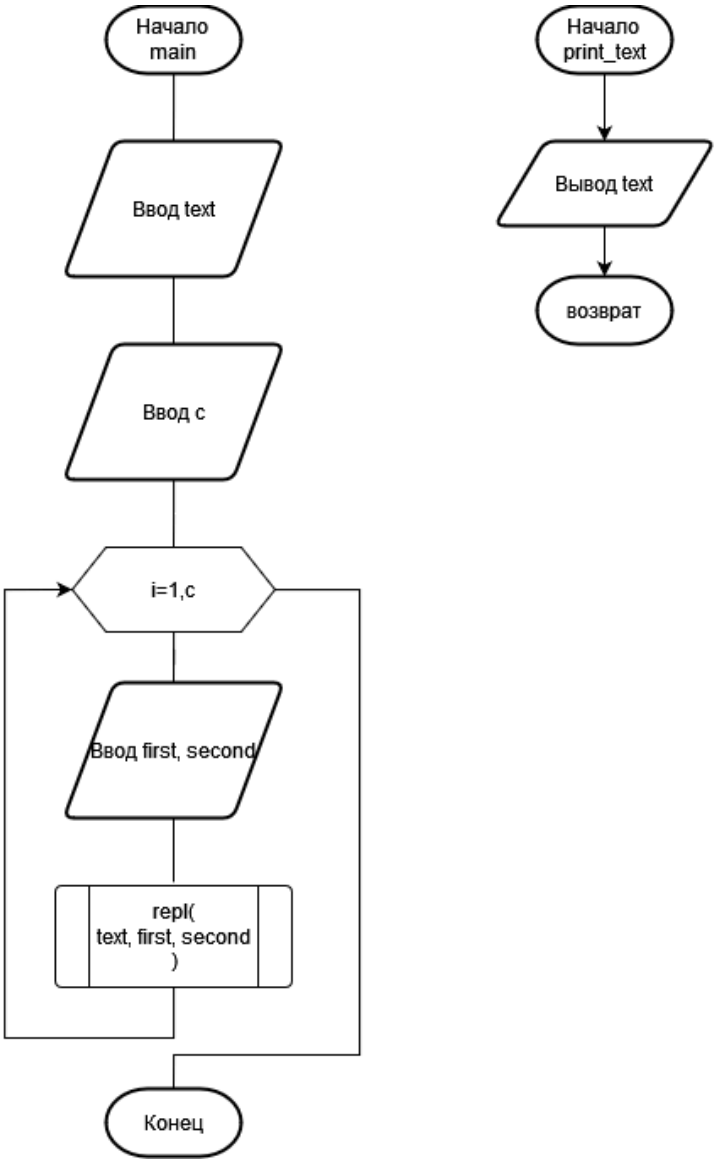

Рисунок 1 – Схема алгоритмов подпрограмм main и print_text

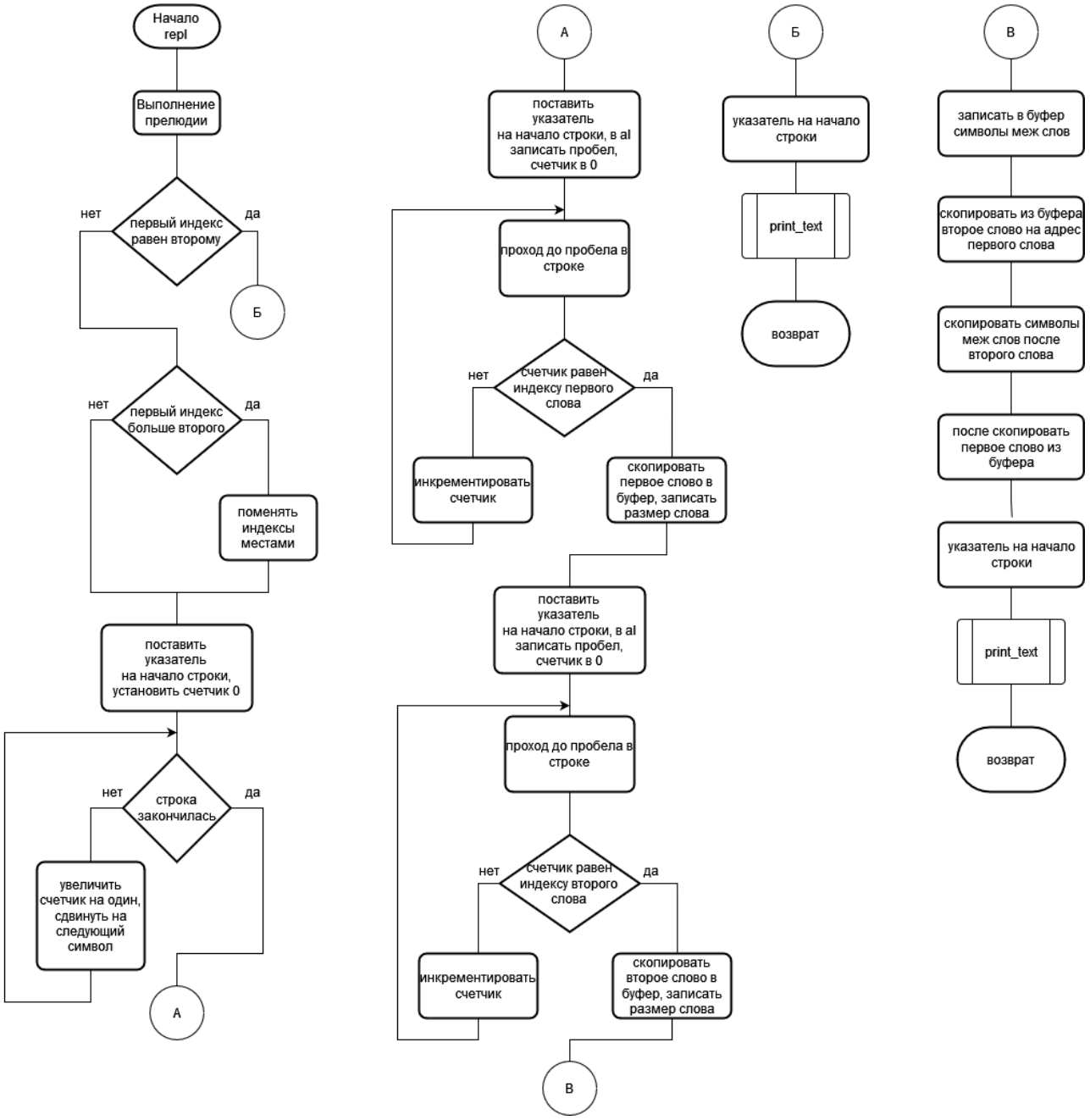

Рисунок 2 – Схема алгоритма подпрограммы repl

Код программы

На рисунке 2 представлен код главной программы. В листинге 1 представлен код ассемблера. На рисунке 3 представлен код вспомогательной подпрограммы.

```c
#include "stdio.h"

#define MAX 256

extern void repl(char * text, unsigned int idx1, unsigned int idx2);

int main() {
    char text[MAX];
    unsigned int c, first, second;
    printf("enter the sentence (less then 256 chars):\n");
    fgets(text, MAX, stdin);
    printf("enter amount of pairs:\n");
    scanf("%d", &c);
    for (int i = 0; i < c; ++i) {
        printf("enter the indexes of the first and the second word to swap them:\n");
        scanf("%d%d", &first, &second);
        repl(text, first, second);
    }
}
```

Рисунок 2 – Главная программа

```c
#include "stdio.h"

extern void print_text(char * text) {
    printf("%s", text);
}
```

Рисунок 3 – Вспомогательная подпрограмма

Листинг 1 – Код ассемблера

```
    section .data
too_big_idx         db    "err: incorrect index of the word",10,0
    section .bss
buffer              resb    255
    section .text
global    repl
extern    print_text

repl:    ; prologue
    push    rbp
    mov    rbp,    rsp
```

```
    push    rsi
    push    rax
    push    rbx
    push    rdx
    push    r8
    push    r9
    push    r10
    push    r11
    push    r12
    push    r13
    push    r14
    push    r15
    ; body
    cld
    push    rdi
    mov    r8,    rsi    ; idx1
    mov    r9,    rdx    ; idx2
    cmp    r8,    r9
    je    .skip            ; if idxes are same, no need to replace
    jl    .less
    xchg    r8,    r9
.less:    mov    r10,    -1    ; current idx
    ; find null (end of string)
    mov    al,    0
    dec    rdi
.l1:    inc    rdi
    inc    r10
    cmp    [rdi],    al
```

```
   jne   .l1
   mov   r11,   r10   ; length of string
   inc   r11
   ; search first word
   xor   r10,   r10
   mov   rdi,   [rsp]   ; get start of string
   mov   rcx,   r11
   mov   al,   " "
.l2:   cmp   r10,   r8
   je   .e2
repne   scasb
   inc   r10
   inc   rcx
   loop   .l2
   cmp   r10,   r8
   je   .e2
   jmp   .etbi
.e2:   mov   r12,   rdi   ; pointer to first word
repne   scasb
   mov   r13,   rdi
   sub   r13,   r12
   dec   r13          ; length of the first word
   ; search second word
   xor   r10,   r10
   mov   rdi,   [rsp]   ; get start of string
   mov   rcx,   r11
   mov   al,   " "
.l3:   cmp   r10,   r9
```

```
   je    .e3
repne    scasb
   inc    rcx
   inc    r10
   loop    .l3
   jmp    .etbi
.e3:    mov    r14,    rdi    ; pointer to second word
repne    scasb
   mov    r15,    rdi
   sub    r15,    r14
   dec    r15            ; length of the second word
   ; copy parts to the buffer
   mov    rsi,    r12
   mov    rdi,    buffer
   mov    rcx,    r13
rep    movsb                ; copy first word to the buffer
   mov    rsi,    r14
   mov    rcx,    r15
rep    movsb                ; copy second word to the buffer
   lea    rsi,    [r12+r13]
   mov    rcx,    r14
   sub    rcx,    r13
   sub    rcx,    r12
rep    movsb                ; copy text between words
   ; replace words
   mov    rdi,    r12
   lea    rsi,    [buffer+r13]
   mov    rcx,    r15
```

```
rep    movsb
    mov    rcx,    r14
    sub    rcx,    r13
    sub    rcx,    r12
rep    movsb
    lea    rsi,    [buffer]
    mov    rcx,    r13
rep    movsb
    ; print the same str
.skip:    pop    rdi
    call    print_text
    ; epilogue
.ret:    pop    r15
    pop    r14
    pop    r13
    pop    r12
    pop    r11
    pop    r10
    pop    r9
    pop    r8
    pop    rdx
    pop    rbx
    pop    rax
    pop    rsi
    pop    rbp
    ret
.etbi:    pop    rax
    mov    rdi,    too_big_idx
```

```
call    print_text
jmp     .ret
```

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Тестирование

№	Поток ввода	Ожидаемый результат	Поток вывода	Вердикт
1	hello my name is john doy 2 1 2 3 4	hello name my john is doy	hello name my john is doy	Верно
2	hello my name is john doy 1 0 5	doy hello my name is john	doy hello my name is john	Верно
3	hello my name is john doy 2 0 5 0 5	hello my name is john doy	hello my name is john doy	Верно

Выводы

Изучены конвенции о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм,

написанных на ассемблере. Написана программа, реализующая вызов ассемблерных инструкций из языка высокого уровня и обратно.

Контрольные вопросы

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

Конвенция о связи – договоренность о том, как передаются параметры в функции и возвращается результат. Конвенция register – параметры передаются через регистры.

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

Пролог – начало функции, там происходит перемещение указателя стека, чтобы локальные переменные не перезаписывали переменные вызывающей функции и сохранение состояния регистров.

Эпилог – восстанавливает стек и регистры в состояние, которое было до вызова функции.

Локальные данные располагаются на стеке.

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

Стек используется для хранения локальных данных и адресов возврата при вызове подпрограмм. При передаче управления стек сохраняет адрес возврата и локальные данные текущей функции, а затем загружает адрес и данные следующей функции.

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

Для оптимизации работы некоторых функций. Например, можно написать основную программу на языке более высокого уровня, а функции, которые должны выполняться очень быстро на языке более низкого уровня.