

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
Московский институт электронной техники»

**ЛАБОРАТОРНАЯ РАБОТА № 4
ПО АРХИТЕКТУРАМ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
МОДУЛИ И ФУНКЦИИ НА АСSEMBЛЕРЕ**

ВАРИАНТ 6

Егоров Вадим, Кудреватых Павел, Саркисов Эрик
ПМ-31

Январь 2022 г.

Содержание

Задание Л4.№1..	1
Задание Л4.№2..	2
Задание Л4.№3..	3
Задание Л4.№4..	4

При выполнении лабораторной работы использовалась ОС Kali Linux x86-64, соглашение System V AMD64 ABI.

Изъятие из учебного пособия "Программирование. Введение в профессию" автора А.В. Столярова:

"В среде ОС Unix традиционно более популярен именно синтаксис AT&T, но в применении к поставленной учебной задаче это создаёт некоторые проблемы. Учебные пособия, ориентированные на программирование на языке ассемблера в синтаксисе Intel, всё-таки существуют, тогда как синтаксис AT&T описывается исключительно в специальной (справочной) технической литературе, не имеющей целью обучение. Кроме того, необходимо учитывать и многолетнее господство среды MS DOS в качестве платформы для аналогичных учебных курсов; всё это позволяет назвать синтаксис Intel существенно более привычным для преподавателей (да и для некоторых студентов, как ни странно, тоже) и лучше поддерживаемым."

Ввиду солидарности с автором приведенных строк мы решились, начиная с данной лабораторной работы, писать ассемблерный код согласно синтаксису Intel.

Цель работы: изучить процесс компиляции программы на C++; научиться включать в проекты на языке C++ ассемблерные модули; изучить стандартные соглашения о вызовах и их соответствие платформам; научиться описывать функции и вызывать из программы на языке C++.

Задание Л4.№1.

Разработайте ассемблерную функцию, вычисляющую целое выражение от двух целых аргументов (в соответствии с вариантом), а также головную программу на языке C/C++, использующую разработанную функцию.

Вариант 6: $z(x, y) = 1234 + x - 5 \cdot y$ Остаток от деления на 2^n считать по правилам математики (неотрицательным даже для отрицательных чисел). Для проверки реализуйте вычисление того же выражения на C/C++.

```
#include <stdio.h>

extern int asm1(int, int);

int c1(int x, int y)
{
    return 1234 + x - 5*y;
}

int main()
{
    //task 1
    int x = 1, y = 2;
    printf("Result c1: %d\n", c1(x, y));
    printf("Result asm1: %d", asm1(x, y));

    return 0;
}
```

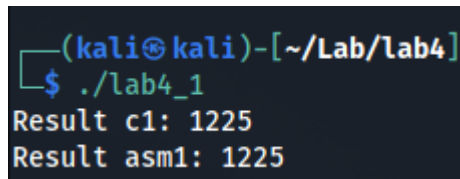
```

global asm1

section .text
asm1:
    push rbx
    mov eax, esi
    mov ebx, 5
    mul ebx
    mov ebx, eax
    mov eax, edi
    add eax, 1234
    sub eax, ebx
    pop rbx
    ret

```

Вывод:



```

(kali@kali)-[~/Lab/lab4]
$ ./lab4_1
Result c1: 1225
Result asm1: 1225

```

Задание Л4.№2.

Разработайте программу, целиком написанную на ассемблере, вычисляющую (вызывая ранее разработанную функцию) значение $z(x, y)$ для $x = 1, y = 0$ и выводящую полученное значение на стандартный вывод с использованием библиотеки `libc` (в частности, функции `printf`).

В этой ЛР и в дальнейшем исходные неизменяемые данные мы будем располагать в секции `.text`. Эти данные во время работы окажутся в сегменте кода, поэтому если будет произведен многочисленный запуск программы, то сегмент кода у них будет один на всех, что приведет к экономии памяти.

```

global main
extern printf, asm1

section .text
str: db "Result: %d", 10, 0
x dd 1
y dd 0

main:
    sub rsp, 8

    mov edi, [x]
    mov esi, [y]
    call asm1

    mov edi, str
    mov esi, eax
    call printf

    add rsp, 8
    xor eax, eax
    ret

```

Здесь осуществляется вызов функции `asm1` из предыдущего задания.

```
(kali㉿kali)-[~/Lab/lab4]
$ ./lab4_2
Result: 1235
```

Задание Л4.№3.

Опишите на произвольном языке высокого уровня (включая C/C++) функцию с пятью целочисленными параметрами, которая выводит свои параметры на экран и возвращает результат, равный пятому параметру. Вызовите её из ассемблерной функции (в том числе из написанной на ассемблере *main()*).

```
#include <stdio.h>

extern int asm3(int, int, int, int, int);

int asm3(int x1, int x2, int x3, int x4, int x5)
{
    printf("1) %d\n2) %d\n3) %d\n4) %d\n5) %d\n", x1, x2, x3, x4, x5);
    return x5;
}
```

```
global main
extern printf, asm3

section .text
str: db "Result: x5 = %d", 10, 0
x1 dd 1
x2 dd 2
x3 dd 3
x4 dd 4
x5 dd 5
```

```
main:        sub rsp, 8

    mov rdi, [x1]
    mov rsi, [x2]
    mov rdx, [x3]
    mov rcx, [x4]
    mov r8, [x5]
    call asm3

    mov rdi, str
    mov rsi, rax
    call printf

    add rsp, 8
    xor rax, rax
    ret
```

```
(kali㉿kali)-[~/Lab/lab4]
$ ./lab4_3
1) 1
2) 2
3) 3
4) 4
5) 5
Result: x5 = 5
```

Задание Л4.№4.

Бонус +2 балла для пар, обязательное для троек. Опишите на ассемблере одну подпрограмму с комплексным параметром $z = (z.re, z.im)$ и комплексным результатом $w = (w.re, w.im)$ (вещественные и мнимые части считать целочисленными) и вызовите её из другой ассемблерной программы.

Используемое нестандартное соглашение должно быть реентерабельным (в том числе при многопоточном выполнении),

Вариант 6: $w = z^2 + z$

Для каждого из заданий указывайте ОС, разрядность программы и соответствующее им соглашение.

В задании Л4.№4, где необходимо разработать и использовать нестандартное соглашение — либо подробно опишите его, либо, если оно незначительно отличается от одного из стандартных — укажите базовое стандартное соглашение и отличия разработанного вами от него.

```
global main
extern printf

section .text
str: db "Result: %d + i*%d", 10, 0
re dd 1
im dd 1

asm4:      push rbx

    mov rax, rsi
    mul rsi
    mov rcx, rax
    mov rax, rdi
    mul rdi
    sub rax, rcx
    add rax, rdi
    mov rcx, rax      ;res re

    mov rax, rdi
    mov rbx, 2
    mul rbx
    mul rsi
    add rax, rsi      ;res im

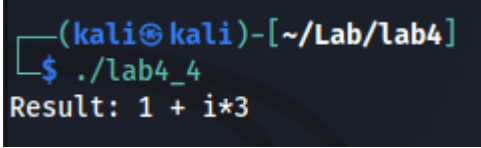
    mov rdi, str      ;str
    mov rdx, rax      ;im
    mov rsi, rcx      ;re
    call printf

    pop rbx
    ret

main:      sub rsp, 8
```

```
mov rdi, [re]
mov rsi, [im]
call asm4

add rsp, 8
xor rax, rax
ret
```

A terminal window with a dark background. The prompt is a green square followed by "(kali@kali)-[~/Lab/lab4]". The user has entered "\$./lab4_4" and the output is "Result: 1 + i*3".

```
(kali@kali)-[~/Lab/lab4]
$ ./lab4_4
Result: 1 + i*3
```