РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Мелкумян Арвин

Группа: НКАбд-04-23

МОСКВА

2023 г.

ПОНЯТИЕ ПОДПРОГРАММЫ. ОТЛАДЧИК GDB

Цель работы: Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

Ход работы.

Создадим программу которая вычисляет значение функции 2x+7 с помощью подпрограммы, где значение x задается с клавиатуры. Исходный код программы показан на рисунке 1.

```
%include 'in_out.asm'

SECTION .data
    msg: DB 'Введите x: ',0h
    result: DB '2x+7=',0h

SECTION .bss
    x: RESB 80
    res: RESB 80

SECTION .text
    GLOBAL _start

_start:
    mov eax, msg
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi

    call _calcul

    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit

_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret
```

Рисунок 1 — Исходный код программы

Результат работы программы показан на рисунке 2.

```
infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09$ ./lab09-1
Введите x: 5
2x+7=17
infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab09$
```

Рисунок 2 — Результат работы программы

Создадим программу, которая выводит приветствие Hello world! (рисунок 3), скомплиируем ее с флагом -g и откроем в отладичике gdb (рисунок 4).

```
1   SECTION .data
 1      msg1: db "Hello, ",0x0
 2      msg1Len: equ $ - msg1
 3
 4      msg2: db "world!",0xa
 5      msg2Len: equ $ - msg2
 6
 7  SECTION .text
 8      global _start
 9
10  _start:
11      mov eax, 4
12      mov ebx, 1
13      mov ecx, msg1
14      mov edx, msg1Len
15      int 0x80
16
17      mov eax, 4
18      mov ebx, 1
19      mov ecx, msg2
20      mov edx, msg2Len
21      int 0x80
22
23      mov eax, 1
24      mov ebx, 0
25      int 0x80
26
```

Рисунок 3 — Исходный код программы

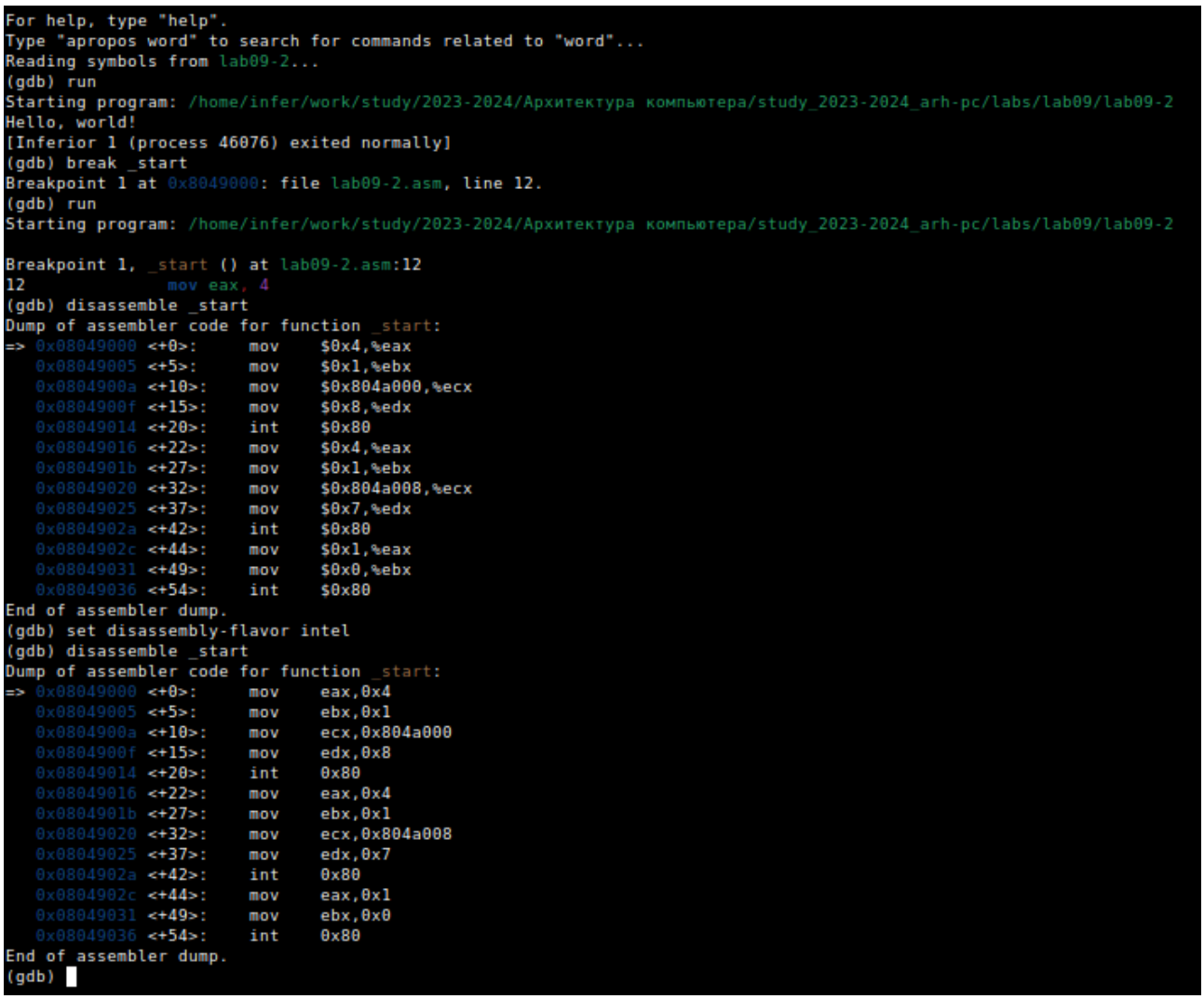


Рисунок 4 — Выполнение программы в отладчике

Как видно из рисунка, в режиме дизассемблирования в режиме intel появляются различия с режимом AT&T. Во-первых в AT&T константы обозначаются символом $, а имена регистров симовлом %, во вторых порядок присваивания слева направо.

На рисунке 5 показано пошаговое выполнение программы в отладчике с отображением кода и значений регистров.

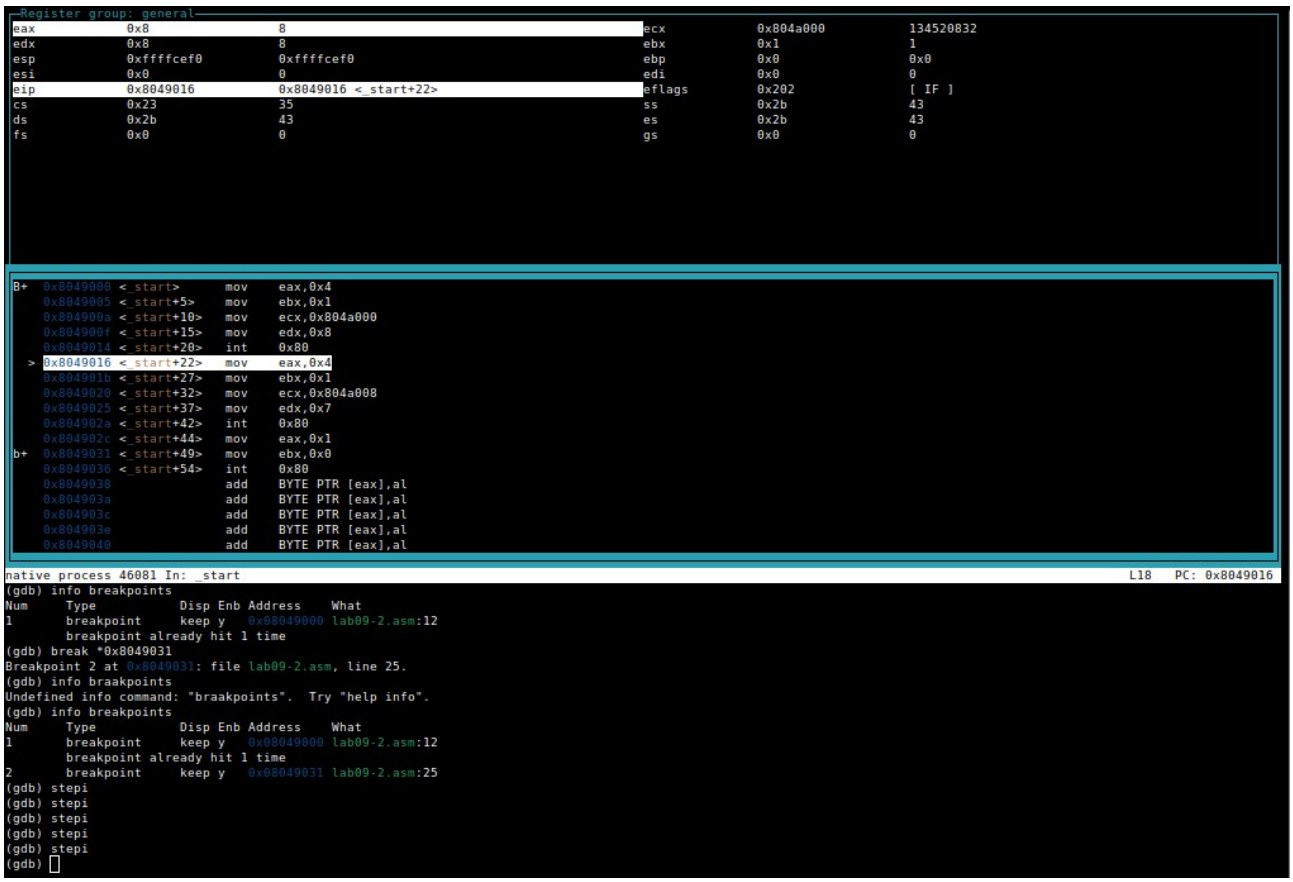
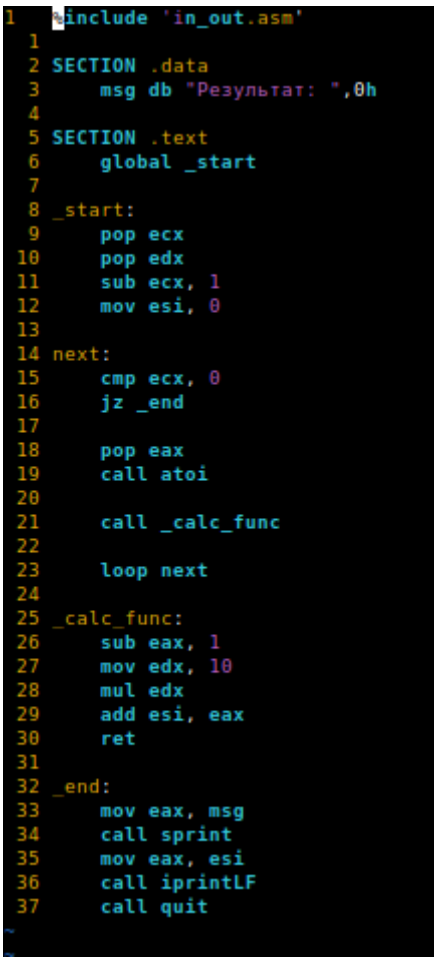


Рисунок 5 — Пошаговое выполнение программы

Переделаем программу вычисления функции из лабораторной работы №8 для работы с подпрограммами. Исходный код полученной программы показан на рисунке 6.



```
%include 'in_out.asm'

SECTION .data
    msg db "Результат: ",0h

SECTION .text
    global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0
    jz _end

    pop eax
    call atoi

    call _calc_func

    loop next

_calc_func:
    sub eax, 1
    mov edx, 10
    mul edx
    add esi, eax
    ret

_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

Рисунок 6 — Исходный код программы

Далее проанализируем код, представленный в методичке листингом 9.3 на предмет ошибок с помощью отладчика. Процесс отладки показан на рисунке 7.

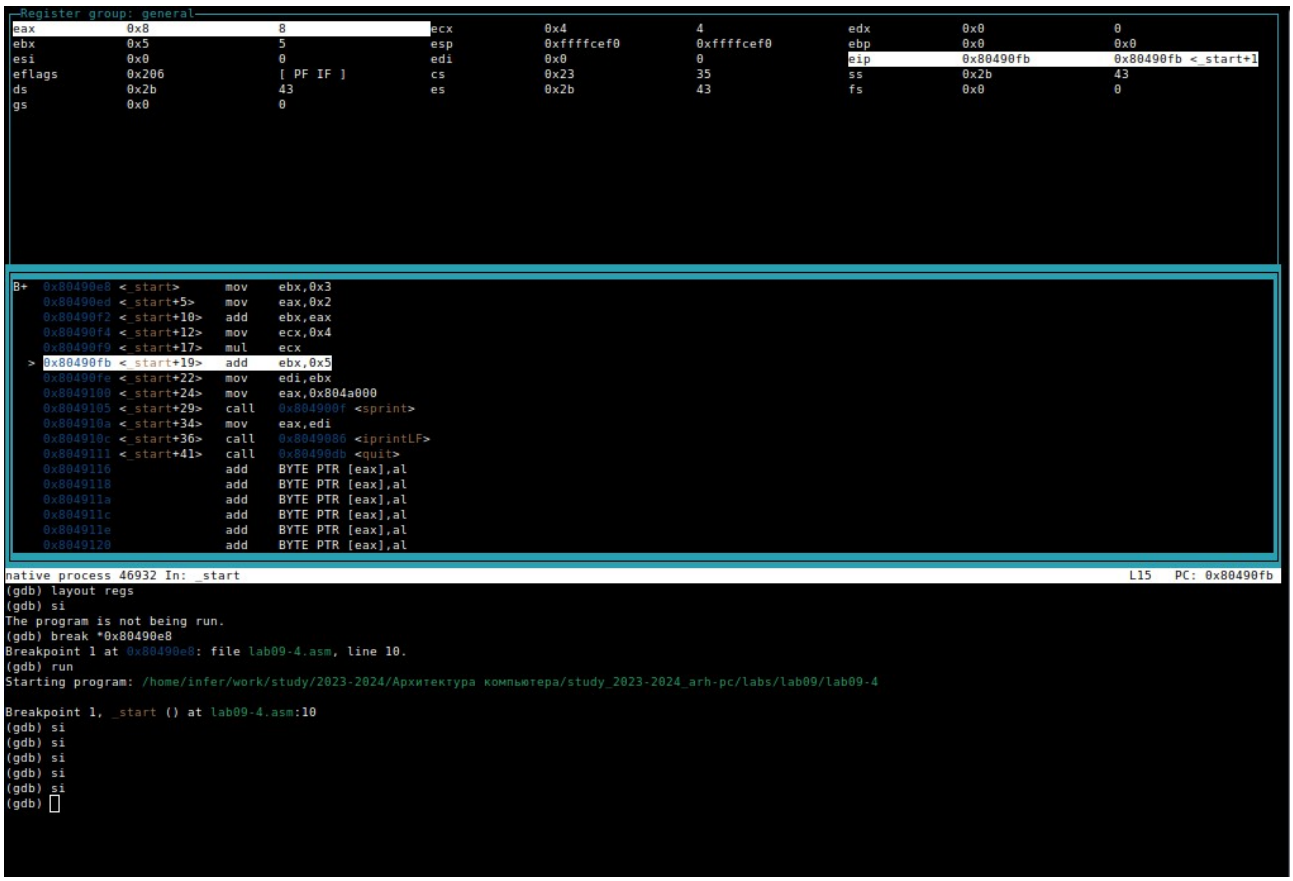


Рисунок 7 — Процесс отладки

По логике работы программы, для получения правильного результат должны сложиться числа 3 и 2, потом сумма умножиться на 4. Из процесса отладки видно, что после 5 итераций в регистре eax оказывается число 8, а не 10, что происходит из-за строки «add ebx, eax», которая должна выглядеть как «add eax, ebx»

Выводы: В ходе лабораторной работы были приобретены навыки использования подпрограмм, а также отладчика gdb.