

Отчет по лабораторной работе № 9

Дисциплина: архитектура компьютеров

Казазаев Даниил Михайлович

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 4 |
| 2 | Задания Лабораторной работы | 5 |
| 3 | Задания Самостоятельной работы | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 5 | Выполнение самостоятельной работы | 24 |
| 5.1 | Задание 1 | 24 |
| 5.2 | Задание 2 | 27 |

Список иллюстраций

| | | |
|------|---|----|
| 4.1 | Создание файла lab9-1.asm | 8 |
| 4.2 | Перенесенный листинг 9.1 | 9 |
| 4.3 | Трансляция и запуск файла lab9-1.asm | 9 |
| 4.4 | Редактирование файла lab9-1.asm | 10 |
| 4.5 | Трансляция и запуск файла lab9-1.asm | 10 |
| 4.6 | Создание файла lab9-2.asm | 12 |
| 4.7 | Перенесенный листинг 9.2 | 13 |
| 4.8 | Трансляция файла lab9-2.asm | 13 |
| 4.9 | Запуск файл в обложке GDB. | 14 |
| 4.10 | Установка брейкпоинта и запуск программы | 14 |
| 4.11 | Просмотр диссимилированного кода | 15 |
| 4.12 | Режим псевдообработки | 16 |
| 4.13 | Проверка наличия точки останова и установление брейкпоинта по адресу инструкции | 17 |
| 4.14 | До выполнения команды stepi 5 | 18 |
| 4.15 | После выполнения команды stepi 5 | 18 |
| 4.16 | Просмотре и изменении значения msg1 | 19 |
| 4.17 | Изменение первого символа переменной | 19 |
| 4.18 | Вывод в различных форматах | 19 |
| 4.19 | Вывод в различных форматах | 19 |
| 4.20 | Вывод в различных форматах | 19 |
| 4.21 | Копирование файла | 21 |
| 4.22 | Редактирование файла lab9-3.asm | 21 |
| 4.23 | Запуск файл в обложке GDB | 21 |
| 4.24 | Установка брейкпоинта и запуск программы | 21 |
| 4.25 | Позиции стека | 22 |
| 5.1 | Копирование файла task.asm | 24 |
| 5.2 | Редактирую файл | 25 |
| 5.3 | Трансляция и запуск файла | 25 |
| 5.4 | Трансляция и запуск файла task2.asm | 27 |
| 5.5 | Запуск файл в обложке GDB | 28 |
| 5.6 | Поиск ошибки | 29 |
| 5.7 | Исправление ошибки | 30 |
| 5.8 | Трансляция и запуск файла task.asm | 30 |

1 Цель работы

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

2 Задания Лабораторной работы

1. Создать файл lab9-1.asm.
2. Ввести в файл lab9-1.asm текст программы из листинга 9.1. Создать исполняемый файл и проверить его работу.
3. Изменить программу, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul` для вычисления выражения $\mathbf{x}(\mathbf{x}(\mathbf{x}))$, где \mathbf{x} вводится с клавиатуры, $\mathbf{x}(\mathbf{x}) = 2\mathbf{x} + 7$, $\mathbf{x}(\mathbf{x}) = 3\mathbf{x} - 1$.
4. Создать файл lab9-2.asm.
5. Ввести в файл lab9-2.asm текст программы из листинга 9.2. Создать исполняемый файл для работы с GDB.
6. Проверить работу программы, запустив ее в оболочке GDB.
7. Установить брейкпоинт на метке `_start`
8. Посмотреть дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`. Переключитесь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`
9. Перечислить различия отображения синтаксиса машинных команд в режимах ATТ и Intel.
10. Включить режим псевдообработки и проверить наличие точки останова, которая была установлена ранее.
11. Поставить точку останова по адресу инструкции.
12. Выполнить пять инструкций с помощью команды `stepi 5` и указать, какие значения регистров изменились.
13. Посмотреть значение переменной `msg1` по имени.

14. Изменить первый симбол в переменной msg1.
15. Заменить любой симбол в переменной msg2.
16. Вывести в различных форматах значение регистра edx.
17. Объяснить разницу вывода команд p/s \$ebx.
18. Скопировать файл lab8-2.asm из прошлой лабораторной работы, назвав его lab9-2.asm.
19. Создать исполняемый файл из файла lab9-2.asm для работы с GDB.
20. Установить брейкпоинт на метке _start.
21. Посмотреть позиции стека.
22. Объяснить, почему шаг изменения адреса рамен 4-м.

3 Задания Самостоятельной работы

1. Преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $\sin(x)$ как подпрограмму.
2. С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправьте ее в тексте листинга программы 9.3.

4 Выполнение лабораторной работы

Создаю файл lab9-1.asm. (рис. [4.1])

```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ touch lab09-1.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ls
lab09-1.asm  presentation  report
```

Рис. 4.1: Создание файла lab9-1.asm

Ввожу в файл lab9-1.asm текст программы из листинга 9.1. (рис. [4.2])


```

1 %include 'in_out.asm'
2
3 SECTION .data
4     msg: DB 'Введите x:',0
5     result: DB '2x+7=',0
6
7 SECTION .bss
8     x: RESB 80
9     res: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13     _start
14
15 mov eax, msg ; вызов подпрограммы печати сообщения
16 call sprint ; 'Введите x: '
17
18 mov ecx, x
19 mov edx, 80
20 call sread ; вызов подпрограммы ввода сообщения
21
22 mov eax, x ; вызов подпрограммы преобразования
23 call atoi ; ASCII кода в число, `eax=x`
24
25 call _calcul
26
27 mov eax, result
28 call sprint
29 mov eax, [res]
30 call iprintLF
31
32 call quit
33
34 _calcul:
35     mov ebx, 2
36     mul ebx
37     add eax, 7
38     mov[res], eax
39
40     ret
41
42

```

Рис. 4.2: Перенесенный листинг 9.1

Транслирую файл lab9-1.asm в объектный файл, после чего запускаю его. (рис. [4.3])

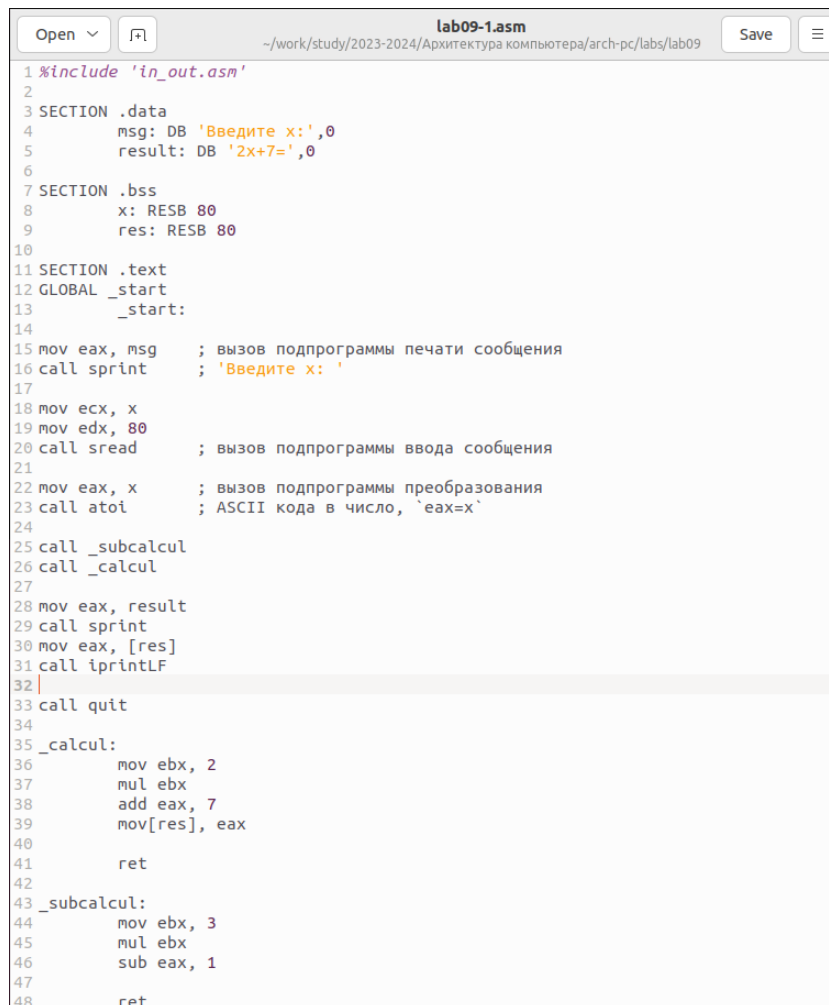
```

dmkazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ nasm -f elf lab09-1.asm
dmkazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ld -m elf_i386 lab09-1.o -o lab09-1
dmkazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ./lab09-1
Введите x:3
2x+7=13

```

Рис. 4.3: Трансляция и запуск файла lab9-1.asm

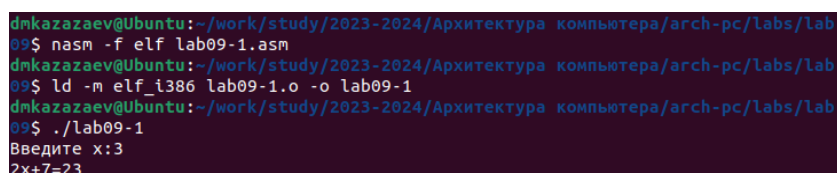
Редактирую файл lab9-1.asm, добавляя подпрограмму _subcalcul в подпрограмму _calcul.(рис. [fig?])



```
1 %include 'in_out.asm'
2
3 SECTION .data
4     msg: DB 'Введите x:',0
5     result: DB '2x+7=',0
6
7 SECTION .bss
8     x: RESB 80
9     res: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13     _start:
14
15     mov eax, msg    ; вызов подпрограммы печати сообщения
16     call sprint     ; 'Введите x: '
17
18     mov ecx, x
19     mov edx, 80
20     call sread      ; вызов подпрограммы ввода сообщения
21
22     mov eax, x
23     call atoi       ; ASCII кода в число, 'eax=x'
24
25     call _subcalcul
26     call _calcul
27
28     mov eax, result
29     call sprint
30     mov eax, [res]
31     call iprintfLF
32
33     call quit
34
35 _calcul:
36     mov ebx, 2
37     mul ebx
38     add eax, 7
39     mov[res], eax
40
41     ret
42
43 _subcalcul:
44     mov ebx, 3
45     mul ebx
46     sub eax, 1
47
48     ret
```

Рис. 4.4: Редактирование файла lab9-1.asm

Транслирую файл lab9-1.asm в объектный файл, после чего запускаю его. (рис. [4.5])



```
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab09-1.asm
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 lab09-1.o -o lab09-1
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ ./lab09-1
Введите x:3
2x+7=23
```

Рис. 4.5: Трансляция и запуск файла lab9-1.asm

Листинг программы lab9-1.asm

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите x:',0
```

```
result: DB '2x+7=',0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
res: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg ; вызов подпрограммы печати сообщения
```

```
call sprint ; 'Введите x: '
```

```
mov ecx, x
```

```
mov edx, 80
```

```
call sread ; вызов подпрограммы ввода сообщения
```

```
mov eax, x ; вызов подпрограммы преобразования
```

```
call atoi ; ASCII кода в число, `eax=x`
```

```
call _subcalcul
```

```
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

```

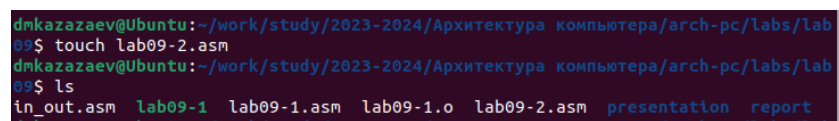
```

_subcalcul:
    mov ebx, 3
    mul ebx
    sub eax, 1

    ret

```

Создаю файл lab9-2.asm. (рис. [4.6])



```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ touch lab09-2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ls
in_out.asm  lab09-1  lab09-1.asm  lab09-1.o  lab09-2.asm  presentation  report

```

Рис. 4.6: Создание файла lab9-2.asm

Переношу в файл lab9-2.asm текст программы из листинга 9.2. (рис. [4.7])



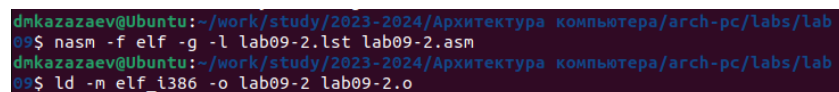
```

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

```

Рис. 4.7: Перенесенный листинг 9.2

Транслирую файл lab9-2.asm в объектный файл для работы с GDB. (рис. [4.8])



```

dmkazazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
dmkazazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ld -m elf_i386 -o lab09-2 lab09-2.o

```

Рис. 4.8: Трансляция файла lab9-2.asm

Запускаю файл в обложке GDB. (рис. [4.9])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/dmkazazaev/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/lab09-2
Hello, world!
[Inferior 1 (process 3738) exited normally]
(gdb)

```

Рис. 4.9: Запуск файл в обложке GDB.

Устанавливаю брейкпоинт на метке `_start` и запускаю программу. (рис. [4.10])

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /home/dmkazazaev/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12      mov eax, 4
(gdb)

```

Рис. 4.10: Установка брейкпоинта и запуск программы

Смотрю дисассимилированный код программы начиная с метки `_start`, после чего переключаюсь на отображение команд с Intel'овским синтаксисом. (рис. [4.11])

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     $0x4,%eax
    0x08049005 <+5>:    mov     $0x1,%ebx
    0x0804900a <+10>:   mov     $0x804a000,%ecx
    0x0804900f <+15>:   mov     $0x8,%edx
    0x08049014 <+20>:   int     $0x80
    0x08049016 <+22>:   mov     $0x4,%eax
    0x0804901b <+27>:   mov     $0x1,%ebx
    0x08049020 <+32>:   mov     $0x804a008,%ecx
    0x08049025 <+37>:   mov     $0x7,%edx
    0x0804902a <+42>:   int     $0x80
    0x0804902c <+44>:   mov     $0x1,%eax
    0x08049031 <+49>:   mov     $0x0,%ebx
    0x08049036 <+54>:   int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     eax,0x4
    0x08049005 <+5>:    mov     ebx,0x1
    0x0804900a <+10>:   mov     ecx,0x804a000
    0x0804900f <+15>:   mov     edx,0x8
    0x08049014 <+20>:   int     0x80
    0x08049016 <+22>:   mov     eax,0x4
    0x0804901b <+27>:   mov     ebx,0x1
    0x08049020 <+32>:   mov     ecx,0x804a008
    0x08049025 <+37>:   mov     edx,0x7
    0x0804902a <+42>:   int     0x80
    0x0804902c <+44>:   mov     eax,0x1
    0x08049031 <+49>:   mov     ebx,0x0
    0x08049036 <+54>:   int     0x80
End of assembler dump.

```

Рис. 4.11: Просмотр диссимилированного кода

Разница в том, что АТТ имена регистров начинаются с символа %, а имена операндов с \$, в то время как в Intel используется привычный нам синтаксис.

Включаю режим псевдообработки с помощью команды “layout asm”, пишу команду “layout regs” и проверяю наличие точки останова, которая была установлена ранее, после чего ставлю точку останова по адресу инструкции. (рис. [4.12])

```
[ Register Values Unavailable ]

0x8049000 <_start>    mov     eax,0x4
0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int     0x80
0x8049016 <_start+22>   mov     eax,0x4
0x804901b <_start+27>   mov     ebx,0x1
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7

exec No process In:                                L??  PC: ??
(gdb) layout regs
(gdb) █
```

Рис. 4.12: Режим псевдообработки


```
[ Register Values Unavailable ]

b+ 0x8049000 <_start>    mov    eax,0x4
    0x8049005 <_start+5> mov    ebx,0x1
    0x804900a <_start+10> mov    ecx,0x804a000
    0x804900f <_start+15> mov    edx,0x8
    0x8049014 <_start+20> int     0x80
    0x8049016 <_start+22> mov    eax,0x4
    0x804901b <_start+27> mov    ebx,0x1
    0x8049020 <_start+32> mov    ecx,0x804a008
    0x8049025 <_start+37> mov    edx,0x7
    0x804902a <_start+42> int     0x80
    0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
    0x8049036 <_start+54> int     0x80

exec No process In:                               L??  PC: ??
(gdb) layout regs
(gdb) i b
Num      Type          Disp Enb Address      What
1        breakpoint    keep y  0x08049000  lab09-2.asm:12
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 25.
(gdb) i b
Num      Type          Disp Enb Address      What
1        breakpoint    keep y  0x08049000  lab09-2.asm:12
2        breakpoint    keep y  0x08049031  lab09-2.asm:25
(gdb)
```

Рис. 4.13: Проверка наличия точки останова и установление брейкпоинта по адресу инструкции

Выполняю пять инструкций с помощью команды `stepi 5`. (рис. [4.15])

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

B+> 0x8049000 <_start> mov    eax,0x4
0x8049005 <_start+5> mov    ebx,0x1
0x804900a <_start+10> mov    ecx,0x804a000
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int    0x80
0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a008
0x8049025 <_start+37> mov    edx,0x7
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038 add    BYTE PTR [eax],al

```

Рис. 4.14: До выполнения команды step 5

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

B+ 0x8049000 <_start> mov    eax,0x4
0x8049005 <_start+5> mov    ebx,0x1
0x804900a <_start+10> mov    ecx,0x804a000
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int    0x80
> 0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a008
0x8049025 <_start+37> mov    edx,0x7
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038 add    BYTE PTR [eax],al

```

Рис. 4.15: После выполнения команды step 5

Изменились значения регистров eax, ecx, edx и ebx.

Смотрю значение переменной msg1 по имени и меняю первый симбол командой set{char}&msg1='h'.(рис. [4.16])

```
0x804a000 <msg1>:      "Hello, "  
(gdb) set {char}&msg1='h'  
(gdb) x/1sb &msg1  
0x804a000 <msg1>:      "hello, "
```

Рис. 4.16: Просмотре и изменении значения msg1

Меняю первый симбол переменной msg2 командой set{char}&msg2='W'. (рис. [4.17])

```
(gdb) set {char}&msg2='W'  
(gdb) x/1sb &msg2  
0x804a008 <msg2>:      "World!\n\034"  
(gdb)
```

Рис. 4.17: Изменение первого символа переменной

Вывожу в различных форматах значение регистра edx.(рис. [4.18],[4.19],[4.20])

```
(gdb) p/f $edx  
$1 = 1.12103877e-44
```

Рис. 4.18: Вывод в различных форматах

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$2 = 50
```

Рис. 4.19: Вывод в различных форматах

```
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$3 = 2
```

Рис. 4.20: Вывод в различных форматах

Листинг программы lab9-2.asm

```
SECTION .data

msg1: db "Hello, ", 0x0
msg1Len: equ $ - msg1

msg2: db "world!", 0xa
msg2Len: equ $ - msg2
```

```
SECTION .text

global _start
```

```
_start:

    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, msg1Len
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, msg2Len
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Копирую файл lab8-2.asm из прошлой лабораторной работы, назвав его lab9-3.asm. (рис. [4.21])

```
dmkazazaev@Ubuntu:~$ cp ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab08/lab8-2.asm ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/Lab09/lab09-3.asm
```

Рис. 4.21: Копирование файла

Транслирую файл lab9-3.asm в объектный файл для работы с GDB. (рис. [4.22])

```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 4.22: Редактирование файла lab9-3.asm

Запускаю файл в обложке GDB. (рис. [4.23])

```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab09$ gdb --args lab09-3 arg1 arg 2 'arg 3'
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
```

Рис. 4.23: Запуск файл в обложке GDB

Устанавливаю брейкпоинт на метке _start и запускаю программу указывая аргументы. (рис. [4.24])

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/dmkazazaev/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab09/lab09-3 arg1 arg 2 arg\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5      pop     ecx ; Извлекаем из стека в `ecx` количество
```

Рис. 4.24: Установка брейкпоинта и запуск программы

Смотрю позиции стека.(рис. [4.25])

```

(gdb) x/x $esp
0xffffd040: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd21f: "/home/dmkazazaev/work/study/2023-2024/Архитектура компьютера/arg
ch-pc/labs/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd28c: "arg1"
(gdb) x/s *(void**)(esp + 12)
0xffffd291: "arg"
(gdb) x/s *(void**)(esp + 16)
0xffffd295: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd297: "arg 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>

```

Рис. 4.25: Позиции стека

Шаг изменения адреса равен 4, т.к. количество аргументов командной строки равно 4.

Листинг программы lab9-3.asm

```

#include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
             ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
             ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
             ; аргументов без названия программы)
next:
    cmp ecx, 0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку `_end`)

    pop eax ; иначе извлекаем аргумент из стека
    call printf ; вызываем функцию печати
    loop next ; переход к обработке следующего

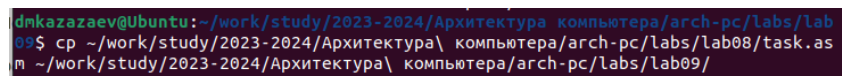
```

```
        ; аргумента (переход на метку `next`)  
_end:  
    call quit
```

5 Выполнение самостоятельной работы

5.1 Задание 1

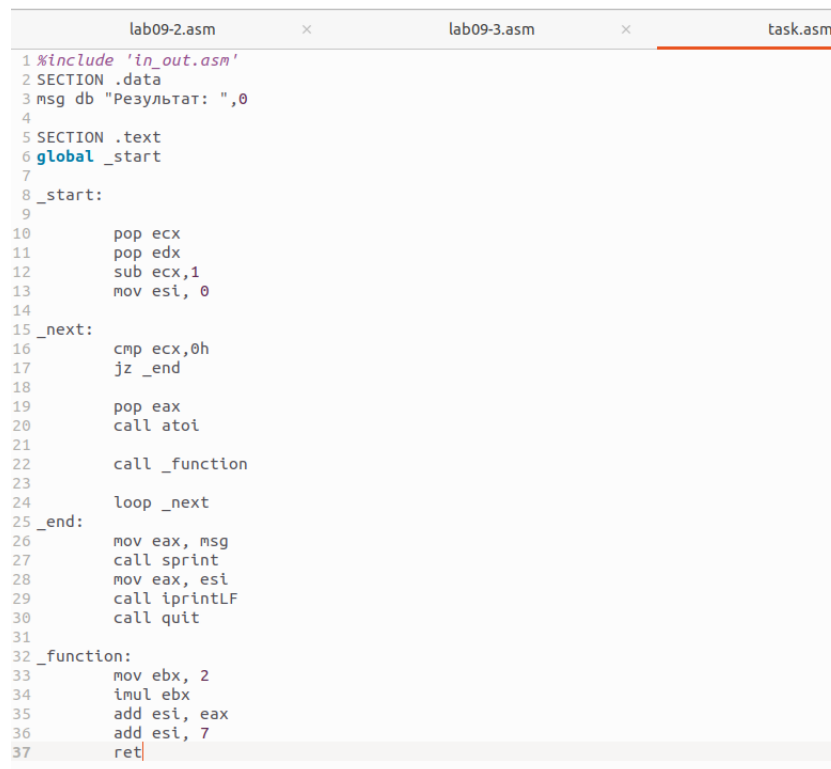
Копирую task.asm из прошлой лабораторной. (рис. [5.1])



```
dnkazazev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ cp ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab08/task.asm ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab09/
```

Рис. 5.1: Копирование файла task.asm

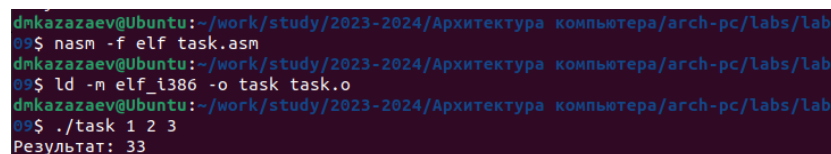
Редактирую файл task.asm добавляя подпрограмму. (рис. [5.2])



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4
5 SECTION .text
6 global _start
7
8 _start:
9
10     pop ecx
11     pop edx
12     sub ecx,1
13     mov esi, 0
14
15 _next:
16     cmp ecx,0h
17     jz _end
18
19     pop eax
20     call atoi
21
22     call _function
23
24     loop _next
25 _end:
26     mov eax, msg
27     call sprint
28     mov eax, esi
29     call iprintLF
30     call quit
31
32 _function:
33     mov ebx, 2
34     imul ebx
35     add esi, eax
36     add esi, 7
37     ret
```

Рис. 5.2: Редактирую файл

Транслирую файл task.asm в объектный файл, после чего запускаю его. (рис. [5.3])



```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ nasm -f elf task.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ld -m elf_i386 -o task task.o
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ./task 1 2 3
Результат: 33
```

Рис. 5.3: Трансляция и запуск файла

Программа работает корректно.

Листинг программы task.asm

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
```

```

SECTION .text
global _start

_start:

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

_next:
    cmp ecx,0h
    jz _end

    pop eax
    call atoi

    call _function

    loop _next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit

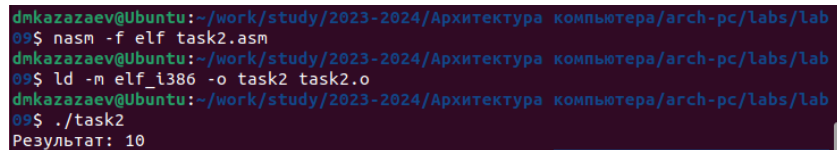
_function:

```

```
mov ebx, 2
imul ebx
add esi, eax
add esi, 7
ret
```

5.2 Задание 2

Транслирую файл task2.asm в объектный файл, после чего запускаю его. (рис. [5.4])



```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ nasm -f elf task2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ld -m elf_i386 -o task2 task2.o
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
09$ ./task2
Результат: 10
```

Рис. 5.4: Трансляция и запуск файла task2.asm

Программа работает некорректно, так как значение, которое она должна вывести должно быть 25.

Запускаю файл в обложке GDB, заранее установив точки останова на инструкциях, в которых происходит математика. (рис. [5.5])

```
Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x5      5
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+12>

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>    mov     $0x2,%eax
B+ 0x80490f2 <_start+10> add     %eax,%ebx
B+ 0x80490f4 <_start+12> mov     $0x4,%ecx
b+ 0x80490f9 <_start+17> mul     %ecx
b+ 0x80490fb <_start+19> add     $0x5,%ebx
b+ 0x80490fe <_start+22> mov     %ebx,%edi
0x8049100 <_start+24> mov     $0x804a000,%eax
0x8049105 <_start+29> call    0x804900f <sprint>
0x804910a <_start+34> mov     %edi,%eax

native process 9074 In: _start L12 PC: 0x80490f4
(gdb) layout regs
(gdb) run
Starting program: /home/dmkazazaev/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/task2

Breakpoint 1, _start () at task2.asm:11
(gdb) continue
Continuing.

Breakpoint 2, _start () at task2.asm:12
(gdb) |
```

Рис. 5.5: Запуск файл в обложке GDB

Ищу ошибку, из-за которой программа выводит неправильный ответ. (рис. [5.6])

```

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fb 0x80490fb <_start+19>

0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>   mov     eax,0x2
B+ 0x80490f2 <_start+10> add     ebx,eax
B+ 0x80490f4 <_start+12> mov     ecx,0x4
B+ 0x80490f9 <_start+17> mul     ecx
B+> 0x80490fb <_start+19> add     ebx,0x5
b+ 0x80490fe <_start+22> mov     edi,ebx
0x8049100 <_start+24> mov     eax,0x804a000
0x8049105 <_start+29> call    0x804900f <sprint>
0x804910a <_start+34> mov     eax,edi

native process 9108 In: _start L14 PC: 0x80490fb

Breakpoint 2, _start () at task2.asm:12
(gdb) continue
Continuing.

Breakpoint 3, _start () at task2.asm:13
(gdb) continue
Continuing.

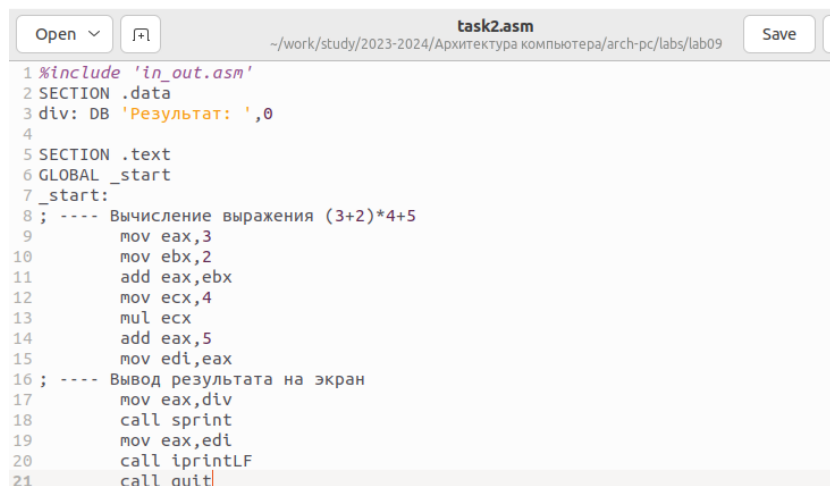
Breakpoint 4, _start () at task2.asm:14
(gdb)

```

Рис. 5.6: Поиск ошибки

Найдя ошибку, которая заключалась в том, что в инструкции `mul ecx` происходит умножение `ecx` на `eax`, то есть 4 на 2, вместо умножения 4 на 5 (регистр `ebx`), так как инструкция `add ebx,eax` не связана с `mul ecx`, но связана инструкция `mov eax,2`.

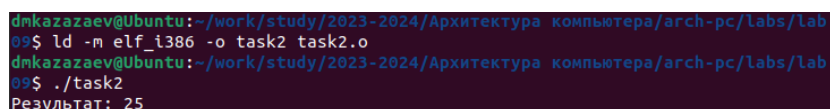
Исправляю ошибку.(рис. [5.7])



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения (3+2)*4+5
9     mov eax,3
10    mov ebx,2
11    add eax,ebx
12    mov ecx,4
13    mul ecx
14    add eax,5
15    mov edi,eax
16 ; ---- Вывод результата на экран
17    mov eax,div
18    call sprint
19    mov eax,edi
20    call iprintLF
21    call quit
```

Рис. 5.7: Исправление ошибки

Транслирую файл task.asm в объектный файл, после чего запускаю его. (рис. [-5.8])



```
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o task2 task2.o
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09$ ./task2
Результат: 25
```

Рис. 5.8: Трансляция и запуск файла task.asm

Успех

Листинг task2.asm

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
    mov eax,3
```

```
mov ebx,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

#Вывод

При выполнении лабораторной работы я приобрел навыки написания программ с использованием подпрограмм и ознакомился с методами отладки при помощи GDB и его основными возможностями.