

# **Отчёт по лабораторной работе №9**

**Дисциплина: архитектура компьютера**

Кузнецова Елизавета Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Задание</b>	<b>7</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>20</b>
<b>6</b>	<b>Выводы</b>	<b>30</b>

# Список иллюстраций

4.1	Создание директории . . . . .	9
4.2	Создание файла . . . . .	9
4.3	Копирование файла . . . . .	9
4.4	Редактирование файла . . . . .	10
4.5	Запуск исполняемого файла . . . . .	10
4.6	Редактирование файла . . . . .	11
4.7	Запуск исполняемого файла . . . . .	11
4.8	Создание файла . . . . .	11
4.9	Редактирование файла . . . . .	12
4.10	Загрузка исполняемого файла в gdb . . . . .	12
4.11	Проверка работы программы . . . . .	12
4.12	Запуск программы . . . . .	13
4.13	Просмотр дисассимилированного кода программы . . . . .	13
4.14	Просмотр дисассимилированного кода программы в режиме intel . . . . .	14
4.15	Режим псевдографики . . . . .	15
4.16	Просмотр точек останова . . . . .	15
4.17	Добавление точки останова . . . . .	16
4.18	Изменение значений регистров . . . . .	16
4.19	Вывод значений переменных . . . . .	17
4.20	Изменение значения переменной . . . . .	17
4.21	Изменение значения переменной . . . . .	17
4.22	Вывод в различных форматах значение регистра edx . . . . .	17
4.23	Изменение значения регистра ebx . . . . .	18
4.24	Загрузка исполняемого файла в gdb . . . . .	18
4.25	Установка точки останова . . . . .	18
4.26	Просмотр в регистре esp . . . . .	19
5.1	Создание файла . . . . .	20
5.2	Редактирование файла . . . . .	21
5.3	Запуск исполняемого файла . . . . .	21
5.4	Создание файла . . . . .	21
5.5	Редактирование файла . . . . .	22
5.6	Запуск исполняемого файла . . . . .	22
5.7	Загрузка исполняемого файла в gdb и проверка работы программы . . . . .	23
5.8	Запуск программы . . . . .	23
5.9	Просмотр дисассимилированного кода программы . . . . .	23
5.10	Просмотр дисассимилированного кода программы в режиме intel . . . . .	24

5.11	Режим псевдографики . . . . .	25
5.12	Просмотр точек останова . . . . .	25
5.13	Добавление точки останова . . . . .	26
5.14	Нахождение ошибок . . . . .	26
5.15	Нахождение ошибок . . . . .	27
5.16	Редактирование файла . . . . .	28
5.17	Загрузка исполняемого файла в gdb и проверка работы программы	29

## Список таблиц

# 1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

## 2 Задание

1. Ознакомиться с листингами и использовать их при работе gdb.
2. Выполнить задания для самостоятельной работы.
3. Загрузить файлы на Github.

### 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: обнаружение ошибки, поиск её местонахождения, определение причины ошибки, исправление ошибки. Наиболее часто применяют следующие методы отладки: создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения); использование специальных программ-отладчиков. GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB может выполнять следующие действия: начать выполнение программы, задав всё, что может повлиять на её поведение; остановить программу при указанных условиях; исследовать, что случилось, когда программа остановилась; изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.



## 4 Выполнение лабораторной работы

С помощью утилиты `mkdir` создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты `cd` (рис. [4.1]).

```
eakuznecova@dk8n52 ~ $ mkdir ~/work/arch-pc/lab09
eakuznecova@dk8n52 ~ $ cd ~/work/arch-pc/lab09
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создала файл `lab09-1.asm` (рис. [4.2]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab09 $ touch lab09-1.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab09 $ ls
lab09-1.asm
```

Рис. 4.2: Создание файла

Скопировала в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, так как он будет использоваться в других программах (рис. [4.3]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab09 $ cp ~/3арызки/in_out.asm in_out.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm
```

Рис. 4.3: Копирование файла

Открыла созданный файл `lab09-1.asm`, вставила в него программу (рис. [4.4]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-1.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
call sprintf

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

call _calcul ; Вызов подпрограммы _calcul

mov eax, result
call sprintf
mov eax, [res]
call sprintf
call quit

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

ret ; выход из подпрограммы

```

Рис. 4.4: Редактирование файла

Создала исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [4.5]).

```

eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
2x+7=11
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 5
2x+7=17

```

Рис. 4.5: Запуск исполняемого файла

Изменила текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul` (рис. [4.6]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09-1.asm
SECTION .data
msg: db "Введите x: ",0
result: db "f(g(x))=",0

SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call _sprintf

    mov ecx, x
    mov edi, 80
    call _strcpy

    mov eax, x
    call _atoi

    call _subcalcul
    call _calcul ; вызов подпрограммы _calcul

    mov eax, result
    call _sprintf
    mov eax, [res]
    call _printf

    call _quit

; _calcul
_calcul:
    push ebx
    mov ebx, 2
    mul ebx
    add eax, ebx
    pop ebx

    ret ; выход из подпрограммы

; _subcalcul
_subcalcul:
    push ebx
    mov ebx, 2
    mul ebx
    dec ebx
    mov [res], eax
    pop ebx

    ret
```

Рис. 4.6: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [4.7]).

```
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
f(g(x))=6
```

Рис. 4.7: Запуск исполняемого файла

С помощью утилиты touch создала файл lab09-2.asm (рис. [4.8]).

```
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ touch lab09-2.asm
eakuznecova@dk4n62 ~/work/arch-pc/lab09 $ ls
in_out.asm lab09-1 lab09-1.asm lab09-1.o lab09-2.asm
```

Рис. 4.8: Создание файла

Открыла созданный файл lab09-2.asm, вставила в него программу (рис. [4.9]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-2.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

```

Рис. 4.9: Редактирование файла

Создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом '-g'. Загрузила исполняемый файл в отладчик gdb (рис. [4.10]).

```

eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ gdb lab09-2

```

Рис. 4.10: Загрузка исполняемого файла в gdb

Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r) (рис. [4.11]).

```

GNU gdb (Gentoo 12.1 vanilla) 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4626) exited normally]
(gdb)

```

Рис. 4.11: Проверка работы программы

Для более подробного анализа программы установила брейкпоинт на метку `_start` и запустила её (рис. [4.12]).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb) █
```

Рис. 4.12: Запуск программы

Посмотрела дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. [4.13]).

```
(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) █
```

Рис. 4.13: Просмотр дисассимилированного кода программы

Переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. [4.14]). В представлении АТТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении `intel` так записываются адреса вторых аргументов.

```

(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.
(gdb)

```

Рис. 4.14: Просмотр дисассимилированного кода программы в режиме intel

Включила режим псевдографики для более удобного анализа программы (рис. [4.15]).

```
[ 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
0x8049031 <_start+49>   mov     ebx, 0x0
0x8049036 <_start+54>   int     0x80

native process 4718 In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 4.15: Режим псевдографики

Посмотрела информацию о наших точках останова. Сделала это с помощью команды `info breakpoints` (кратко `i b`) (рис. [4.16]).

```
native process 4718 In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) i b
Num      Type             Disp Enb Address      What
1        breakpoint       keep y  0x08049000 lab09-2.asm:11
breakpoint already hit 1 time
(gdb)
```

Рис. 4.16: Просмотр точек останова

Добавила еще одну точку останова по адресу инструкции. Посмотрела информацию о всех установленных точках останова (рис. [4.17]).

```

(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 24.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y 0x08049000 lab09-2.asm:11
          breakpoint already hit 1 time
2        breakpoint     keep y 0x08049031 lab09-2.asm:24
(gdb)

```

Рис. 4.17: Добавление точки останова

Выполнила 5 инструкций с помощью команды step и проследила за изменением значений регистров. Значение регистров eax,ecx,edx,ebx (рис. [4.18]).

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc300 0xffffc300
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       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

native process 4718 In: _start L17 PC: 0x8049016
1 breakpoint keep y 0x08049000 lab09-2.asm:11
  breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 24.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y 0x08049000 lab09-2.asm:11
          breakpoint already hit 1 time
2        breakpoint     keep y 0x08049031 lab09-2.asm:24
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 4.18: Изменение значений регистров

Вывела текущее значение переменных (msg1 и msg2) по имени и по адресу (рис. [4.19]).



```

native process 10955 In: start
(gdb) x/1sb &msg1
0x804a000: "Hello, "
(gdb) x/1sb 0x804a000
0x804a000: "Hello, "
(gdb) x/1sb &msg2
0x804a008: "world!\n\034"

```

Рис. 4.19: Вывод значений переменных

Изменила значение переменной `msg1` с помощью команды `set`, задав ей в качестве аргумента адрес переменной и имя переменной (рис. [4.20]).

```

(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hhlllo, "

```

Рис. 4.20: Изменение значения переменной

Изменила значение переменной `msg2` с помощью команды `set`, задав ей в качестве аргумента адрес переменной (рис. [4.21]).

```

(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>: "Lor d!\n\034"

```

Рис. 4.21: Изменение значения переменной

Вывела в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра `edx` (рис. [4.22]).

```

(gdb) p/s $edx
$1 = 8
(gdb) p/t
$2 = 1000
(gdb) p/x
$3 = 0x8

```

Рис. 4.22: Вывод в различных форматах значение регистра `edx`

С помощью команды `set` изменила значение регистра `ebx` (рис. [4.23]). При попытке задать строчное значение происходит ошибка. Завершила работу в `gdb` командами `continue`, она закончила выполнение программы, и `exit`, она завершила сеанс `gdb`.

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
```

Рис. 4.23: Изменение значения регистра ebx

Скопировала файл из лабораторной 8, создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом '-g'. Загрузила исполняемый файл с аргументами в отладчик gdb, использовала ключ -args (рис. [4.24]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ gdb --args lab09-3 arg1 arg 2 'arg 3'
GNU gdb (Gentoo 12.1 vanilla) 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab09-3...
```

Рис. 4.24: Загрузка исполняемого файла в gdb

Установила точку останова перед первой инструкцией в программе и запустила ее (рис. [4.25]).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 11.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-3 arg1 arg 2 arg\ 3
Breakpoint 1, _start () at lab09-3.asm:11
11      pop ecx      ; Извлекаем из стека в 'ecx' количество
(gdb)
```

Рис. 4.25: Установка точки останова

Посмотрела на содержимое того, что расположено по адресу, находящемуся в регистре esp. Увидела, что число аргументов 5 (рис. [4.26]). Далее посмотрела на все остальные аргументы в стеке. Их адреса располагаются в 4 байтах друг от друга (именно столько занимает элемент стека).

```
(gdb) x/x $esp
0xffffc2e0: 0x05
(gdb) x/s *(void**)( $esp + 4)
0xffffc576: "/afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xffffc5be: "arg1"
(gdb) x/s *(void**)( $esp + 12)
0xffffc5c3: "arg"
(gdb) x/s *(void**)( $esp + 16)
0xffffc5c7: "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffc5c9: "arg 3"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb) █
```

Рис. 4.26: Просмотр в регистре esp

## 5 Выполнение заданий для самостоятельной работы

С помощью утилиты touch создала файл lab09-4.asm (рис. [5.1]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ touch lab09-4.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2      lab09-2.lst  lab09-3      lab09-3.lst  lab09-4.asm
lab09-1     lab09-1.o    lab09-2.asm  lab09-2.o    lab09-3.asm  lab09-3.o
```

Рис. 5.1: Создание файла

Открыла созданный файл lab09-4.asm, вставила в него программу из 8 лабораторной, но добавила подпрограмму для вычисления функции 20 варианта (рис. [5.2]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-4.asm Измен
#include 'in_out.asm'

SECTION .data
f_x db "функция: 3(10+x)",0h
msg db 10, 'результат: ',0h

SECTION .text
global _start

_f:
push ebx
add eax,10
mov ebx, 3
mul ebx
pop ebx
ret

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

next:
cmp ecx,0h
jz _end
pop eax
call atoi
call _f
add esi, eax

loop next

_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintLF

call quit
```

Рис. 5.2: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся правильным (рис. [5.3]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ./lab09-4
функция: 3(10+x)
результат: 0
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4 5
функция: 3(10+x)
результат: 195
```

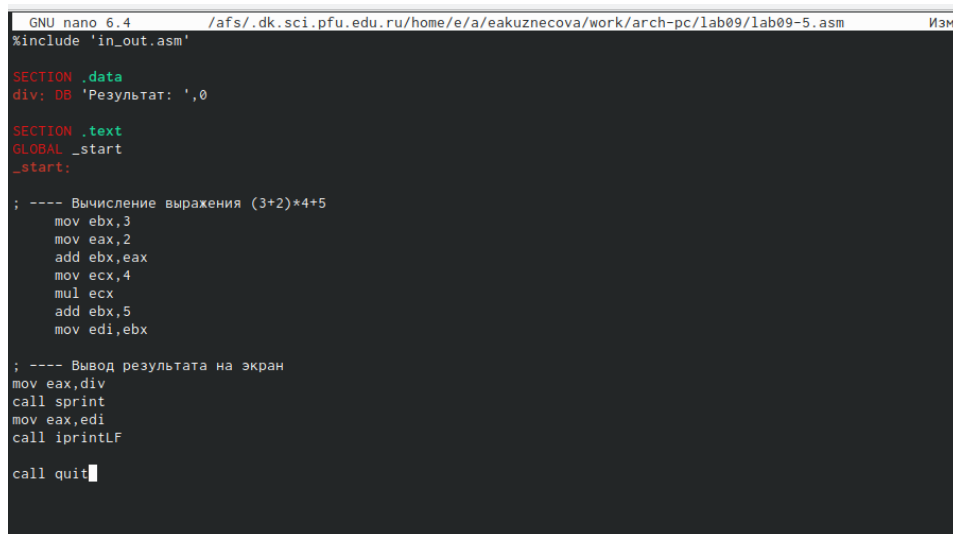
Рис. 5.3: Запуск исполняемого файла

С помощью утилиты touch создала файл lab09-5.asm (рис. [5.4]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ touch lab09-5.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm  lab09-2      lab09-2.lst  lab09-3      lab09-3.lst  lab09-4      lab09-4.o
lab09-1     lab09-1.o     lab09-2.asm  lab09-2.o   lab09-3.asm  lab09-3.o   lab09-4.asm  lab09-5.asm
```

Рис. 5.4: Создание файла

Открыла созданный файл lab09-5.asm, вставила в него программу листинга 9.3 (рис. [5.5]).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-5.asm
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:

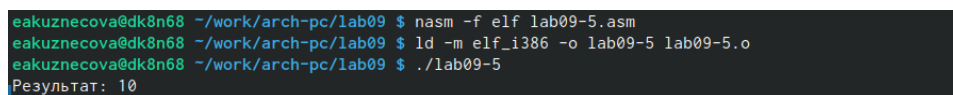
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx

; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF

call quit
```

Рис. 5.5: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Результат данной программы являлся неправильным (рис. [5.6]).



```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10
```

Рис. 5.6: Запуск исполняемого файла

Создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом '-g'. Загрузила исполняемый файл в отладчик gdb. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). Программа работает неправильно (рис. [5.7]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-5.lst lab09-5.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 12.1 vanilla) 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab09-5...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-5
Результат: 10
[Inferior 1 (process 7595) exited normally]
```

Рис. 5.7: Загрузка исполняемого файла в gdb и проверка работы программы

Для более подробного анализа программы установила брейкпоинт на метку `_start` и запустила её (рис. [5.8]).

```
(gdb) break _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 11.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:11
11      mov     ebx,3
(gdb)
```

Рис. 5.8: Запуск программы

Посмотрела дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. [5.9]).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:      mov     $0x3,%ebx
0x080490ed <+5>:      mov     $0x2,%eax
0x080490f2 <+10>:     add     %eax,%ebx
0x080490f4 <+12>:     mov     $0x4,%ecx
0x080490f9 <+17>:     mul     %ecx
0x080490fb <+19>:     add     $0x5,%ebx
0x080490fe <+22>:     mov     %ebx,%edi
0x08049100 <+24>:     mov     $0x804a000,%eax
0x08049105 <+29>:     call    0x804900f <sprint>
0x0804910a <+34>:     mov     %edi,%eax
0x0804910c <+36>:     call    0x8049086 <iprintLF>
0x08049111 <+41>:     call    0x80490db <quit>
End of assembler dump.
(gdb)
```

Рис. 5.9: Просмотр дисассимилированного кода программы

Переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. [5.10]).

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:    mov     ebx,0x3
    0x080490ed <+5>:    mov     eax,0x2
    0x080490f2 <+10>:   add     ebx,eax
    0x080490f4 <+12>:   mov     ecx,0x4
    0x080490f9 <+17>:   mul     ecx
    0x080490fb <+19>:   add     ebx,0x5
    0x080490fe <+22>:   mov     edi,ebx
    0x08049100 <+24>:   mov     eax,0x804a000
    0x08049105 <+29>:   call    0x804900f <sprint>
    0x0804910a <+34>:   mov     eax,edi
    0x0804910c <+36>:   call    0x8049086 <fprintf>
    0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb) |
```

Рис. 5.10: Просмотр дисассимилированного кода программы в режиме intel

Включила режим псевдографики для более удобного анализа программы (рис. [5.11]).



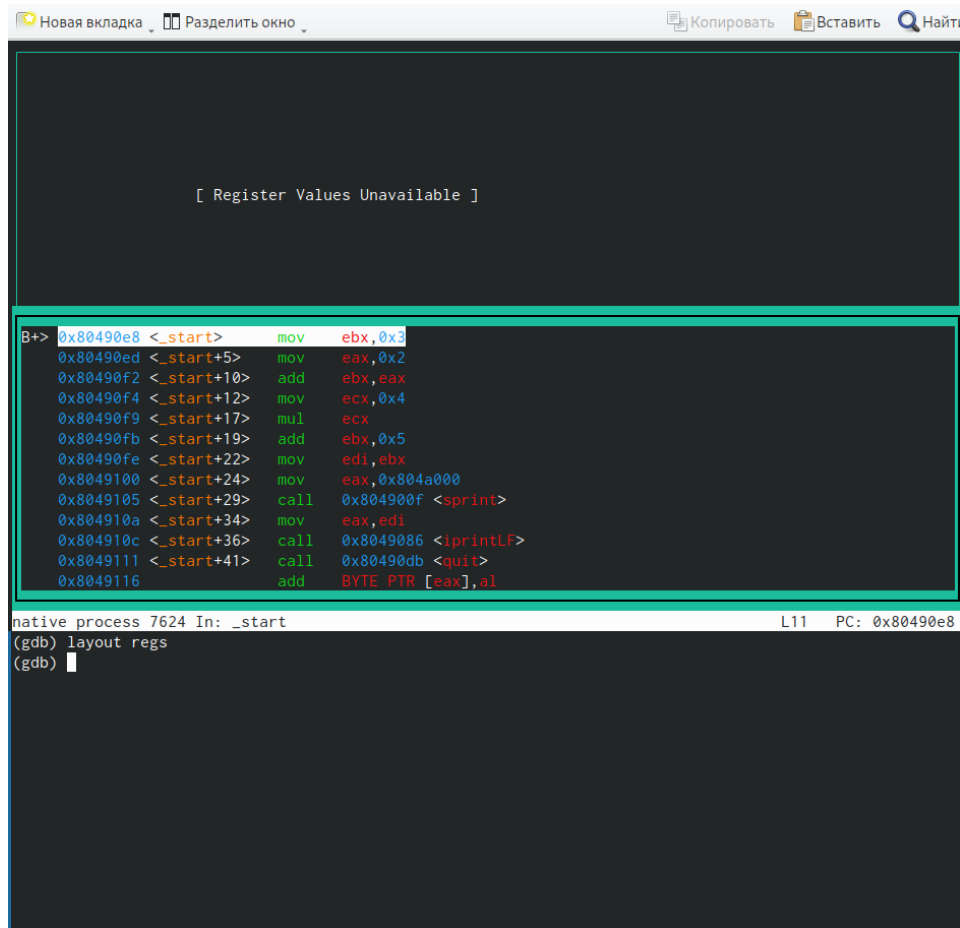


Рис. 5.11: Режим псевдографики

Посмотрела информацию о наших точках останова. Сделала это с помощью команды `info breakpoints` (кратко `i b`) (рис. [5.12]).

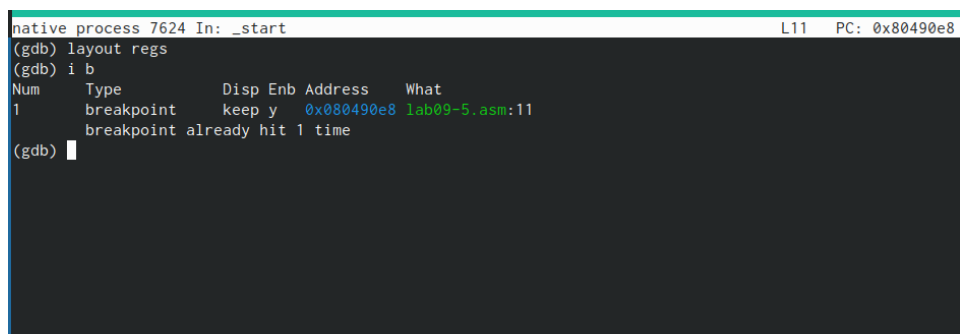


Рис. 5.12: Просмотр точек останова

Добавила еще одну точку останова по адресу инструкции.Посмотрела информацию о всех установленных точках останова (рис. [5.13]).

```
(gdb) break *0x804910c
Breakpoint 2 at 0x804910c: file lab09-5.asm, line 23.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x080490e8 lab09-5.asm:11
          breakpoint already hit 1 time
2        breakpoint     keep y   0x0804910c lab09-5.asm:23
(gdb) █
```

Рис. 5.13: Добавление точки останова

Выполнила 6 инструкций с помощью команды stepi и нашла все ошибки (рис. [5.14]) и (рис. [5.15])

```
Register group: general
eax      0x2          2
ecx      0x0          0
edx      0x0          0
ebx      0x5          5
esp      0xffffc2b0   0xffffc2b0
ebp      0x0          0x0
esi      0x0          0
edi      0x0          0
eip      0x80490f4     0x80490f4 <_start+12>
eflags   0x206        [ PF IF ]
cs       0x23         35
ss       0x2b         43

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
> 0x80490f4 <_start+12> mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
0x8049100 <_start+24>   mov     eax,0x804a000
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
b+ 0x804910c <_start+36> call   0x8049086 <iprintlnLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al

native process 7624 In: _start L14 PC: 0x80490f4
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x080490e8 lab09-5.asm:11
          breakpoint already hit 1 time
(gdb) break *0x804910c
Breakpoint 2 at 0x804910c: file lab09-5.asm, line 23.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x080490e8 lab09-5.asm:11
          breakpoint already hit 1 time
2        breakpoint     keep y   0x0804910c lab09-5.asm:23
(gdb) si
(gdb) si
(gdb) si
(gdb) █
```

Рис. 5.14: Нахождение ошибок

```

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffc2b0 0xffffc2b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fe 0x80490fe <_start+22>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     43

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
> 0x80490fe <_start+22> mov     edi,ebx
0x8049100 <_start+24>   mov     eax,0x804a000
0x8049105 <_start+29>   call    0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
b+ 0x804910c <_start+36> call    0x8049086 <iprintfLF>
0x8049111 <_start+41>   call    0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al

native process 7624 In: _start L17 PC: 0x80490fe
breakpoint already hit 1 time
(gdb) break *0x804910c
Breakpoint 2 at 0x804910c: file lab09-5.asm, line 23.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y 0x080490e8 lab09-5.asm:11
      breakpoint already hit 1 time
2      breakpoint      keep y 0x0804910c lab09-5.asm:23
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 5.15: Нахождение ошибок

Ошибка была в строках

```

add ebx, eax
mov ecx, 4
mul ecx
add ebx, 5
mov edi, ebx

```

Изменила текст программы lab09-5.asm, чтобы выводился правильный ответ (рис. [5.16]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-5.asm
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения (3+2)*4+5
    mov ebx,3
    mov eax,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
```

Рис. 5.16: Редактирование файла

Создала новый исполняемый файл программы. Для работы с GDB в исполняемый файл добавила отладочную информацию, провела трансляцию программ с ключом '-g'. Загрузила исполняемый файл в отладчик gdb. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r). Результат вывелся правильно (рис. [5.17]).

```

eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-5.lst lab09-5.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
eakuznecova@dk8n68 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 12.1 vanilla) 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab09-5...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 8939) exited normally]
(gdb) █

```

Рис. 5.17: Загрузка исполняемого файла в gdb и проверка работы программы

## 6 Выводы

В результате выполнения работы, я научилась организовывать код в подпрограммы и познакомилась с базовыми функциями отладчика gdb.