

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Байдина Елизавета Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация программы	7
3.2	Отладка программ	9
3.3	Добавление точек останова	13
3.4	Работа с данными программы в GDB	15
3.5	Обработка аргументов командной строки в GDB	17
3.6	Выполнение самостоятельной работы	19
	Вывод	24

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Текст программы	8
3.3	Работа программы	9
3.4	Измененный текст программы	10
3.5	загрузка	11
3.6	запуск	11
3.7	запуск	11
3.8	просмотр	12
3.9	переключение	12
3.10	включение режима псевдографики	13
3.11	Проверка	14
3.12	определение адреса	14
3.13	Просмотр информации	15
3.14	просмотр значения	16
3.15	меняю значение	16
3.16	изменение значения	17
3.17	копирование файла	17
3.18	создание файла	17
3.19	Запуск	18
3.20	устанавливаю точку останова	18
3.21	проверка	18
3.22	просмотр позиций стекла	19
3.23	Текст программы	20
3.24	Запуск программы	21
3.25	Текст программы	21
3.26	Запуск программы	22
3.27	Запуск программы в отладчике	22
3.28	Анализ регистров	23

Список таблиц

1 Цель работы

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

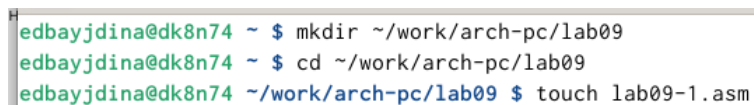
2 Задание

1. Реализация программы
2. Отладка программ
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Выполнение самостоятельной работ

3 Выполнение лабораторной работы

3.1 Реализация программы

Согласно заданию создаю каталог lab09-1.asm



```
edbayjdina@dk8n74 ~ $ mkdir ~/work/arch-pc/lab09
edbayjdina@dk8n74 ~ $ cd ~/work/arch-pc/lab09
edbayjdina@dk8n74 ~/work/arch-pc/lab09 $ touch lab09-1.asm
```

Рис. 3.1: Создание каталога и файла

Ввожу программу из листинга 9.1

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintLF
26 call quit
27 ;-----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx, 2
32 mul ebx
33 add eax, 7
34 102 Демидова А. В.
35 Архитектура ЭВМ
36 mov [res], eax
37 ret ; выход из подпрограммы

```

Рис. 3.2: Текст программы

Запускаю программу

[H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

Рис. 3.3: Работа программы

3.2 Отладка программ

Создаю программу и ввожу текст программы

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

Рис. 3.4: Измененный текст программы

Загружаю исполняемый файл в отладчик gbd и для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается

выполнение любой ассемблерной программы, и запускаю её.

```
edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm>
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...
(No debugging symbols found in lab09-2)
(gdb) █
```

Рис. 3.5: загрузка

```
Reading symbols from lab09-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/d/edbayjdina/work/arch-pc/lab09-2/lab09-2
Hello, world!
[Inferior 1 (process 4626) exited normally]
(gdb) █
```

Рис. 3.6: запуск

```
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/d/edbayjdina/work/arch-pc/lab09-2/lab09-2
Hello, world!
[Inferior 1 (process 4626) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) █
```

Рис. 3.7: запуск

Посматриваю дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`.

```

(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. 3.8: просмотр

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`

```

(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. 3.9: переключение

Включаю режим псевдографики для более удобного анализа программы

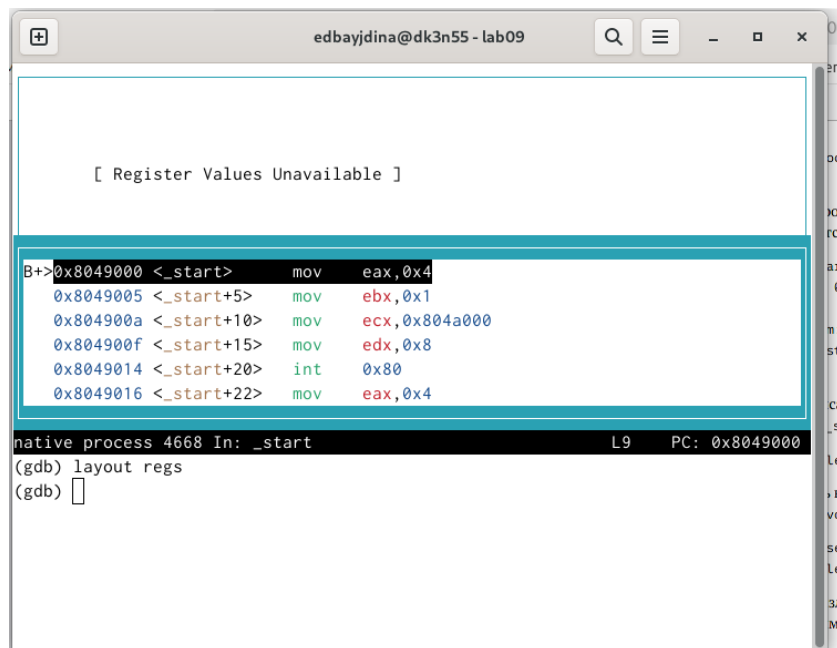


Рис. 3.10: включение режима псевдографики

3.3 Добавление точек останова

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверяю это с помощью команды `info breakpoints` (кратко `i b`):

```

[ 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

native process 4668 In: _start          L9    PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y 0x08049000 lab09-2.asm:9
      breakpoint already hit 1 time
(gdb)

```

Рис. 3.11: Проверка

Определяю адрес предпоследней инструкции (mov ebx,0x0) и устанавливаю точку останова.

```

[ Register Values Unavailable ]

    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
    0x804903a                add     BYTE PTR [eax], al

native process 4668 In: _start          L9    PC: 0x8049000
Num   Type             Disp Enb Address      What
1     breakpoint       keep y 0x08049000 lab09-2.asm:9
      breakpoint already hit 1 time
(gdb) break *0x8049031>
A syntax error in expression, near '<0x8049031>'.
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb)

```

Рис. 3.12: определение адреса

Посмотриваю информацию о всех установленных точках останова

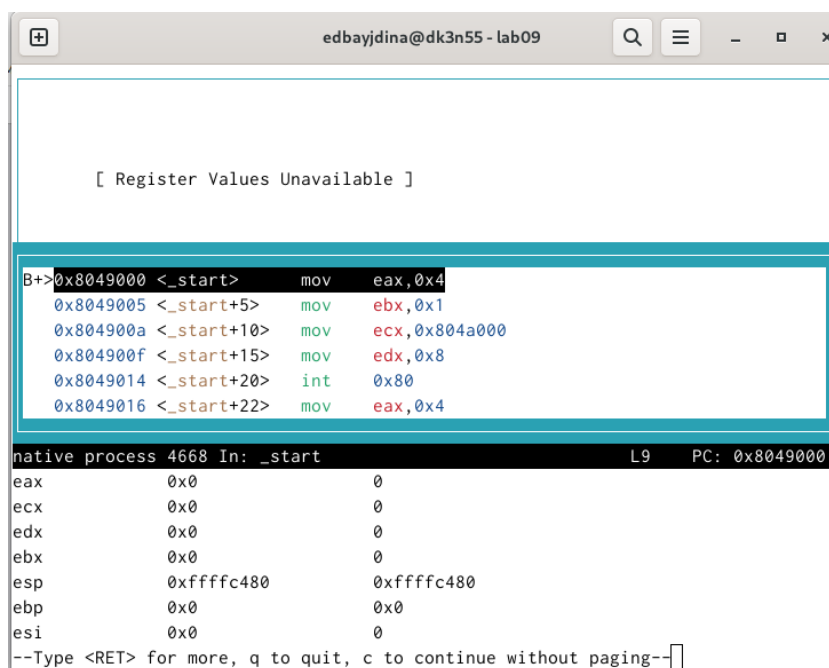
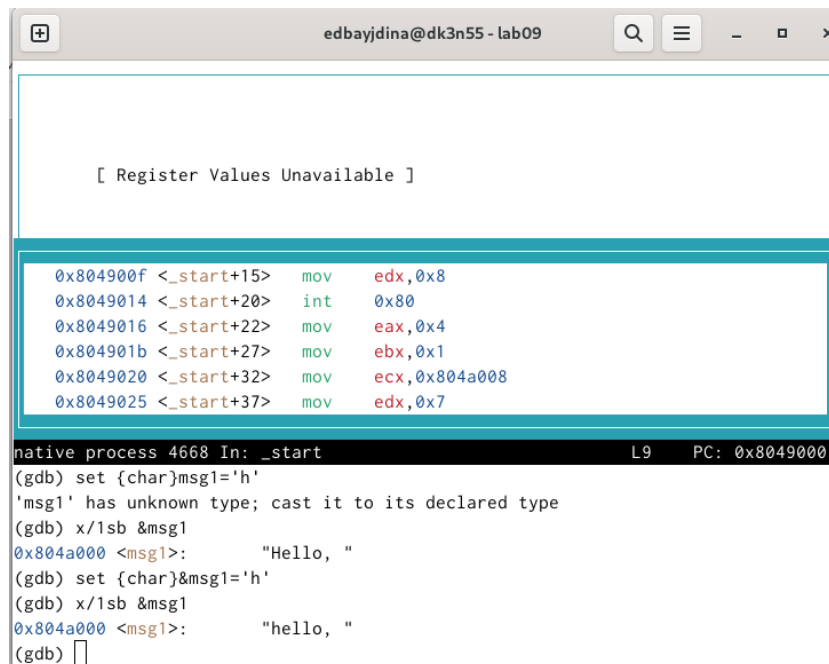


Рис. 3.13: Просмотр информации

3.4 Работа с данными программы в GDB

Просматриваю значение переменной msg1 и меняю первый символ переменной msg1



The screenshot shows a GDB window titled "edbajjdina@dk3n55 - lab09". The main pane displays assembly code with addresses and instructions:

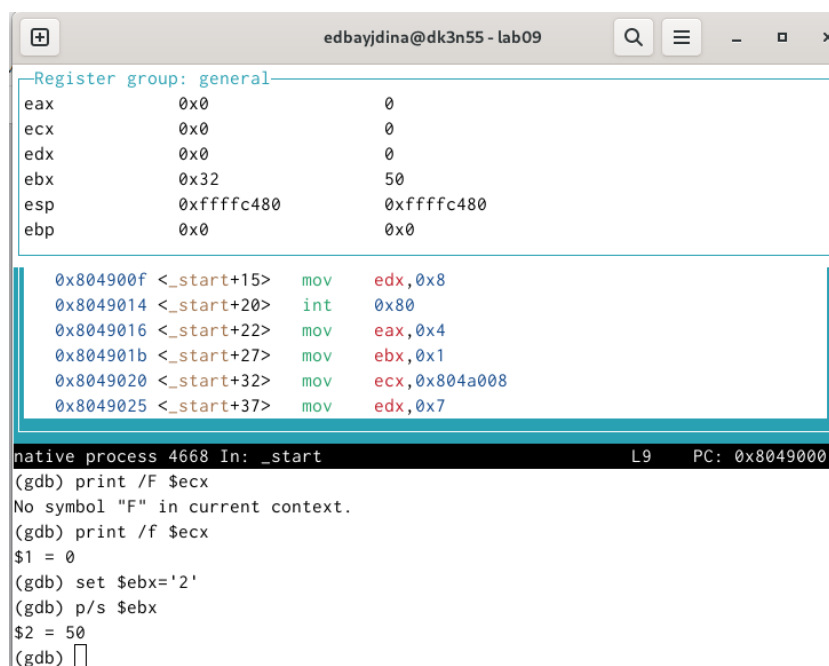
```
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int     0x80
0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a008
0x8049025 <_start+37> mov    edx,0x7
```

Below the assembly code, the status bar shows "native process 4668 In: _start" and "L9 PC: 0x8049000". The console pane shows the following GDB commands and output:

```
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) 
```

Рис. 3.14: просмотр значения

С помощью команды set меняю значение регистра ebx



The screenshot shows a GDB window titled "edbajjdina@dk3n55 - lab09". The main pane displays the "Register group: general" with the following values:

Register	Value
eax	0x0
ecx	0x0
edx	0x0
ebx	0x32
esp	0xfffffc480
ebp	0x0

Below the register values, the assembly code is shown:

```
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
```

The status bar shows "native process 4668 In: _start" and "L9 PC: 0x8049000". The console pane shows the following GDB commands and output:

```
(gdb) print /F $ecx
No symbol "F" in current context.
(gdb) print /f $ecx
$1 = 0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$2 = 50
(gdb) 
```

Рис. 3.15: меняю значение



Рис. 3.16: изменение значения

3.5 Обработка аргументов командной строки в GDB

Копирую файл согласно заданию

```

edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ ls
in_out.asm lab09-1 lab09-1.asm lab09-1.o lab09-2 lab09-2.asm lab09-2.o lab09-3.asm
edbayjdina@dk3n55 ~/work/arch-pc/lab09 $

```

Рис. 3.17: копирование файла

Создаю исполняемый файл и запускаю его

```

edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
edbayjdina@dk3n55 ~/work/arch-pc/lab09 $

```

Рис. 3.18: создание файла

```

edbayjdina@dk3n55 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2013 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...
(gdb) 

```

Рис. 3.19: Запуск

Для начала устанавливаю точку останова перед первой инструкцией в программе и запускаю её.

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/d/edbayjdina/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент 3
nt\ 3

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

```

Рис. 3.20: устанавливаю точку останова

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы)

```

(gdb) x/x $esp
0xffffc440:      0x00000005
(gdb) 

```

Рис. 3.21: проверка

Посматриваю остальные позиции стека – по адресу [esp+4] располагается адрес в памяти, где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д.

```

(gdb) x/x $esp
0xffffc440: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffc696: "/afs/.dk.sci.pfu.edu.ru/home/e/d/edbajjdina/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc6dd: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc6ef: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc700: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc702: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 3.22: просмотр позиций стека

Я посмотрела все позиции стека. По первому адрему хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

3.6 Выполнение самостоятельной работы

Я преобразовала программу из лабораторной работы №8 и реализовала вычисления как подпрограмму.

```

#include 'in_out.asm'

SECTION .data
prim DB "f(x)=12x-7",0
msg db "Результат: ",0

SECTION .text
global _start

_start:
pop ecx.

pop edx.

sub ecx,1.

mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx, 0
jz _end.

pop eax
call atoi
call fir
add esi,eax

loop next.

_end:
mov eax,msg
call sprintf
mov eax,esi
call iprintLF
call quit

fir:
mov ebx,12
mul ebx

```

Рис. 3.23: Текст программы

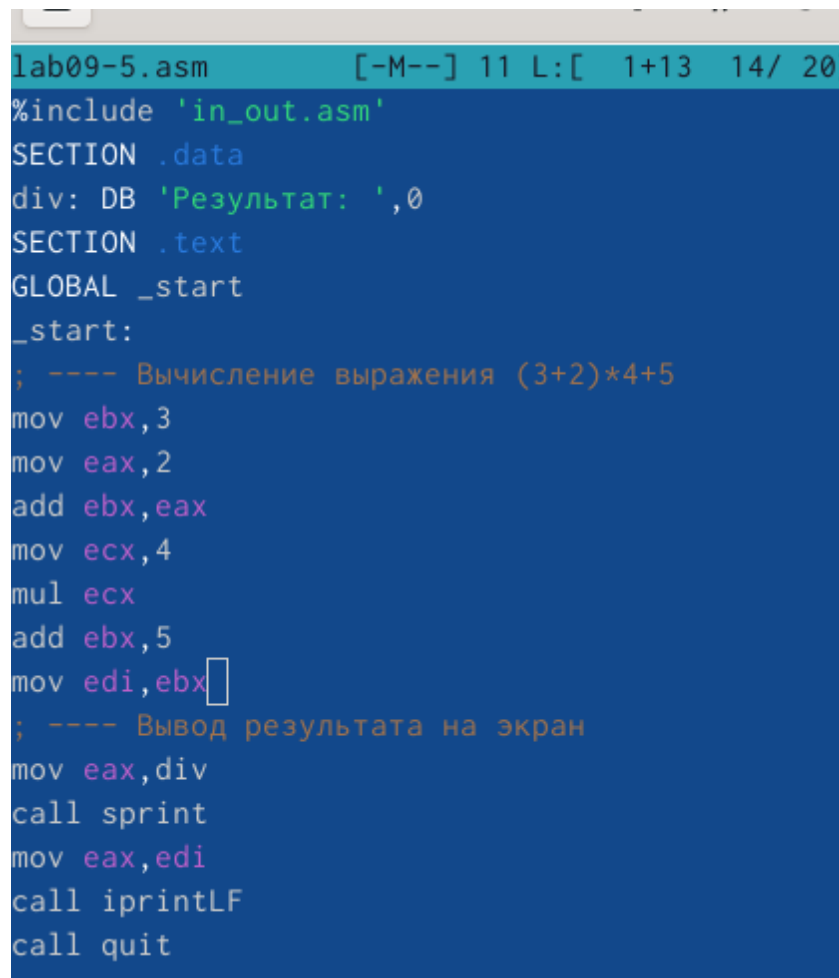
```

edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-4.lst lab09-4.asm
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4
f(x)=12x-7
Результат: 92
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-4.lst lab09-4.asm
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3
f(x)=12x-7
Результат: 51

```

Рис. 3.24: Запуск программы

Я переписала программу и попробовала запустить ее чтобы увидеть ошибку. Ошибка была арифметическая.



```

lab09-5.asm [-M--] 11 L: [ 1+13 14/ 20
#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. 3.25: Текст программы

```

edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-5.lst lab09-5.asm
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ █

```

Рис. 3.26: Запуск программы

После появления ошибки, я запустила программу в отладчике.

```

edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2023 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) b _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 8.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/d/edbayjdina/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:8
8      mov ebx,3
(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 <iprintf>
0x08049111 <+41>:     call    0x80490db <quit>
End of assembler dump.

```

Рис. 3.27: Запуск программы в отладчике

Я открыла регистры и проанализировала их, поняла что некоторые регистры стоят не на своих местах и исправил это.

```
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-5.lst lab09-5.asm
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
edbayjdina@dk6n66 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2023 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) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/d/edbayjdina/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 6516) exited normally]
(gdb) □
```

Рис. 3.28: Анализ регистров

Вывод

Я приобрела навыки написания программ использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.