

# **Лабораторная работа №9**

**Понятие подпрограммы. Отладчик GDB.**

Солдатов Алексей

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация подпрограмм в NASM . . . . .	8
4.2	Отладка программ с помощью GDB . . . . .	13
4.3	Выполнение заданий для самостоятельной работы . . . . .	22
<b>5</b>	<b>Выводы</b>	<b>30</b>
	<b>Список литературы</b>	<b>31</b>

## Список иллюстраций

4.1	Подготовка к работе . . . . .	8
4.2	Ввод команд . . . . .	9
4.3	Перенос файла . . . . .	11
4.4	Создание и запуск файла . . . . .	11
4.5	Изменение текста программы . . . . .	11
4.6	Создание и проверка работы файла . . . . .	13
4.7	Создание файла . . . . .	13
4.8	Ввод команд . . . . .	14
4.9	Работа с файлом . . . . .	15
4.10	Установка точки останова . . . . .	16
4.11	Дисассимилированный код программы . . . . .	16
4.12	Отображение команд с Intel'овским синтаксисом . . . . .	16
4.13	Режим псевдографики . . . . .	17
4.14	info breakpoints . . . . .	17
4.15	Новая точка останова . . . . .	18
4.16	Работа "stepi" . . . . .	18
4.17	info registers . . . . .	18
4.18	Значения msg1 и msg2 . . . . .	19
4.19	Изменение msg1 . . . . .	19
4.20	Изменение msg2 . . . . .	19
4.21	Значение регистра edx . . . . .	20
4.22	Изменение регистра edx . . . . .	20
4.23	Копирование и запуск файла . . . . .	21
4.24	Установка точки останова . . . . .	21
4.25	Позиции стека . . . . .	22
4.26	Создание файла . . . . .	22
4.27	Ввод текста программы . . . . .	23
4.28	Проверка работы программы . . . . .	25
4.29	Создание файла . . . . .	25
4.30	Ввод текста программы . . . . .	25
4.31	Проверка работы программы . . . . .	26
4.32	Дисассемблированный код программы . . . . .	27
4.33	Изменение текста программы . . . . .	28
4.34	Проверка работы программы . . . . .	29

# Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

# 1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. Познакомиться с методами отладки при помощи GDB и его основными возможностями.

## 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

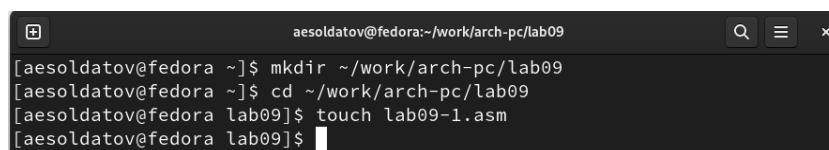
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

## 4 Выполнение лабораторной работы

### 4.1 Реализация подпрограмм в NASM

Создал каталог для программ лабораторной работы №9, перешел в него и создал файл “lab9-1.asm” (рис. 4.1).



```
aesoldatov@fedora:~/work/arch-pc/lab09
[aesoldatov@fedora ~]$ mkdir ~/work/arch-pc/lab09
[aesoldatov@fedora ~]$ cd ~/work/arch-pc/lab09
[aesoldatov@fedora lab09]$ touch lab09-1.asm
[aesoldatov@fedora lab09]$
```

Рис. 4.1: Подготовка к работе

Внимательно изучил текст программы из листинга 9.1 со страницы в ТУИС и ввел в файл “lab9-1.asm” текст программы (рис. 4.2).



```

aesoldatov@fedora:~/work/arch-pc/lab09
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab09/lab09-1.asm Изменён
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы

Имя файла для записи: /home/aesoldatov/work/arch-pc/lab09/lab09-1.asm
^G Справка      M-D Формат DOS   M-A Доп. в начало M-B Резерв. копия
^C Отмена       M-M Формат Mac   M-P Доп. в конец  ^T Обзор

```

Рис. 4.2: Ввод команд

```

#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:
;-----
; Основная программа

```

```

;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы

```

Перенес файл “in\_out.asm” из прошлой папки с лабораторной работой в нынешнюю (рис. 4.3).

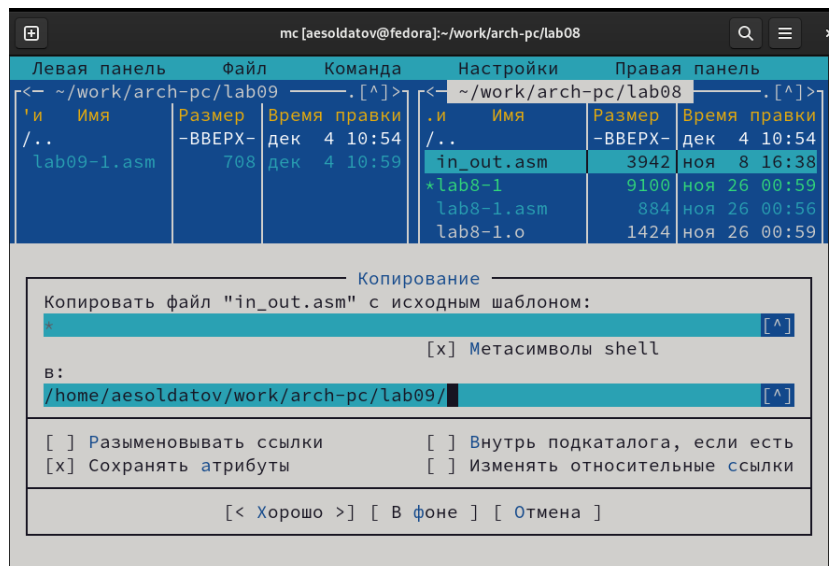


Рис. 4.3: Перенос файла

Создал исполняемый файл и запустил его (рис. 4.4).

```
[aesoldatov@fedora lab09]$ nasm -f elf lab09-1.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[aesoldatov@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
[aesoldatov@fedora lab09]$
```

Рис. 4.4: Создание и запуск файла

Далее изменил текст программы добавив подпрограмму “\_subcalcul” в подпрограмму “\_calcul” согласно заданию (рис. 4.5).

```
_calcul:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx,3
mul ebx
add eax,-1
ret
```

Рис. 4.5: Изменение текста программы

```

%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2*(3x-1)+7=',0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----

mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"_calcul:

```

```

call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx,3
mul ebx
add eax,-1
ret

```

Создал исполняемый файл и запустил его, все работает как надо (рис. 4.6).

```

[aesoldatov@fedora lab09]$ nasm -f elf lab09-1.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[aesoldatov@fedora lab09]$ ./lab09-1
Введите x: 1
2*(3x-1)+7=11
[aesoldatov@fedora lab09]$

```

Рис. 4.6: Создание и проверка работы файла

## 4.2 Отладка программ с помощью GDB

Создал файл “lab9-2.asm” в каталоге “~/work/arch-pc/lab09” (рис. 4.7).

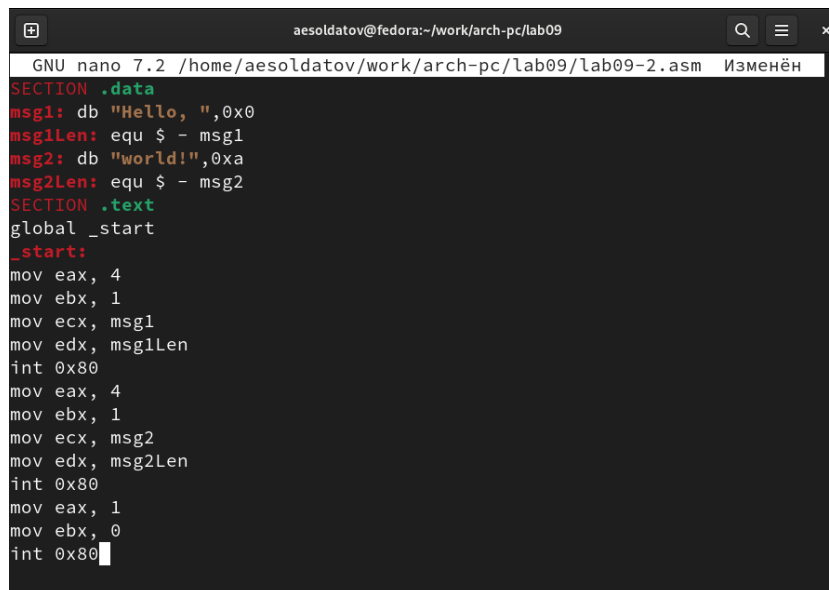
```

[aesoldatov@fedora lab09]$ touch lab09-2.asm
[aesoldatov@fedora lab09]$

```

Рис. 4.7: Создание файла

Внимательно изучил текст программы из листинга 9.2 и ввел его в файл (рис. 4.8).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab09/lab09-2.asm  Изменён
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. 4.8: Ввод команд

```
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

```

Создал исполняемый файл с отладочной информацией, загрузил его в отладчик gdb и проверил его работу, запустив в оболочке gdb, с помощью команды “run”. Программа вывела “Hello, world!” (рис. 4.9).

```

[aesoldatov@fedora lab09]$ nasm -f elf lab09-2.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[aesoldatov@fedora lab09]$ gdb lab09-2
GNU gdb (Fedora Linux) 13.2-10.fc39
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-redhat-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...
(No debugging symbols found in lab09-2)
(gdb) run
Starting program: /home/aesoldatov/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Hello, world!
[Inferior 1 (process 5260) exited normally]
(gdb)

```

Рис. 4.9: Работа с файлом

Для более подробного анализа программы установил брейкпоинт на метку “\_start” и запустил отладку (рис. 4.10).

```
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/aesoldatov/work/arch-pc/lab09/lab09-2

Breakpoint 1, 0x08049000 in _start ()
(gdb)
```

Рис. 4.10: Установка точки останова

Посмотрел дисассимилированный код программы с помощью команды “disassemble” начиная с метки \_start (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)
```

Рис. 4.11: Дисассимилированный код программы

Переключился на отображение команд с Intel’овским синтаксисом, введя команду “set disassembly-flavor intel” (рис. 4.12).

```
(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)
```

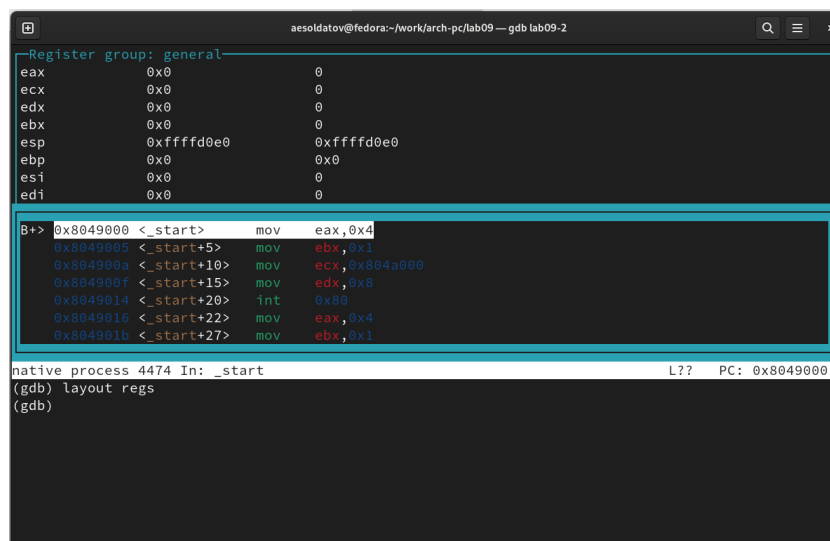
Рис. 4.12: Отображение команд с Intel’овским синтаксисом



Различия отображения синтаксиса машинных команд в режимах АТТ и Intel:

1. Порядок операндов (в режиме АТТ начала источник, потом приемник, а в Intel сначала приемник, потом источник)
2. Отсутствие специальных символов (\$, %) в Intel

Включил режим псевдографики для более удобного анализа программы (рис. 4.13).



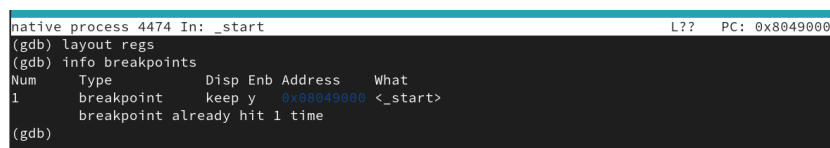
```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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

native process 4474 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 4.13: Режим псевдографики

Проверил работу команды “info breakpoints” (рис. 4.14).



```
native process 4474 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y  0x8049000  <_start>
      breakpoint already hit 1 time
(gdb)
```

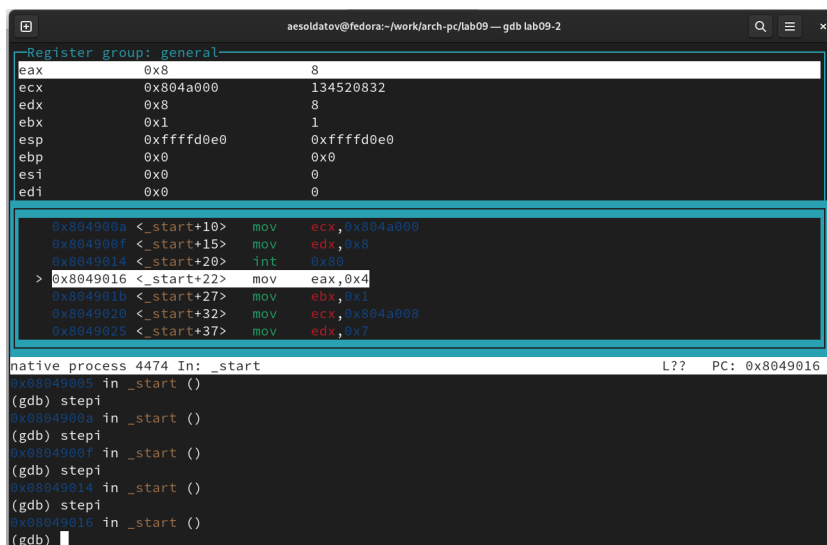
Рис. 4.14: info breakpoints

Установил еще одну точку останова по адресу инструкции и проверил информацию о всех установленных точках останова (рис. 4.15).

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000  <_start>
      breakpoint already hit 1 time
2     breakpoint       keep y   0x08049031  <_start+49>
(gdb)
```

Рис. 4.15: Новая точка останова

Выполнил 5 инструкций с помощью команды “stepi” и проследил за изменением значений регистров (рис. 4.16).



The screenshot shows the GDB interface with the 'Register group: general' window. The registers and their values are:

Register	Type	Value
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0e0	0xffffd0e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

Below the register window, the assembly instructions are shown:

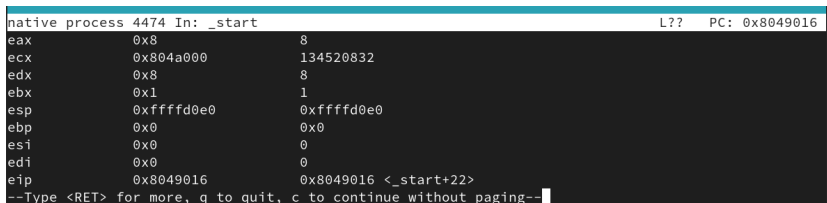
```
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x8
> 0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
```

The command window shows the execution of the 'stepi' command five times, stepping through the instructions at addresses 0x804900a, 0x804900f, 0x8049014, 0x8049016, and 0x804901b.

Рис. 4.16: Работа “stepi”

Изменились значения регистров eax, ebx, ecx, edx

Посмотрел содержимое регистров с помощью команды “info registers” (рис. 4.17).



The screenshot shows the GDB interface with the 'Register group: general' window. The registers and their values are:

Register	Type	Value
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0e0	0xffffd0e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

The command window shows the execution of the 'info registers' command, displaying the current state of the registers.

Рис. 4.17: info registers

Посмотрите значение переменной “msg1” по имени и посмотрел значение переменной “msg2” по адресу. Адрес переменной определил по дизассемблированной инструкции (рис. 4.18).

```
(gdb) x/1sb &msg1
0x804a000:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008:      "world!\n"
(gdb)
```

Рис. 4.18: Значения msg1 и msg2

Изменил первый символ переменной “msg1” (рис. 4.19).

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

Рис. 4.19: Изменение msg1

Изменил первый символ переменной “msg2” (рис. 4.20).

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

Рис. 4.20: Изменение msg2

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. 4.21).

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

Рис. 4.21: Значение регистра edx

С помощью команды “set” измените значение регистра ebx (рис. 4.22).

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

Рис. 4.22: Изменение регистра ebx

Значение регистра отличаются, так как в первом случае мы выводим код символа 2, который в десятичной системе счисления равен 50, а во втором случае выводится число 2, представленное в этой же системе.

Завершаю выполнение программы с помощью команды “continue” и выхожу из GDB с помощью команды “quit”

Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm и создал исполняемый файл. Загрузил исполняемый файл в отладчик, указав аргументы (рис. 4.23).

```
[aesoldatov@fedora lab09]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab
09/lab09-3.asm
[aesoldatov@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[aesoldatov@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 13.2-10.fc39
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-redhat-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 lab09-3...
(gdb)
```

Рис. 4.23: Копирование и запуск файла

Поставил точку останова перед первой инструкцией в программе и запустил ее (рис. 4.24).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) run
Starting program: /home/aesoldatov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент
2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb)
```

Рис. 4.24: Установка точки останова

Просмотрел адрес вершины стека, который хранится в регистре esp, а также другие позиции стека - по адресу (рис. 4.25).

```

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd080: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd246: "/home/aesoldatov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd272: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd284: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd295: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd297: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 4.25: Позиции стека

Количество аргументов командной строки 4, поэтому шаг равен четырем.

## 4.3 Выполнение заданий для самостоятельной работы

Создал файл “samrab1.asm”, скопировав файл из прошлой лабораторной работы (рис. 4.26).

```

[aesoldatov@fedora lab09]$ cp ~/work/arch-pc/lab08/samrab.asm ~/work/arch-pc/lab09/samrab1.asm
[aesoldatov@fedora lab09]$

```

Рис. 4.26: Создание файла

Преобразовал программу из лабораторной работы №8, реализовав вычисление значения функции как подпрограмму (рис. 4.27).

```

cmp ecx,0
je _end
pop eax
call atoi
call _calculation
loop next
_end:
mov eax,msggr
call sprint
mov eax,[rez]
call iprintLF
call quit
_calculation:
add eax,2
mov ebx,5
mul ebx
add eax,[rez]
mov [rez],eax
ret

```

Рис. 4.27: Ввод текста программы

```

#include 'in_out.asm'
SECTION .data
msgf db "Функция: f(x)=5*(2+x)",0
msggr db "Результат: ",0
rez dd 0d
SECTION .bss
rezx resb 80
SECTION .text

```

```

global _start
_start:
mov  eax,msgf
call sprintf
pop  ecx
pop  edx
sub  ecx,1
next:
cmp  ecx,0
je   _end
pop  eax
call atoi
call _calculation
loop next
_end:
mov  eax,msggr
call sprintf
mov  eax,[rez]
call iprintLF
call quit
_calculation:
add  eax,2
mov  ebx,5
mul  ebx
add  eax,[rez]
mov  [rez],eax
ret

```

Создал исполняемый файл и проверил его работу (рис. -4.28).



```
[aesoldatov@fedora lab09]$ nasm -f elf samrab1.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o samrab1 samrab1.o
[aesoldatov@fedora lab09]$ ./samrab1 1 2 3 4
Функция: f(x)=5*(2+x)
Результат: 90
[aesoldatov@fedora lab09]$
```

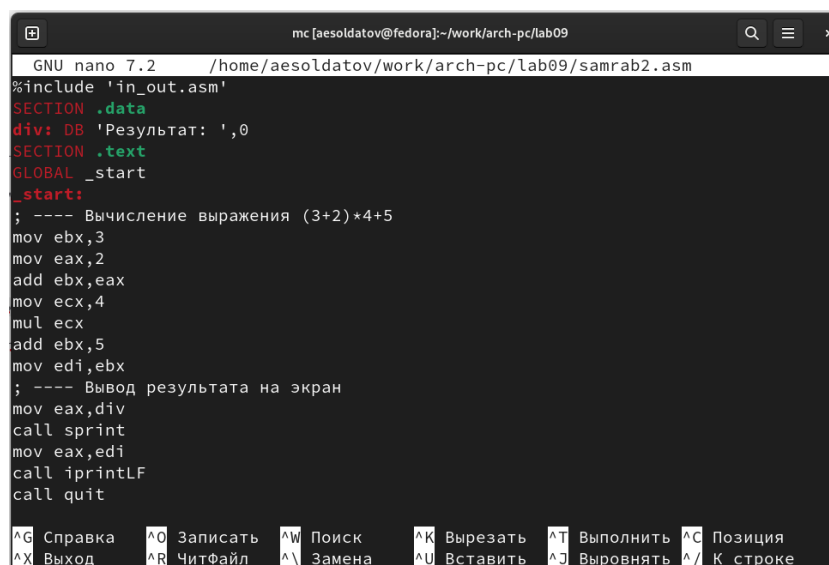
Рис. 4.28: Проверка работы программы

Создал файл “samrab2.asm” (рис. 4.29).

```
(gdb) break *0x08049031
Breakpoint 2 at 0x08049031
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint      keep y   0x08049000  <_start>
          breakpoint already hit 1 time
2        breakpoint      keep y   0x08049031  <_start+49>
(gdb)
```

Рис. 4.29: Создание файла

Ввел текст программы из листинга 9.3 со страницы в ТУИС (рис. 4.30).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab09/samrab2.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

^G Справка  ^O Записать ^W Поиск    ^K Вырезать ^T Выполнить ^C Позиция
^X Выход    ^R ЧитФайл ^_ Замена  ^U Вставить ^J Вровнять ^_ К строке
```

Рис. 4.30: Ввод текста программы

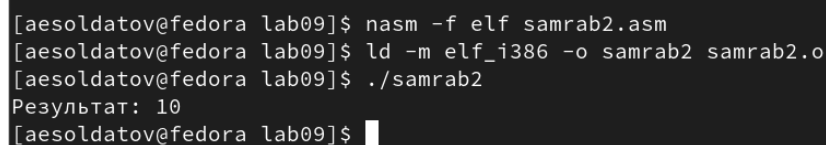
```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
```

```

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

```

Создал исполняемый файл и проверил его работу, программа действительно выдает неверный результат (рис. -4.31).



```

[aesoldatov@fedora lab09]$ nasm -f elf samrab2.asm
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o samrab2 samrab2.o
[aesoldatov@fedora lab09]$ ./samrab2
Результат: 10
[aesoldatov@fedora lab09]$

```

Рис. 4.31: Проверка работы программы

Просмотрел дисассемблированный код программы, поставил точку останова перед прибавлением 5 и открываю значения регистров на данном этапе (рис. -4.32).

```

(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 <iprintf>
   0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb) b *0x080490fb
Breakpoint 1 at 0x80490fb: file samrab2.asm, line 13.
(gdb) run
Starting program: /home/aesoldatov/work/arch-pc/lab09/samrab2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000

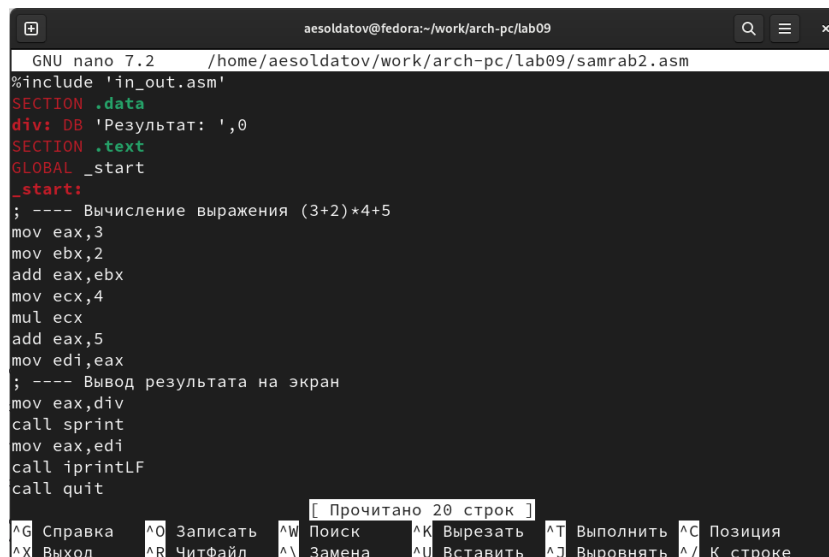
Breakpoint 1, _start () at samrab2.asm:13
13      add ebx,5
(gdb) i r
eax                0x8                8
ecx                0x4                4
edx                0x0                0
ebx                0x5                5

```

Рис. 4.32: Дисассемблированный код программы

Как можно увидеть, регистр `ecx` со значением 4 умножается не на `ebx`, сложенным с `eax`, а только с `eax` со значением 2. Значит нужно поменять значения регистров (например присвоить `eax` значение 3 и просто прибавить 2).

После изменений программа будет выглядеть следующим образом: (рис. -4.33).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab09/samrab2.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
```

Рис. 4.33: Изменение текста программы

```
%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
```

Создал исполняемый файл и проверил его работу, теперь программа работает как надо (рис. -4.34).

```
[aesoldatov@fedora lab09]$ nasm -f elf -g -l samrab2.lst samrab2.asm  
[aesoldatov@fedora lab09]$ ld -m elf_i386 -o samrab2 samrab2.o  
[aesoldatov@fedora lab09]$ ./samrab2  
Результат: 25  
[aesoldatov@fedora lab09]$
```

Рис. 4.34: Проверка работы программы

## 5 Выводы

Приобрел навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.

## Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.  
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.