

# **Отчёт по лабораторной работе 9**

**Дисциплина: архитектура компьютера**

Закиров Нурислам Дамирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация подпрограмм в NASM . . . . .	6
2.2	Отладка программ с помощью GDB . . . . .	9
2.2.1	Точки остановки . . . . .	12
2.2.2	Работа с данными программы в GDB . . . . .	13
2.2.3	Обработка аргументов командной строки в GDB . . . . .	18
2.3	Задание для самостоятельной работы . . . . .	21
<b>3</b>	<b>Выводы</b>	<b>27</b>

## Список иллюстраций

2.1	Код программы lab9-1.asm . . . . .	7
2.2	Компиляция и запуск программы lab9-1.asm . . . . .	7
2.3	Код программы lab9-1.asm . . . . .	8
2.4	Компиляция и запуск программы lab9-1.asm . . . . .	8
2.5	Код программы lab9-2.asm . . . . .	9
2.6	Компиляция и запуск программы lab9-2.asm в отладчике . . . . .	10
2.7	Дизассемблированный код . . . . .	11
2.8	Дизассемблированный код в режиме интел . . . . .	12
2.9	Точка остановки . . . . .	13
2.10	Изменение регистров . . . . .	14
2.11	Изменение регистров . . . . .	15
2.12	Изменение значения переменной . . . . .	16
2.13	Вывод значения регистра . . . . .	17
2.14	Вывод значения регистра . . . . .	18
2.15	Вывод значения регистра . . . . .	20
2.16	Код программы prog-1.asm . . . . .	21
2.17	Компиляция и запуск программы prog-1.asm . . . . .	22
2.18	Код с ошибкой . . . . .	23
2.19	Отладка . . . . .	24
2.20	Код исправлен . . . . .	25
2.21	Проверка работы . . . . .	26

## **Список таблиц**

# 1 Цель работы

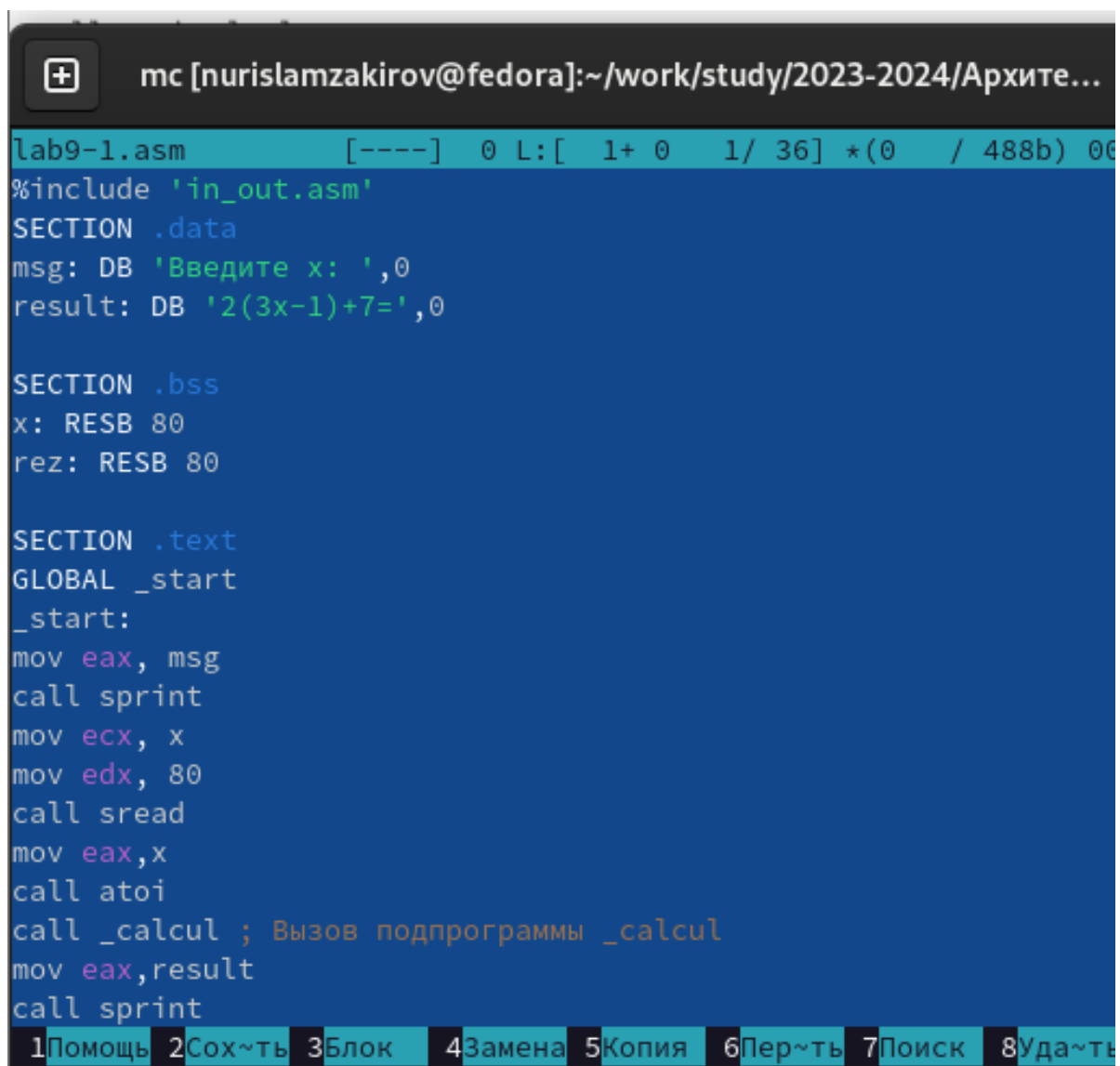
Цель работы состоит в том, чтобы приобрести навыки разработки программ с использованием подпрограмм и приобретение знание методов отладки с помощью GDB и его основных возможностей.

## 2 Выполнение лабораторной работы

### 2.1 Реализация подпрограмм в NASM

Я создал новую директорию и перешел в нее, чтобы выполнить лабораторную работу номер 9. Затем я создал файл lab9-1.asm.

Рассмотрим программу, которая вычисляет арифметическое. Подпрограмма для расчета используется для получения выражения  $f(x) = 2x + 7$ . В этом примере переменная  $x$  вводится с клавиатуры. а сама формула вычисляется в рамках подпрограммы.(рис. [2.1]) (рис. [2.2])



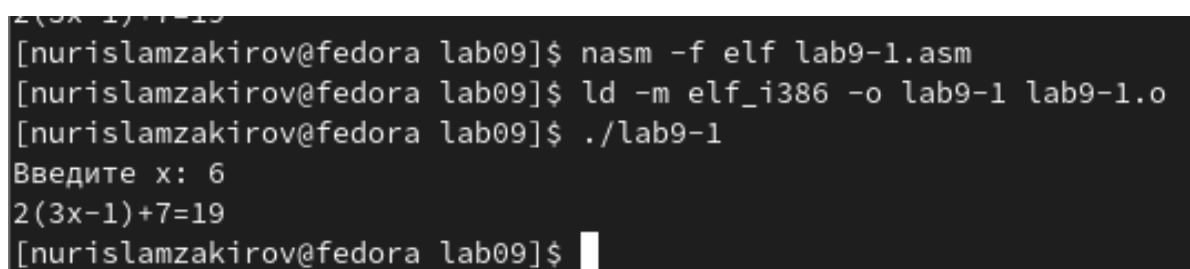
```
mc [nurislamzakirov@fedora]:~/work/study/2023-2024/Архите...
lab9-1.asm [----] 0 L:[ 1+ 0 1/ 36] *(0 / 488b) 00
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить

Рис. 2.1: Код программы lab9-1.asm



```
[nurislamzakirov@fedora lab09]$ nasm -f elf lab9-1.asm
[nurislamzakirov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[nurislamzakirov@fedora lab09]$ ./lab9-1
Введите x: 6
2(3x-1)+7=19
[nurislamzakirov@fedora lab09]$
```

Рис. 2.2: Компиляция и запуск программы lab9-1.asm

Следующим шагом было включение подпрограммы `subcalcul` в подпрограмму `calcul` и внесение изменений в текст программы. Это позволяет вычислить составное выражение  $f(g(x))$ , в котором также вводится значение  $x$  с клавиатуры. Функции определены следующим образом:  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ . (рис. [2.3]) (рис. [2.4])

```

lab9-1.asm  [-----]  0 L:[ 1+ 0 1/ 41] *(0 / 532b) 0037 0x025 [*] [X]
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
  
```

Рис. 2.3: Код программы lab9-1.asm

```

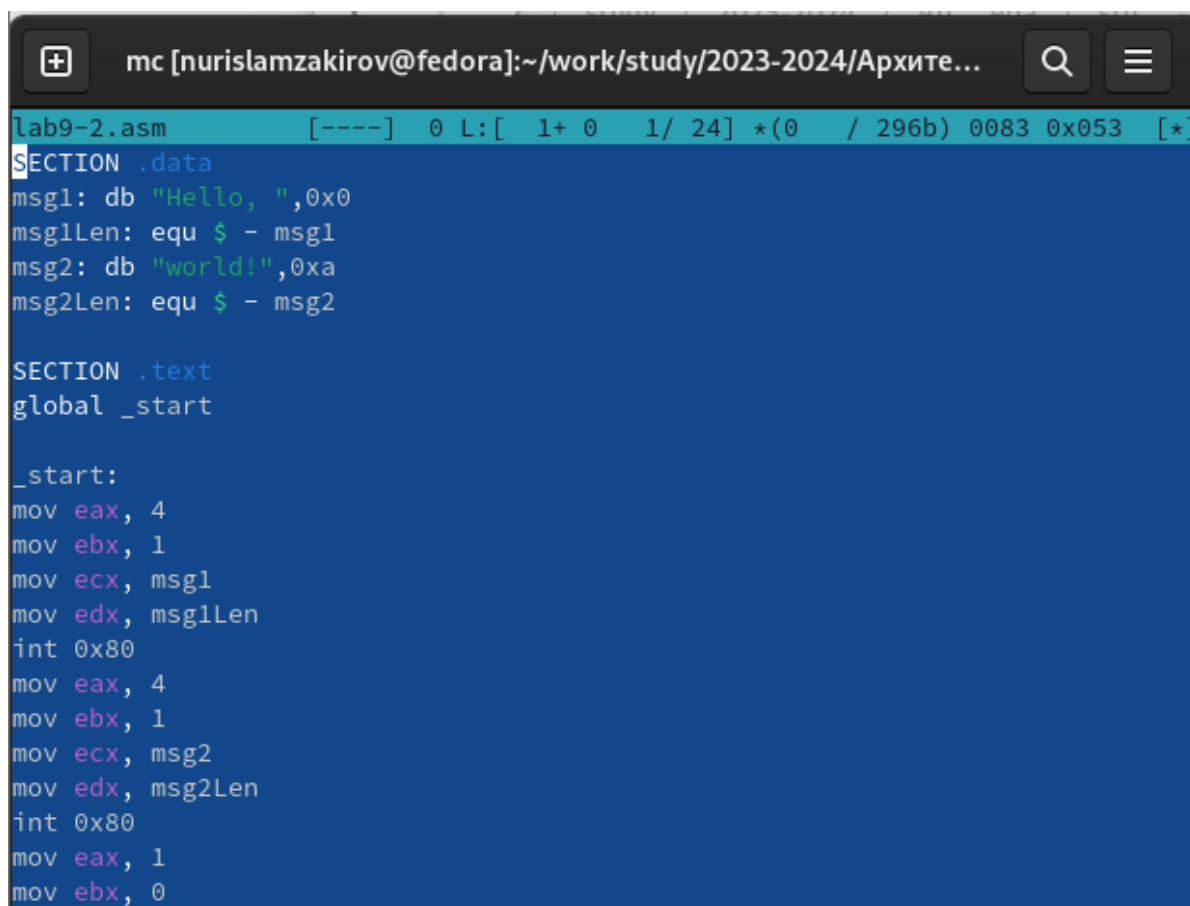
[nurislamzakirov@fedora lab09]$ nasm -f elf lab9-1.asm
[nurislamzakirov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[nurislamzakirov@fedora lab09]$ ./lab9-1
Введите x: 6
2(3x-1)+7=41
  
```

Рис. 2.4: Компиляция и запуск программы lab9-1.asm



## 2.2 Отладка программ с помощью GDB

Я создал файл с именем lab9-2.asm, в котором содержится текст программы из Листинга 9.2. Эта программа отвечает за печать сообщения “Hello world!”. (рис. [2.5])



```
lab9-2.asm [----] 0 L: [ 1+ 0 1/ 24] *(0 / 296b) 0083 0x053 [*]
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
```

Рис. 2.5: Код программы lab9-2.asm

Следующим шагом я скомпилировал файл и получил исполняемый файл. Чтобы добавить отладочную информацию для работы с отладчиком GDB, использовал ключ “-g”.

После чего я загрузил полученный исполняемый файл в отладчик GDB и проверил его работу, запустив программу с помощью команды “r”. (рис. [2.6])

```

[nurislamzakirov@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[nurislamzakirov@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[nurislamzakirov@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
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 lab9-2...
(gdb) r
Starting program: /home/nurislamzakirov/work/study/2023-2024/Архитектура компь
epa/study_2023-2024_arh-pc/lab09/lab9-2

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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!

```

Рис. 2.6: Компиляция и запуск программы lab9-2.asm в отладчике

Чтобы провести более подробный анализ программы, я установил точку остановки на метке «start» и запустил ее. Затем я просмотрел дизассемблированный код программы.(рис. [2.7]) (рис. [2.8])

```

undefined command: y : try help :
(gdb) break _start
Breakpoint 1 at 0x4010e0: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/nurislamzakirov/work/study/2023-2024/Архитектура ко
epa/study_2023-2024_arh-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     $0x4,%eax
      0x004010e5 <+5>:      mov     $0x1,%ebx
      0x004010ea <+10>:     mov     $0x402118,%ecx
      0x004010ef <+15>:     mov     $0x8,%edx
      0x004010f4 <+20>:     int     $0x80
      0x004010f6 <+22>:     mov     $0x4,%eax
      0x004010fb <+27>:     mov     $0x1,%ebx
      0x00401100 <+32>:     mov     $0x402120,%ecx
      0x00401105 <+37>:     mov     $0x7,%edx
      0x0040110a <+42>:     int     $0x80
      0x0040110c <+44>:     mov     $0x1,%eax
      0x00401111 <+49>:     mov     $0x0,%ebx
      0x00401116 <+54>:     int     $0x80
End of assembler dump.
(gdb)

```

Рис. 2.7: Дизассемблированный код

```

End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x004010e0 <+0>:      mov     eax,0x4
    0x004010e5 <+5>:      mov     ebx,0x1
    0x004010ea <+10>:     mov     ecx,0x402118
    0x004010ef <+15>:     mov     edx,0x8
    0x004010f4 <+20>:     int     0x80
    0x004010f6 <+22>:     mov     eax,0x4
    0x004010fb <+27>:     mov     ebx,0x1
    0x00401100 <+32>:     mov     ecx,0x402120
    0x00401105 <+37>:     mov     edx,0x7
    0x0040110a <+42>:     int     0x80
    0x0040110c <+44>:     mov     eax,0x1
    0x00401111 <+49>:     mov     ebx,0x0
    0x00401116 <+54>:     int     0x80
End of assembler dump.
(gdb) █

```

Рис. 2.8: Дизассемблированный код в режиме интел

### 2.2.1 Точки останова

Я использовал команду «info breakpoints» или «i b», чтобы проверить точку останова по имени метки «\_start». Затем я установил еще одну точку останова по адресу инструкции, определив адрес предыдущей инструкции «mov ebx, 0x0». (рис. [2.9])

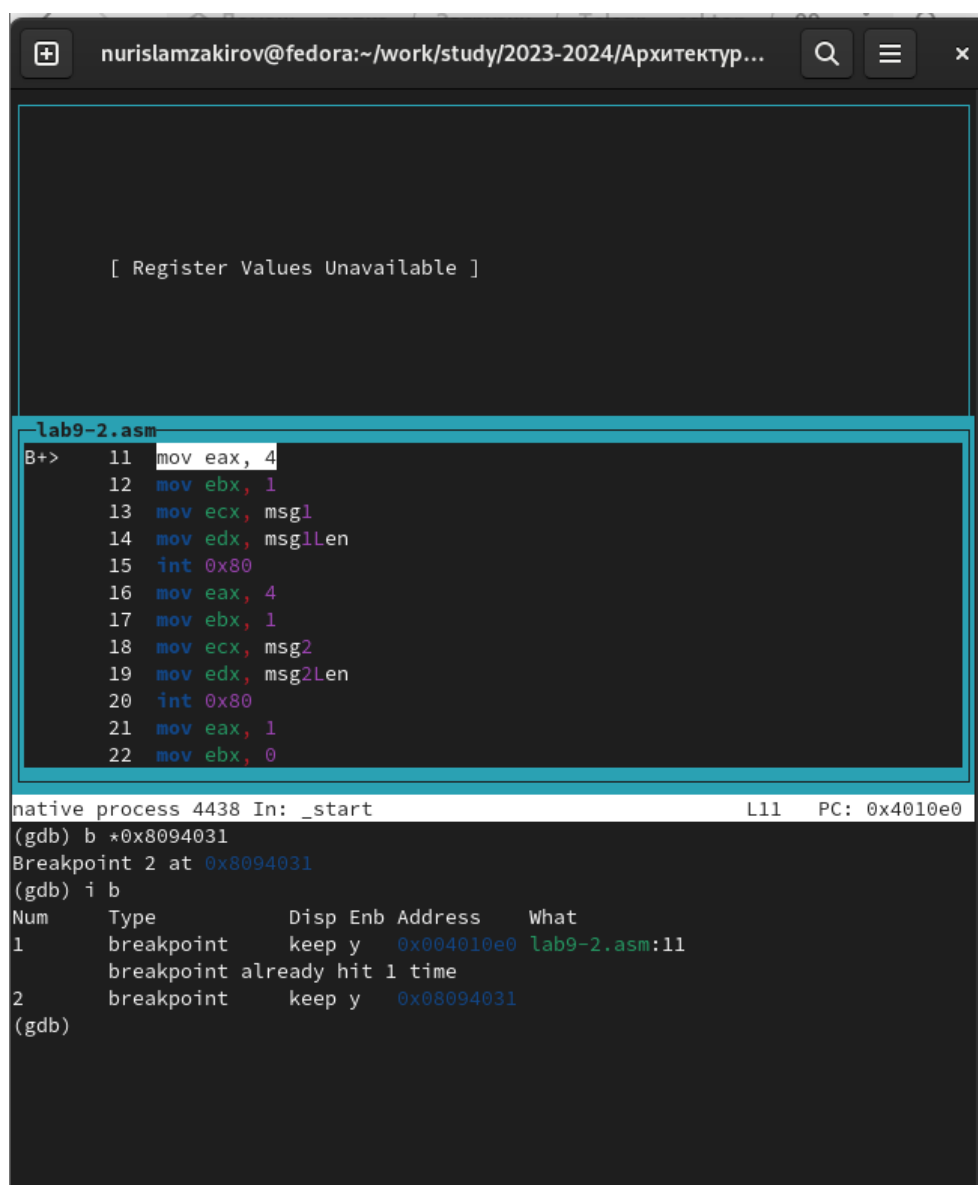


Рис. 2.9: Точка остановки

## 2.2.2 Работа с данными программы в GDB

В отладчике GDB можно просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды 'si' и отследил изменение значений регистров. (рис. [2.10]) (рис. [2.11])

The screenshot shows a GDB debugger window with the title bar "nurislamzakirov@fedora:~/work/study/2023-2024/Архитектур...". The window is divided into several sections:

- Register group: general**: A table showing the values of general-purpose registers.

Register	Value (hex)	Value (dec)
eax	0x4	4
ecx	0x0	0
edx	0x0	0
ebx	0x1	1
esp	0xffffd050	0xffffd050
ebp	0x0	0x0
- lab9-2.asm**: A window showing the assembly code being executed.

```
B+ 11 mov eax, 4
    12 mov ebx, 1
    > 13 mov ecx, msg1
    14 mov edx, msg1Len
    15 int 0x80
    16 mov eax, 4
```
- native process 3632 In: \_start**: A status bar showing the current instruction pointer (L13) and program counter (PC: 0x4010ea).
- Breakpoint 1, \_start () at lab9-2.asm:11**: A message indicating the current breakpoint.
- (gdb) 'si**: The command entered in the GDB prompt.
- Undefined command: "". Try "help".**: A message indicating that the command is not recognized.
- (gdb) si**: The command entered in the GDB prompt.

Рис. 2.10: Изменение регистров

The screenshot shows a GDB terminal window with the title bar "nurislamzakirov@fedora:~/work/study/2023-2024/Архитектур...". The window is divided into two main sections. The top section, titled "Register group: general", displays the current values of several registers:

Register	Value (Hex)	Value (Dec)
eax	0x8	8
ecx	0x402118	4202776
edx	0x8	8
ebx	0x1	1
esp	0xffffd050	0xffffd050
ebp	0x0	0x0

The bottom section shows assembly code being executed, with the instruction at line 16 highlighted:

```
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
> 16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
```

Below the assembly code, the GDB prompt shows the current state of the process:

```
native process 3632 In: _start L16 PC: 0x4010f6
(gdb) 'si
Undefined command: ". Try "help".
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)
```

Рис. 2.11: Изменение регистров

Просмотрел значение переменной msg1 по имени и получил нужные данные.

Команда set была использована для изменения значения ячейки памяти или регистра, указав в качестве аргумента имя регистра или адрес, изменил начало переменной msg1. (рис. [2.12])(рис. [2.13])

```
nurislamzakirov@fedora:~/work/study/2023-2024/Архитектур...
Register group: general
eax      0x8      8
ecx      0x402118  4202776
edx      0x8      8
ebx      0x1      1
esp      0xffffd050 0xffffd050
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x4010f6  0x4010f6 <_start+22>
eflags   0x202    [ IF ]

lab9-2.asm
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
> 16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2Len
20 int 0x80
21 mov eax, 1

native process 3632 In: _start L16 PC: 0x4010f6
0x402118 <msg1>: "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/lsb &msg1
0x402118 <msg1>: "hello, "
(gdb) set {char}0x402118='L'
(gdb) x/lsb &msg1
0x402118 <msg1>: "Lello, "
(gdb) 
```

Рис. 2.12: Изменение значения переменной



The screenshot shows a debugger window with the title bar "nurislamzakirov@fedora:~/work/study/2023-2024/Архитектур...". The window is divided into three main sections:

- Register group: general**: A table showing the current values of general-purpose registers.

Register	Value (Hex)	Value (Dec)
eax	0x8	8
ecx	0x402118	4202776
edx	0x8	8
ebx	0x1	1
esp	0xffffd050	0xffffd050
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x4010f6	0x4010f6 <_start+22>
eflags	0x202	[ IF ]
- lab9-2.asm**: A list of assembly instructions with line numbers. Line 16 is highlighted with a cursor.

```
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
> 16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2Len
20 int 0x80
21 mov eax, 1
```
- native process 3632 In: \_start**: A GDB console window showing the execution of commands and their results.

```
0x402118 <msg1>: "lello, "
(gdb) p/s $ecx
$1 = 4202776
(gdb) p/x $ecx
$2 = 0x402118
(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb) 
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение.  
(рис. [2.14])

The screenshot shows a GDB terminal window with the following content:

```
nurislamzakirov@fedora:~/work/study/2023-2024/Архитектур...  
Register group: general  
eax      0x8      8  
ecx      0x402118  4202776  
edx      0x32     50  
ebx      0x2      2  
esp      0xffffd050 0xffffd050  
ebp      0x0      0x0  
esi      0x0      0  
edi      0x0      0  
eip      0x4010f6  0x4010f6 <_start+22>  
eflags   0x202    [ IF ]  
lab9-2.asm  
13 mov ecx, msg1  
14 mov edx, msg1Len  
15 int 0x80  
> 16 mov eax, 4  
17 mov ebx, 1  
18 mov ecx, msg2  
19 mov edx, msg2Len  
20 int 0x80  
21 mov eax, 1  
native process 3632 In: _start L16 PC:  
$3 = 8  
(gdb) p/t $edx  
$4 = 1000  
(gdb) p/x $edx  
$5 = 0x8  
(gdb) set $edx='2'  
(gdb) p/s $ebx  
$6 = 1  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$7 = 2  
(gdb)
```

Рис. 2.14: Вывод значения регистра

### 2.2.3 Обработка аргументов командной строки в GDB

Скопировал файл lab8-2.asm, который был создан во время выполнения лабораторной работы номер 8, и он содержит программу, которая позволяет выводить аргументы командной строки. скопировал файл и создал исполняемый файл.

Используя ключ `—args`, можно загрузить программу с аргументами в gdb, а

затем загрузить исполняемый файл в отладчик с этими аргументами.

Установил точку останова и запустил первую инструкцию программы.

В регистре esp хранится адрес вершины стека, который содержит количество аргументов командной строки, в том числе имя программы. По этой ссылке можно найти число, показывающее количество аргументов. В данном случае количество аргументов, включая имя программы lab9-3, а также сами аргументы: аргумент1, аргумент2 и «аргумент 3», видно.

Просмотрел остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] - второго и так далее. (рис. [2.15])

```

[nurislamzakirov@fedora lab09]$ nasm -f elf -g -l lab9-03.lst lab9-3.asm
[nurislamzakirov@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[nurislamzakirov@fedora lab09]$ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x4011a8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/nurislamzakirov/work/study/2023-2024/Архитектура компьютера/study_

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.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd000: 0x00000005
(gdb) x/s *(void**)(esp +4)
0xffffd1d4: "/home/nurislamzakirov/work/study/2023-2024/Архитектура компьютера/study_
(gdb) x/s *(void**)(esp +8)
0xffffd228: "аргумент1"
(gdb) x/s *(void**)(esp +12)
0xffffd23a: "аргумент"
(gdb) x/s *(void**)(esp +16)
0xffffd24b: "2"
(gdb) x/s *(void**)(esp +20)
0xffffd24d: "аргумент 3"
(gdb) x/s *(void**)(esp +24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 2.15: Вывод значения регистра

Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]).

## 2.3 Задание для самостоятельной работы

Изменил программы из лабораторной работы No8 (Задание No1 для самостоятельной работы), добавив подпрограмму для вычисления значения функции  $f(x)$ . Мой вариант был 1. (рис. [2.16]) (рис. [2.17])

```
lab9-4.asm      [----]  0 L:[ 1+ 0  1/ 38] *(0
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 15x + 2',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx.
pop edx
sub ecx,1
mov esi, 0

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

loop next

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

_funk:
mov ebx,15
mul ebx
add eax,2
```

Рис. 2.16: Код программы prog-1.asm

```
[nurislamzakirov@fedora lab09]$ nasm -f elf lab9-4.asm
[nurislamzakirov@fedora lab09]$ ld -m elf_i386 lab9-4.o -o lab9-4
[nurislamzakirov@fedora lab09]$ ./lab9-4
f(x)= 15x + 2
Результат: 0
[nurislamzakirov@fedora lab09]$ ./lab9-4 1
f(x)= 15x + 2
Результат: 17
[nurislamzakirov@fedora lab09]$ ./lab9-4 1 2 3 4 5
f(x)= 15x + 2
Результат: 235
[nurislamzakirov@fedora lab09]$
```

Рис. 2.17: Компиляция и запуск программы prog-1.asm

В листинге приведена программа вычисления выражения  $(3 + 2) * 4 + 5$ . При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции `add`. Также обнаружил, что по окончании работы в `edi` отправляется `ebx` вместо `eax`. (рис. [2.18])

```

lab9-5.asm      [----]  0  L:[
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*
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. 2.18: Код с ошибкой

```
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd220 0xffffd220
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x8049100 <_start+24>    mov     eax,0x804a000
0x8049105 <_start+29>    call    0x804900f <sprint>
0x804910a <_start+34>    mov     eax,edi
0x804910c <_start+36>    call    0x8049086 <iprintLF>
0x8049111 <_start+41>    call    0x80490db <quit>

>                                04a000
                                rint>

native process 3583 In: _start L16 PC: 0x8049100
Breakpoint No process In: g-2.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 10
[Inferior 1 (process 3583) exited normally]
(gdb) 
```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. [2.19])

Исправленный код программы (рис. [2.20]) (рис. [2.21])



```
lab9-5.asm      [----]  0  L:[
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*
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. 2.20: Код исправлен

```
nurislamzakirov@fedora:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/lab0...
eax 0x19 25 ecx 0x4 4
edx 0x0 0 ebx 0x3 3
esp 0xffffd050 0xffffd050 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x4011de 0x4011de <_start+22> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

lab9-5.asm
8+ 8 mov ebx,3
14 mov edi,eaxд результата на экран
15 mov eax,div
16 call sprint
17 mov eax,edi
18 call iprintLF
19 call quit
20
21
22
23
24
25
26
27

native process 4011 In: _start L14 PC: 0x4011
Breakpoint 1, _start () at lab9-5.asm:8
Starting program: /home/nurislamzakirov/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/lab09/lab5
5

Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 4011) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

## 3 Выводы

В ходе лабораторной работы, я приобрел навыки разработки программ с использованием подпрограмм и знание методов отладки с помощью GDB и его основных возможностей. # Список литературы 1. Лабораторная работа №7