

Отчёта по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Хохлачёва Полина Дмитриевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание 1	17
4	Задание 2	19
5	Выводы	21

Список иллюстраций

2.1	И в нём новый файл	6
2.2	Заполняем файл	6
2.3	Запускаем файл и проверяем его работу	7
2.4	Редактируем файл	7
2.5	Запускаем файл и смотрим на его работу	7
2.6	Создаём файл	8
2.7	Заполняем файл	8
2.8	Загружаем исходный файл	9
2.9	Запускаем программу командой run	9
2.10	Запускаем программу с брейкпоинтом	9
2.11	Смотрим код программы	10
2.12	Переключаемся на синтаксис Inter	10
2.13	Включаем отображение регистров, их значений и результат дисас- симилирования программы	12
2.14	Используем команду и создаём новую точку останова	12
2.15	Смотрим информацию	12
2.16	Отслеживаем регистры	13
2.17	Смотрим значение переменной	13
2.18	Смотрим значение переменной	13
2.19	Меняем символ	14
2.20	Меняем символ	14
2.21	Значение регистра	14
2.22	Изменяем регистр	14
2.23	Прописываем команды	15
2.24	Копируем	15
2.25	Создаём и запускаем	15
2.26	Устанавливаем точку останова	15
2.27	Изучаем полученные данные	16
3.1	Копируем файл	17
3.2	Изменяем файл	17
3.3	Проверяем работу	18
4.1	Создаём файл	19
4.2	Изменяем файл	19
4.3	Смотрим на работу программы	19
4.4	Меняем файл	20

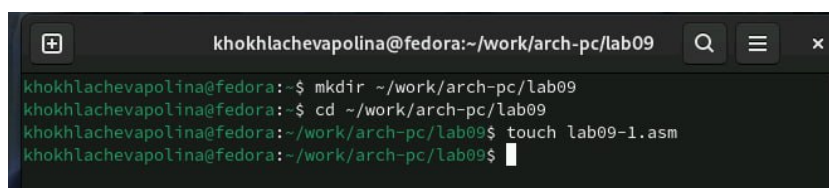
4.5 Проверяем работу	20
--------------------------------	----

1 Цель работы

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

2 Выполнение лабораторной работы

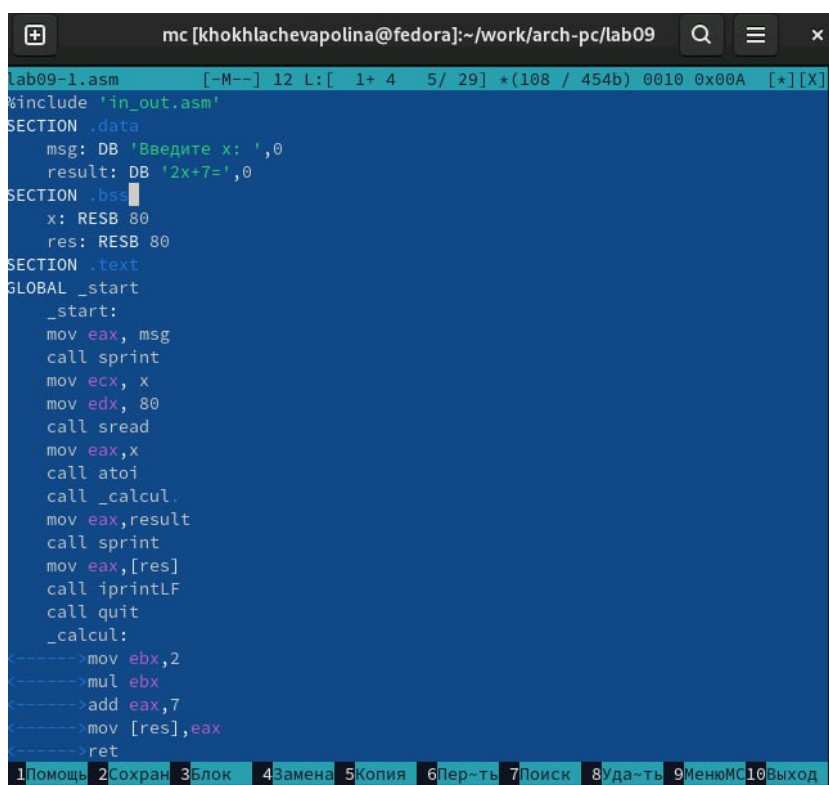
Создаём новый каталог для лабораторной №9



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

Рис. 2.1: И в нём новый файл

Открываем файл и заполняем его в соответствии с листингом



```
lab09-1.asm [-M--] 12 L: [ 1+ 4 5/ 29] *(108 / 454b) 0010 0x00A [*][X]
#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 sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    call _calcul
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF
    call quit
    _calcul:
    <-----> mov ebx, 2
    <-----> mul ebx
    <-----> add eax, 7
    <-----> mov [res], eax
    <-----> ret
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

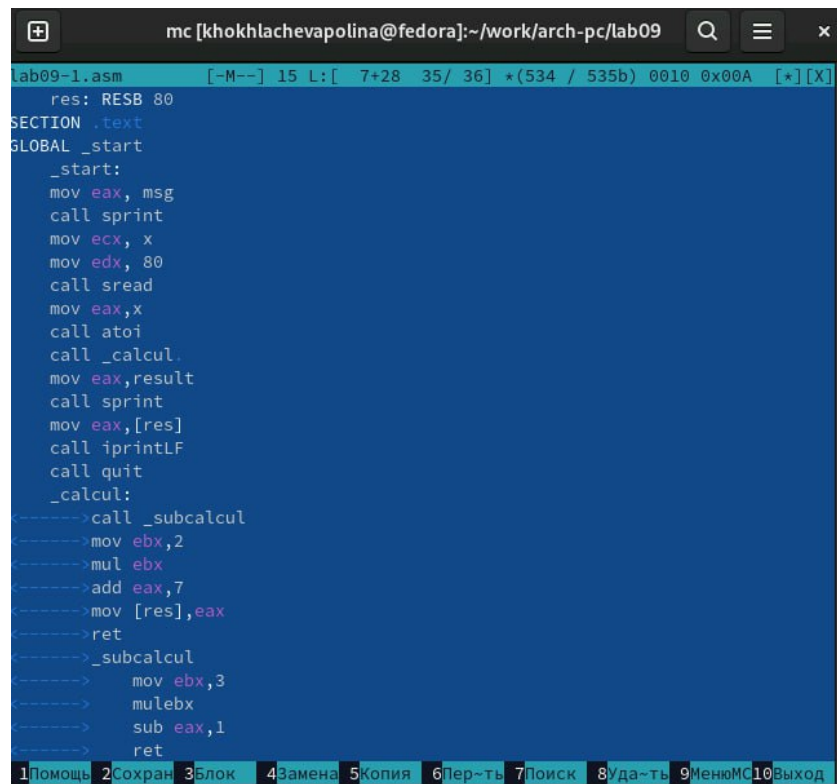
Рис. 2.2: Заполняем файл

Создаём файл и запускаем его

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

Рис. 2.3: Запускаем файл и проверяем его работу

Открываем файл для редактирования



```
lab09-1.asm [-M--] 15 L: [ 7+28 35/ 36] *(534 / 535b) 0010 0x00A [*] [X]
res: 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
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF
    call quit
    _calcul:
<-----> call _subcalcul
<-----> mov ebx, 2
<-----> mul ebx
<-----> add eax, 7
<-----> mov [res], eax
<-----> ret
<-----> _subcalcul
<-----> mov ebx, 3
<-----> mulebx
<-----> sub eax, 1
<-----> ret
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 2.4: Редактируем файл

Создаём файл и запускаем его

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=35
khokhlachevapolina@fedora:~/work/arch-pc/lab09$
```

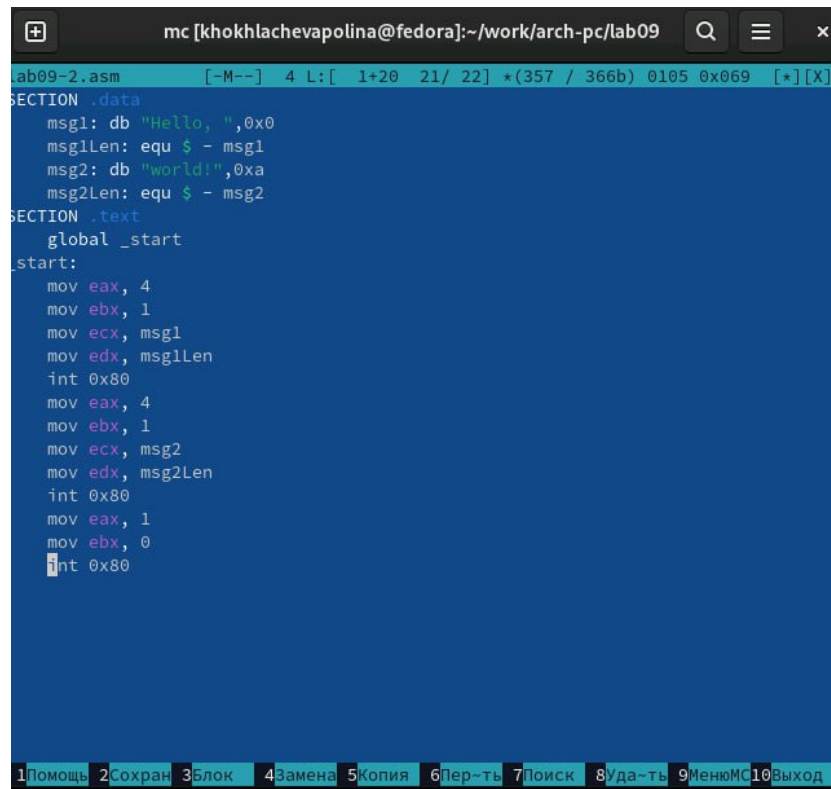
Рис. 2.5: Запускаем файл и смотрим на его работу

Создаём новый файл в каталоге

```
khokhlachevapolina@fedora: ~/work/arch-pc/lab09$ touch lab09-2.asm
khokhlachevapolina@fedora: ~/work/arch-pc/lab09$
```

Рис. 2.6: Создаём файл

Открываем файл и заполняем его в соответствии с листингом



```
lab09-2.asm [-M--] 4 L: [ 1+20 21/ 22] *(357 / 366b) 0105 0x069 [*][X]
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. 2.7: Заполняем файл

Получаем исходный файл с использованием отладчика gdb


```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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...
(gdb)
```

Рис. 2.8: Загружаем исходный файл

Запускаем команду в отладчике

```
(gdb) run
Starting program: /home/khokhlachevapolina/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 8447) exited normally]
(gdb)
```

Рис. 2.9: Запускаем программу командой run

Устанавливаем брейкпоинт на метку _start и запускаем программу

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/khokhlachevapolina/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 2.10: Запускаем программу с брейкпоинтом

Смотрим кол программы

```

(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. 2.11: Смотрим код программы

Переключаемся на отображение команд с Intelовским синтаксисом

```

(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. 2.12: Переключаемся на синтаксис Inter

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel:

- 1.Порядок операндов: В АТТ синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel синтаксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.
- 2.Разделители: В АТТ синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).
- 3.Префиксы размера операндов: В АТТ синтаксисе размер операнда указывается перед операндом с использованием префиксов, таких как “b” (byte), “w” (word), “l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.
- 4.Знак операндов: В АТТ синтаксисе операнды с позитивными значениями предваряются символом “+”.
- 5.Обозначение адресов: В АТТ синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.
- 6.Обозначение регистров: В АТТ синтаксисе обозначение регистра начинается с символа “%”. В Intel синтаксисе обозначение регистра может начинаться с символа “R” или “E” (например, “%eax” или “RAX”).

Включаем режим псевдографики

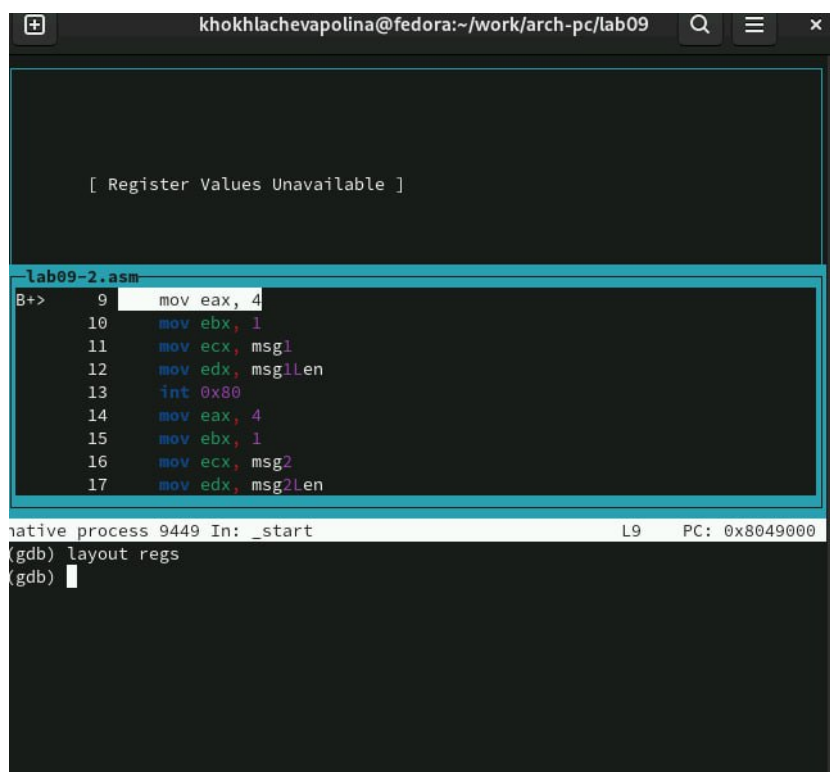


Рис. 2.13: Включаем отображение регистров, их значений и результат дисассимилирования программы

Проверяем установку точки основы и устанавливаем точку останова предпоследней инструкции

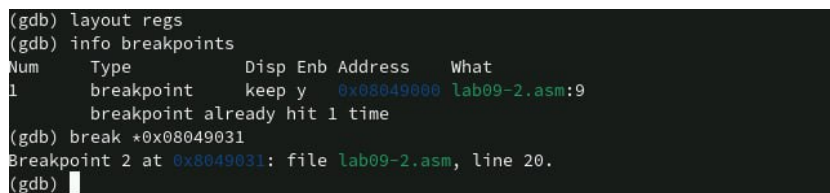


Рис. 2.14: Используем команду и создаём новую точку останова

Посмотрим информацию о всех установленных точках останова

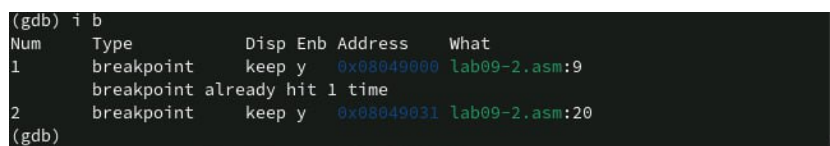
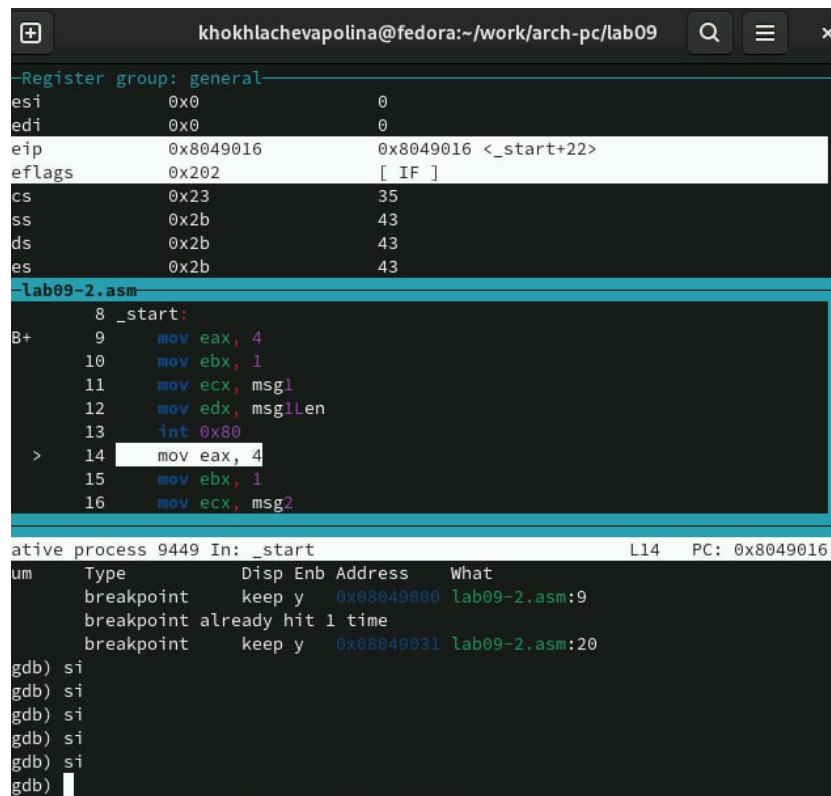


Рис. 2.15: Смотрим информацию

Выполняем 5 инструкций командой si



The screenshot shows a GDB window with the title bar 'khokhlachevapolina@fedora:~/work/arch-pc/lab09'. The window is divided into three main sections. The top section, titled 'Register group: general', displays the following register values: esi (0x0), edi (0x0), eip (0x8049016, pointing to <_start+22>), eflags (0x202, with [IF] flags), cs (0x23, 35), ss (0x2b, 43), ds (0x2b, 43), and es (0x2b, 43). The middle section, titled 'lab09-2.asm', shows assembly code with line numbers 8 through 16. Line 14, 'mov eax, 4', is highlighted with a mouse cursor. The bottom section shows the active process '9449 In: _start' with a list of breakpoints. The first breakpoint is at address 0x8049000 (lab09-2.asm:9) and is marked as 'already hit 1 time'. The second breakpoint is at address 0x8049031 (lab09-2.asm:20). The GDB prompt shows a series of 'si' commands being entered.

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09
--Register group: general--
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--lab09-2.asm
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

active process 9449 In: _start L14 PC: 0x8049016
um      Type      Disp Enb Address  What
breakpoint keep y   0x8049000 lab09-2.asm:9
breakpoint already hit 1 time
breakpoint keep y   0x8049031 lab09-2.asm:20
gdb) si
gdb) si
gdb) si
gdb) si
gdb) si
gdb) 
```

Рис. 2.16: Отслеживаем регистры

Во время выполнения команд менялись регистры: ebx, ecx, edx, eax, eip.

Смотрим значение переменной msg1 по имени

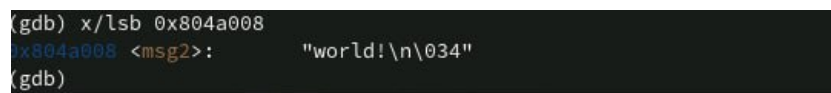


The screenshot shows a GDB session where the command 'x/lsb &msg1' has been entered. The output shows the memory address 0x804a000 containing the string 'Hello, '.

```
(gdb) x/lsb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) 
```

Рис. 2.17: Смотрим значение переменной

Смотрим значение переменной



The screenshot shows a GDB session where the command 'x/lsb 0x804a008' has been entered. The output shows the memory address 0x804a008 containing the string 'world!\n\034'.

```
(gdb) x/lsb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) 
```

Рис. 2.18: Смотрим значение переменной

Изменим первый символ

```
(gdb) set {char}&msg1='h'
(gdb) x/lsd &msg1
0x804a000 <msg1>:      104
(gdb) █
```

Рис. 2.19: Меняем символ

Изменим первый символ другой переменной

```
(gdb) set {char}&msg2='L'
(gdb) x/lsd &msg2
0x804a008 <msg2>:      76
(gdb) █
```

Рис. 2.20: Меняем символ

Смотрим значение в разных форматах

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

Рис. 2.21: Значение регистра

Изменяем регистр ebx

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

Рис. 2.22: Изменяем регистр

Выводятся разные значения, так как команда без кавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы

```
(gdb) c
Continuing.
World!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) █
```

Рис. 2.23: Прописываем команды

Копируем файл в файл

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$
```

Рис. 2.24: Копируем

Создаём исполняемый файл и запускаем его

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '5'
```

Рис. 2.25: Создаём и запускаем

Установим точку останова перед первой инструкцией в программе и запустим её

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/khokhlachevapolina/work/arch-pc/lab09/lab09-3 2 3 5

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 lab09-3.asm:5
5      pop ecx
(gdb) x/x $esp
0xffffd030: 0x00000004
(gdb)
```

Рис. 2.26: Устанавливаем точку останова

Смотрим позиции стека по разным адресам

```
(gdb) x/s *(void**)(esp + 4)
0xffffd1f7:  "/home/khokhlachevapolina/work/arch-pc/lab09/lab09-3"
(gdb) ) x/s *(void**)(esp + 8)
Undefined command: "). Try "help".
(gdb) x/s *(void**)(esp + 12)
0xffffd22d:  "3"
(gdb) x/s *(void**)(esp + 16)
0xffffd22f:  "5"
(gdb) x/s *(void**)(esp + 20)
0x0:  <error: Cannot access memory at address 0x0>
```

Рис. 2.27: Изучаем полученные данные

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

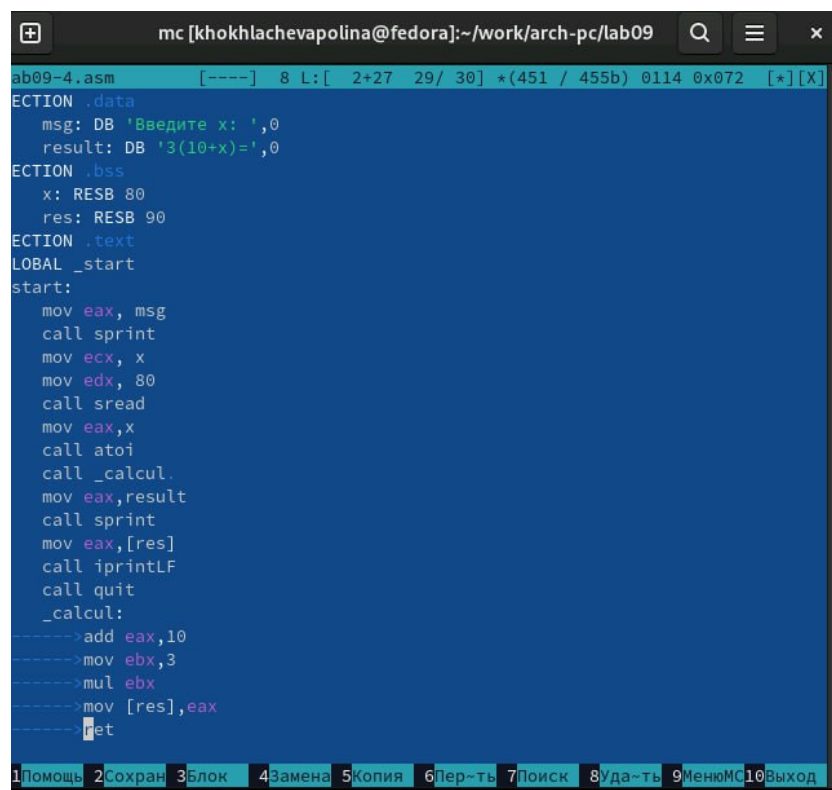
3 Задание 1

Копируем файл в файл

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab09-4.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$
```

Рис. 3.1: Копируем файл

Открываем файл и меняем его



```
mc [khokhlachevapolina@fedora]:~/work/arch-pc/lab09 Q ≡ x
ab09-4.asm [----] 8 L: [ 2+27 29/ 30] *(451 / 455b) 0114 0x072 [*][X]
SECTION .data
    msg: DB 'Введите x: ',0
    result: DB '3(10+x)=',0
SECTION .bss
    x: RESB 80
    res: RESB 90
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
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF
    call quit
    _calcul:
    -----> add eax, 10
    -----> mov ebx, 3
    -----> mul ebx
    -----> mov [res], eax
    -----> ret
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход
```

Рис. 3.2: Изменяем файл

Создаём Исполняемый файл и запускаем его

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ./lab09-4
Введите x: 5
3(10+x)=45
khokhlachevapolina@fedora:~/work/arch-pc/lab09$
```

Рис. 3.3: Проверяем работу

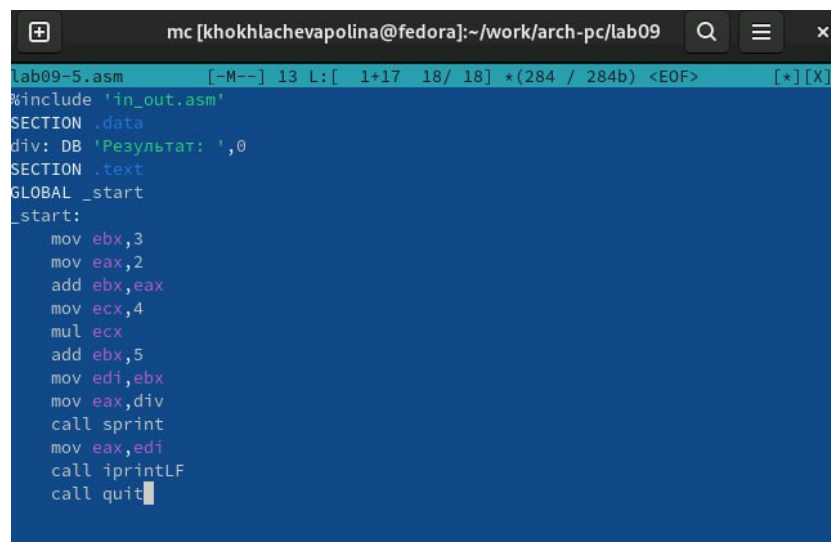
4 Задание 2

Создаём новый файл в дирректории

```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ touch lab09-5.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$
```

Рис. 4.1: Создаём файл

Открываем файл и заполняем его в соответствии с листингом



```
mc [khokhlachevapolina@fedora]:~/work/arch-pc/lab09
lab09-5.asm [-M--] 13 L: [ 1+17 18/ 18] *(284 / 284b) <EOF> [*][X]
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
    mov ebx,3
    mov eax,2
    add ebx,eax
    mov ecx,4
    mul ecx
    add ebx,5
    mov edi,ebx
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

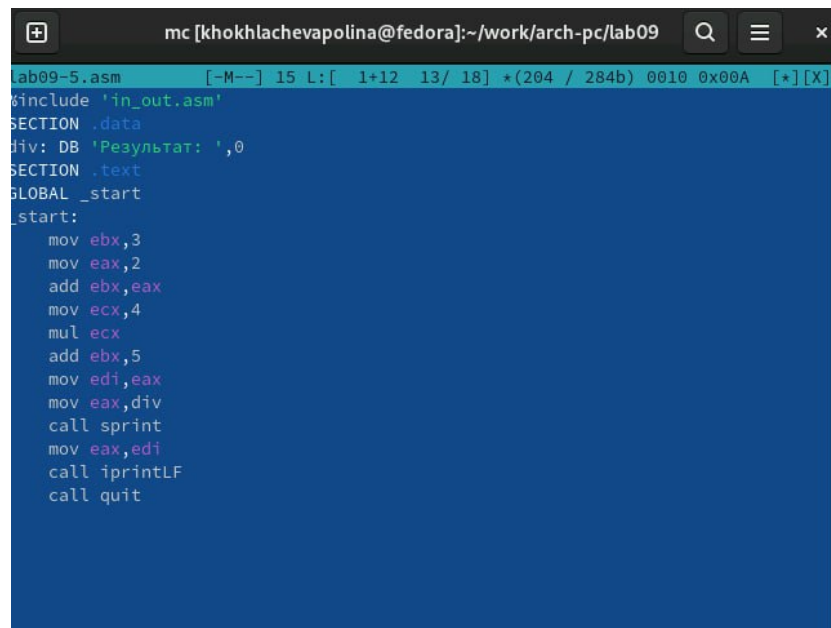
Рис. 4.2: Изменяем файл

Создаём исполняемый файл и запускаем его

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

Рис. 4.3: Смотрим на работу программы

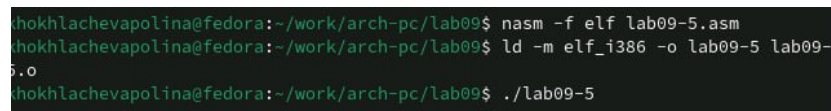
Создаём файл и смотрим на изменение регистров, ищем ошибку
Изменяем программу



```
lab09-5.asm [-M--] 15 L: [ 1+12 13/ 18] *(204 / 284b) 0010 0x00A [*] [X]
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
    mov ebx,3
    mov eax,2
    add ebx,eax
    mov ecx,4
    mul ecx
    add ebx,5
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис. 4.4: Меняем файл

Создаём файл и запускаем его



```
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
khokhlachevapolina@fedora:~/work/arch-pc/lab09$ ./lab09-5
```

Рис. 4.5: Проверяем работу

5 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.