

Отчёт по лабораторной работе №9

Понятие подпрограммы.Отладчик GBD

Майоров Дмитрий Андреевич

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
3	Задание для самостоятельной работы	21
4	Выводы	26
	Список литературы	27

Список иллюстраций

2.1	Создаем каталог для програм лабораторной работы и файл в нем .	7
2.2	Открываем файл и заполняем его в соответствии с листингом . . .	8
2.3	Создаем исполняемый файл и запускаем его	9
2.4	Изменяем файл. Добавляем подпрограмму	10
2.5	Создаем исполняемый файл и запускаем его	10
2.6	Создаем новый файл	11
2.7	Открываем файл и заполняем его в соответствии с листингом . . .	11
2.8	Загружаем исходный файл в отладчик	12
2.9	Запускаем программу командой run	12
2.10	Запускаем программу с брейкпоинтом	12
2.11	Смотрим дисассимилированный код программы	13
2.12	Переключаемся на синтаксис intel	13
2.13	Включаем отображение регистров, их значений и результат дисассимилирования программы	15
2.14	Создаем новую точку останова	16
2.15	Смотрим информацию	16
2.16	Отслеживаем регистры командой si	17
2.17	Смотрим значение переменной msg1 по имени	18
2.18	Смотрим значение переменной msg2 по адресу	18
2.19	Изменим первый символ переменной msg1	18
2.20	Изменим первый символ переменной msg2	18
2.21	Смотрим значение регистра edx в разных форматах	18
2.22	Изменяем регистр ebx	18
2.23	Прописываем команды для завершения программы и выхода из GDB	19
2.24	Копируем файл lab8-2.asm в файл с именем lab09-3.asm	19
2.25	Создаем исполняемый файл и запускаем его в отладчике GDB . . .	19
2.26	Устанавливаем точку останова	19
2.27	Изучаем полученные данные	20
3.1	Копируем файл lab8-4.asm в файл с именем lab09-4.asm	21
3.2	Изменяем файл. Создаем подпрограмму	22
3.3	Создаем исполняемый файл и запускаем его	22
3.4	Создаем файл	22
3.5	Открываем файл и заполняем его в соответствии с листингом . . .	23
3.6	Создаем исполняемый файл и запускаем его. Программа работает неправильно	23
3.7	Ищем ошибку регистров в отладчике	24

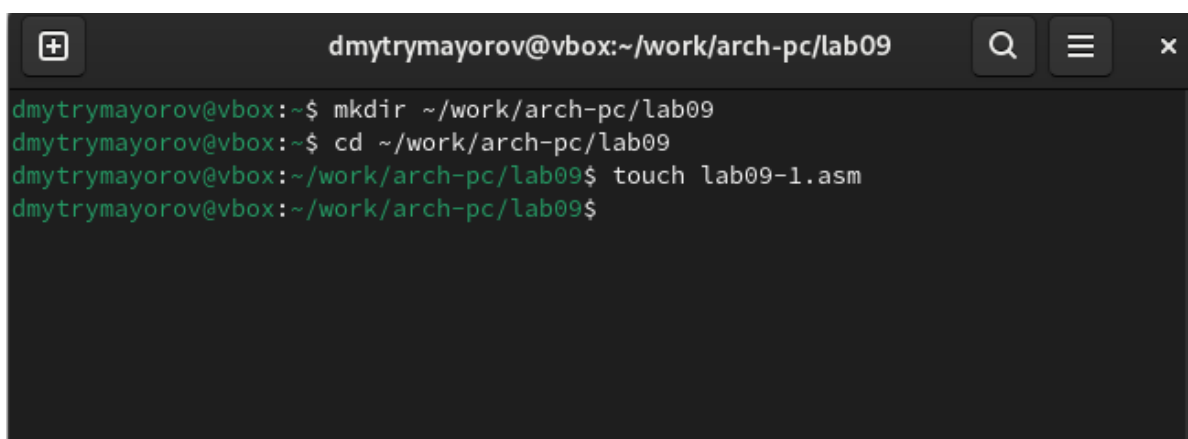
3.8	Изменяем файл	25
3.9	Создаем исполняемый файл и запускаем его. Программа работает правильно	25

Список таблиц

1 Цель работы

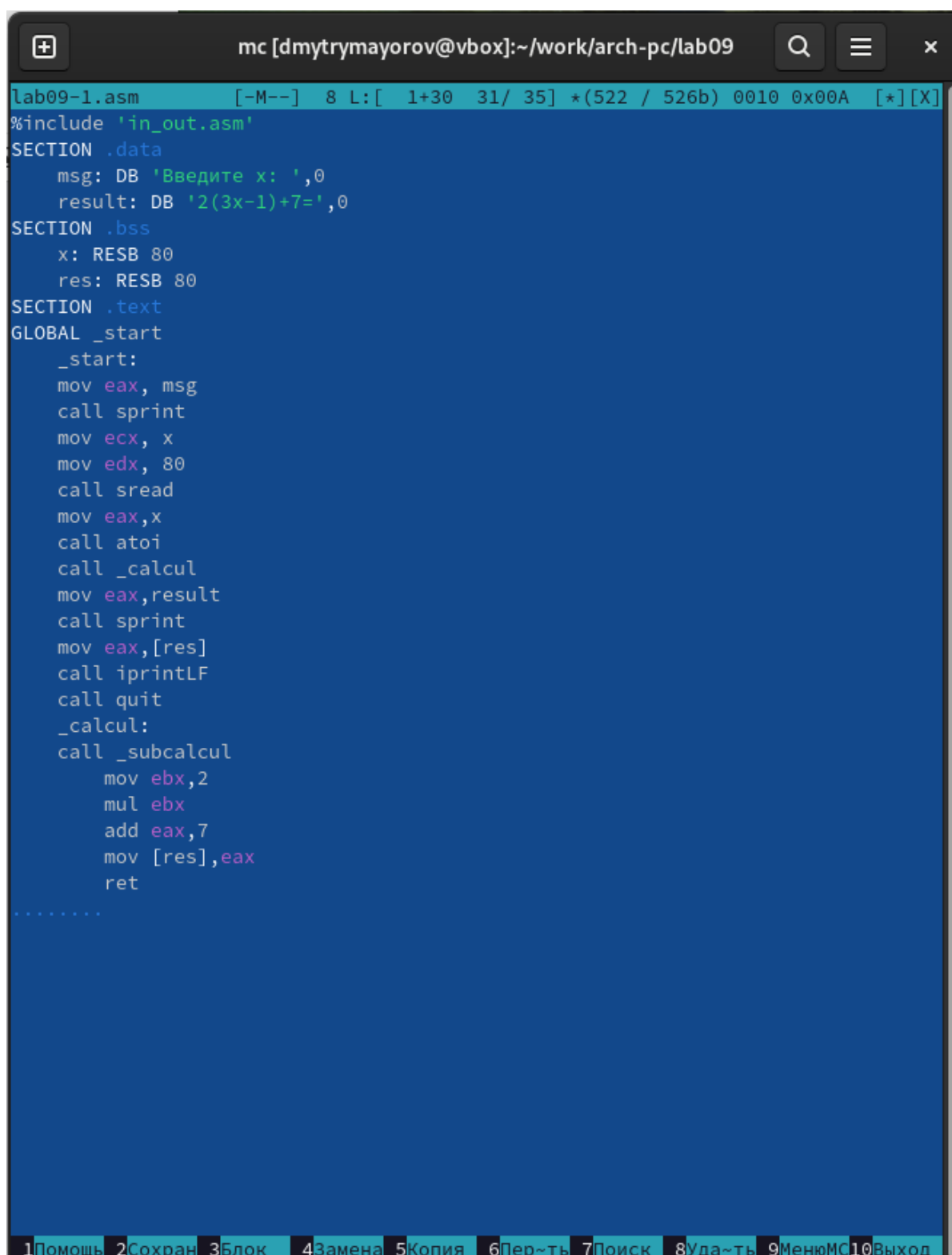
Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы



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

Рис. 2.1: Создаем каталог для програм лабораторной работы и файл в нем

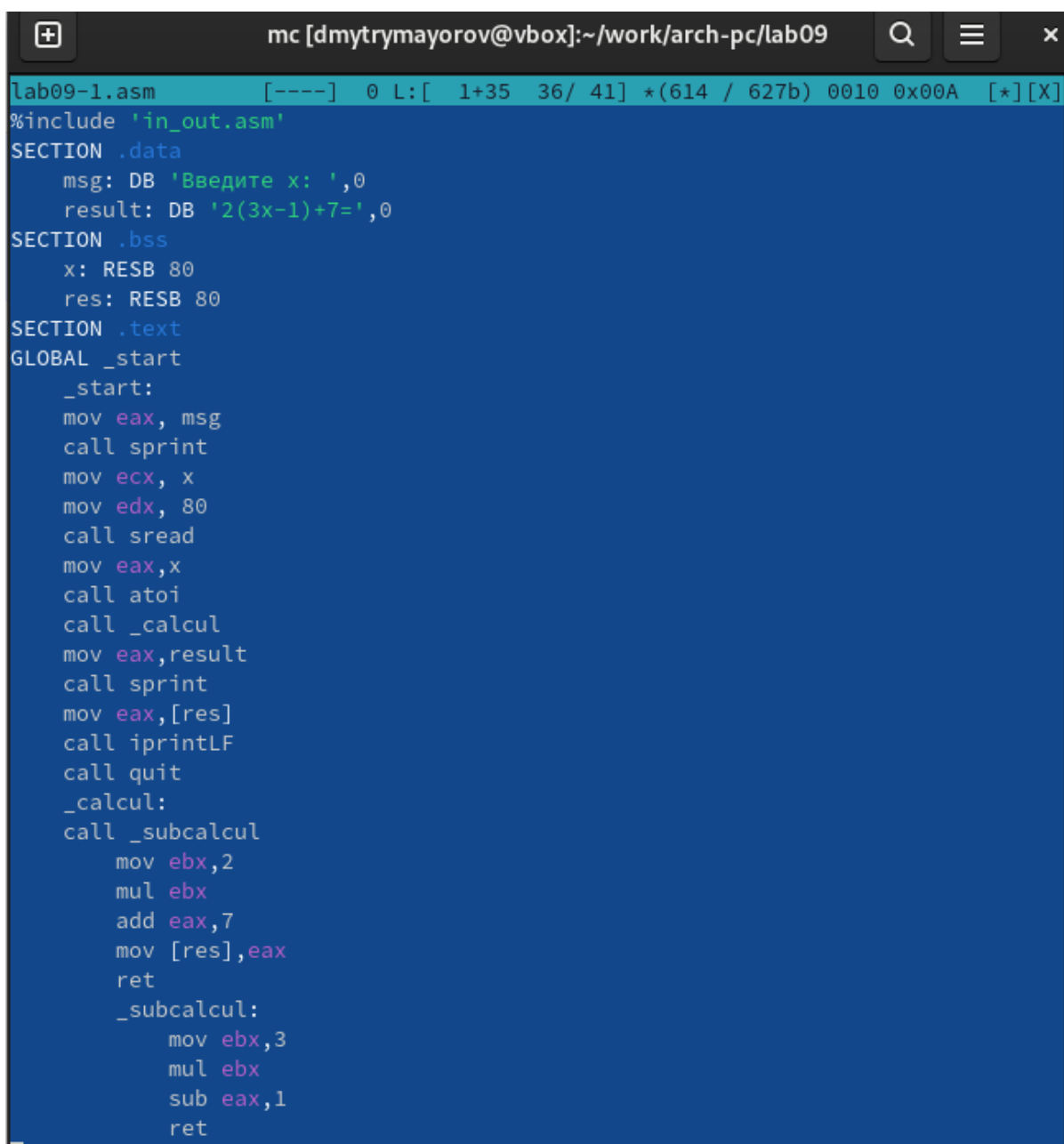


```
lab09-1.asm [-M--] 8 L: [ 1+30 31/ 35] *(522 / 526b) 0010 0x00A [*] [X]
#include 'in_out.asm'
SECTION .data
    msg: DB 'Введите x: ',0
    result: DB '2(3x-1)+7=',0
SECTION .bss
    x: RESB 80
    res: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprint
    mov ecx, x
    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
.....
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 2.2: Открываем файл и заполняем его в соответствии с листингом

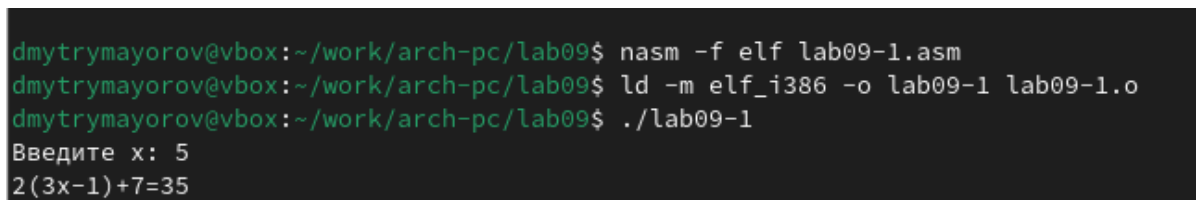

```
dmytrymayorov@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
 $2(3x-1)+7=17$ 
dmytrymayorov@vbox:~/work/arch-pc/lab09$
```

Рис. 2.3: Создаем исполняемый файл и запускаем его



```
lab09-1.asm [----] 0 L: [ 1+35 36/ 41] *(614 / 627b) 0010 0x00A [*][X]
#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:
    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
    mul ebx
    sub eax, 1
    ret
```

Рис. 2.4: Изменяем файл. Добавляем подпрограмму

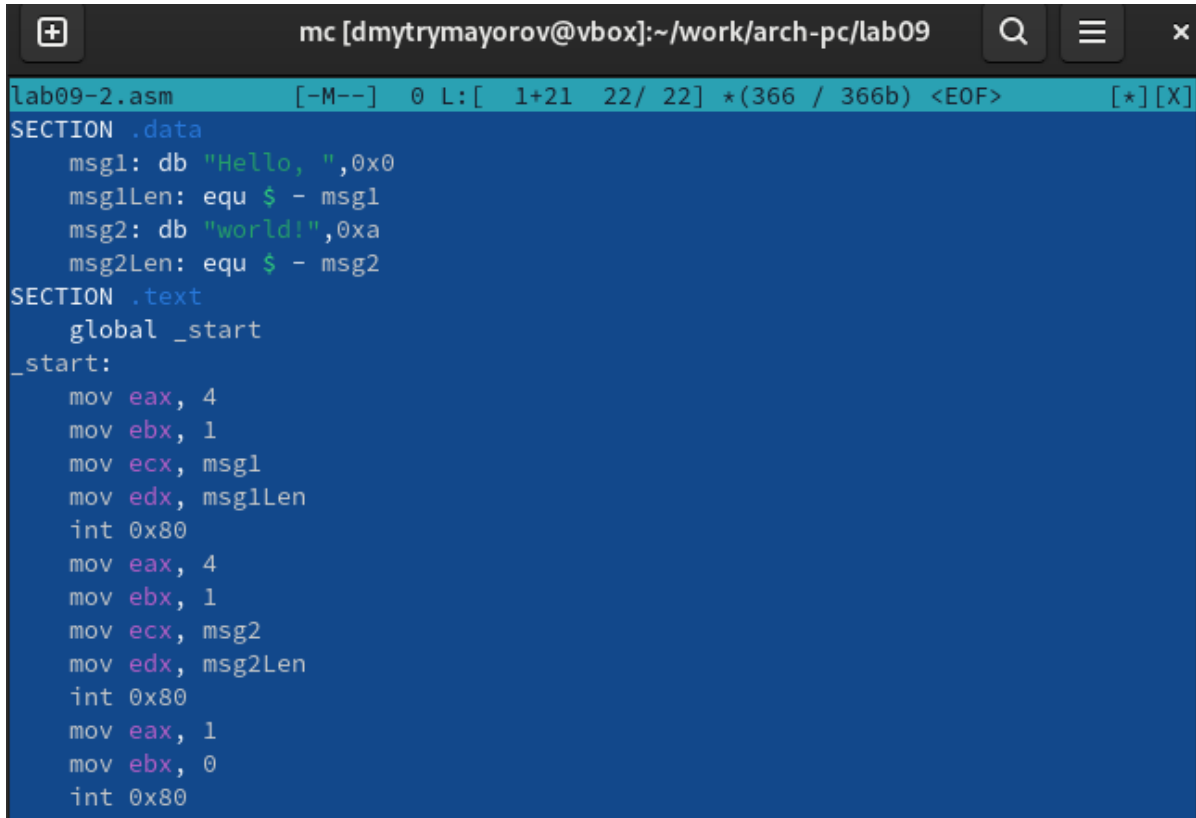


```
dmytrymayorov@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2(3x-1)+7=35
```

Рис. 2.5: Создаем исполняемый файл и запускаем его

```
dmytrymayorov@vbox:~/work/arch-pc/lab09$ touch lab09-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$
```

Рис. 2.6: Создаем новый файл



```
mc [dmytrymayorov@vbox]:~/work/arch-pc/lab09
lab09-2.asm [-M--] 0 L: [ 1+21 22/ 22] *(366 / 366b) <EOF> [*] [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: Открываем файл и заполняем его в соответствии с листингом

```

dmytrymayorov@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
dmytrymayorov@vbox:~/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...
(no debugging symbols found in lab09-2)
(gdb)

```

Рис. 2.8: Загружаем исходный файл в отладчик

```

(gdb) run
Starting program: /home/dmytrymayorov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 5193) exited normally]
(gdb)

```

Рис. 2.9: Запускаем программу командой run

```

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

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

```

Рис. 2.10: Запускаем программу с брейкпоинтом

```

(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 дисассимилированный код программы

```

(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 intel

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”).

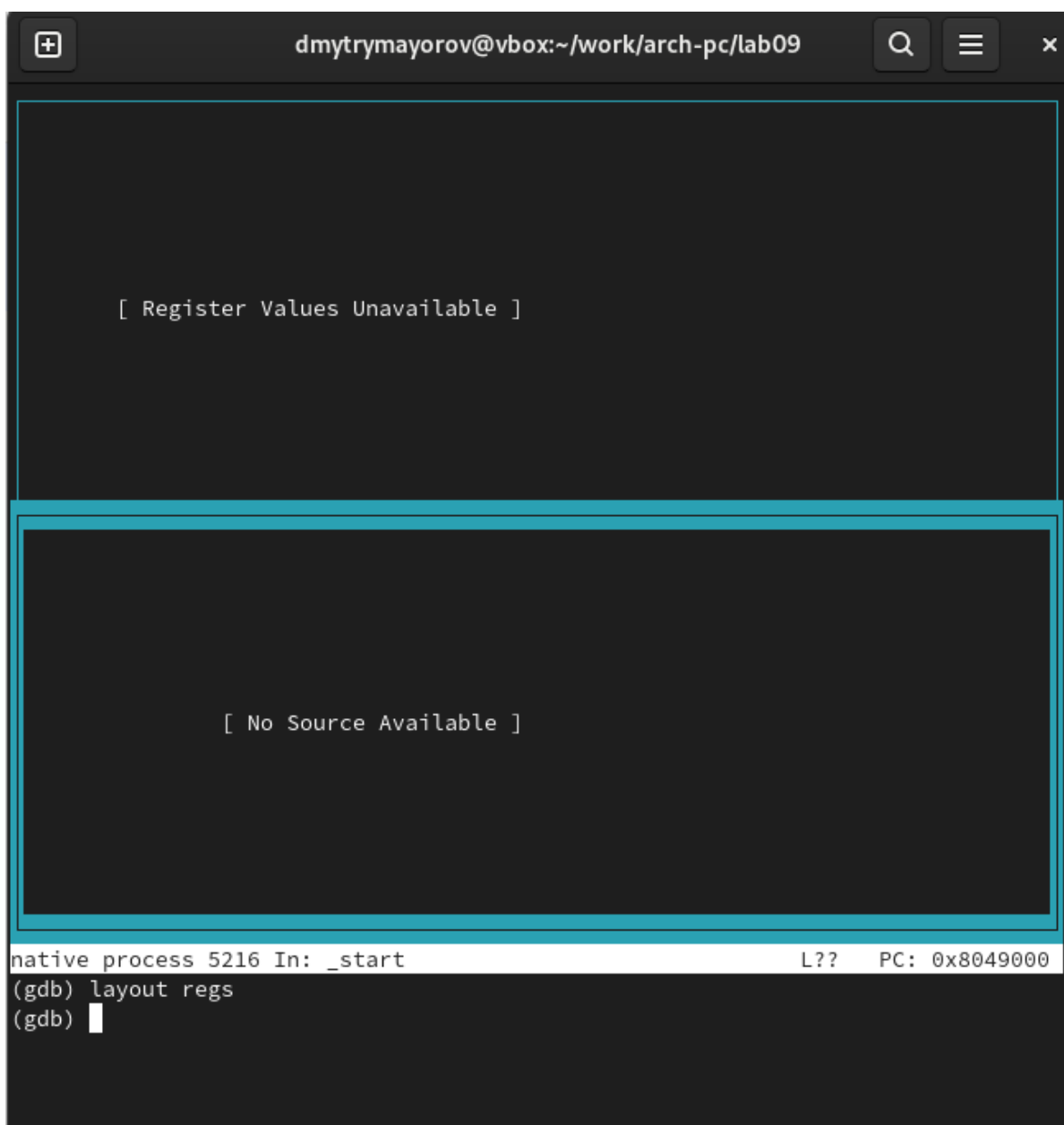


Рис. 2.13: Включаем отображение регистров, их значений и результат дисассемблирования программы

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

Рис. 2.14: Создаем новую точку останова

```
(gdb) i b
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. 2.15: Смотрим информацию


```
dmytrymayorov@vbox:~/work/arch-pc/lab09

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd080 0xffffd080
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       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 5410 In: _start L?? PC: 0x8049016

Breakpoint 1, 0x08049000 in _start ()
(gdb) layout asm
(gdb) layout regs
(gdb) si
0x08049005 in _start ()
(gdb) si
0x0804900a in _start ()
(gdb) si
0x0804900f in _start ()
(gdb) si
0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) 
```

Рис. 2.16: Отслеживаем регистры командой si

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

```
(gdb) x/1sb &msg1
0x804a000:      "Hello, "
(gdb)
```

Рис. 2.17: Смотрим значение переменной msg1 по имени

```
(gdb) x/1sb 0x804a008
0x804a008:      "world!\n"
(gdb)
```

Рис. 2.18: Смотрим значение переменной msg2 по адресу

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

Рис. 2.19: Изменим первый символ переменной msg1

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

Рис. 2.20: Изменим первый символ переменной msg2

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

Рис. 2.21: Смотрим значение регистра edx в разных форматах

```
(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 ebx

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

```
(gdb) c
Continuing.
Lor!d!

Breakpoint 2, 0x08049031 in _start ()
(gdb)
```

Рис. 2.23: Прописываем команды для завершения программы и выхода из GDB

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

Рис. 2.24: Копируем файл lab8-2.asm в файл с именем lab09-3.asm

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

Рис. 2.25: Создаем исполняемый файл и запускаем его в отладчике GDB

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/dmytrymayorov/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
0xffffd070:    0x00000004
(gdb)
```

Рис. 2.26: Устанавливаем точку останова

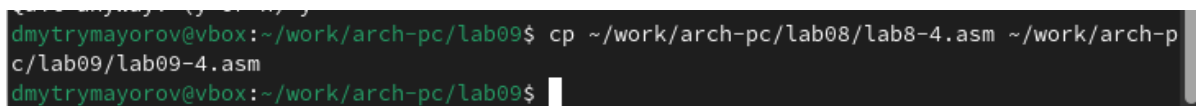
```
(gdb) x/s *(void**)(esp + 4)
0xffffd234:    "/home/dmytrymayorov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd263:    "2"
(gdb) x/s *(void**)(esp + 12)
0xffffd265:    "3"
(gdb) x/s *(void**)(esp + 16)
0xffffd267:    "5"
(gdb) x/s *(void**)(esp + 20)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.27: Изучаем полученные данные

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

3 Задание для самостоятельной работы

Задание 1

A terminal window with a dark background and green text. The prompt is 'dmytrymayorov@vbox: ~/work/arch-pc/lab09\$'. The command entered is 'cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab09-4.asm'. The prompt is repeated on the next line with a cursor at the end.

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

Рис. 3.1: Копируем файл lab8-4.asm в файл с именем lab09-4.asm

```

#include 'in_out.asm'
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

```

Рис. 3.2: Изменяем файл. СОздаем подпрограмму

```

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

```

Рис. 3.3: Создаем исполняемый файл и запускаем его

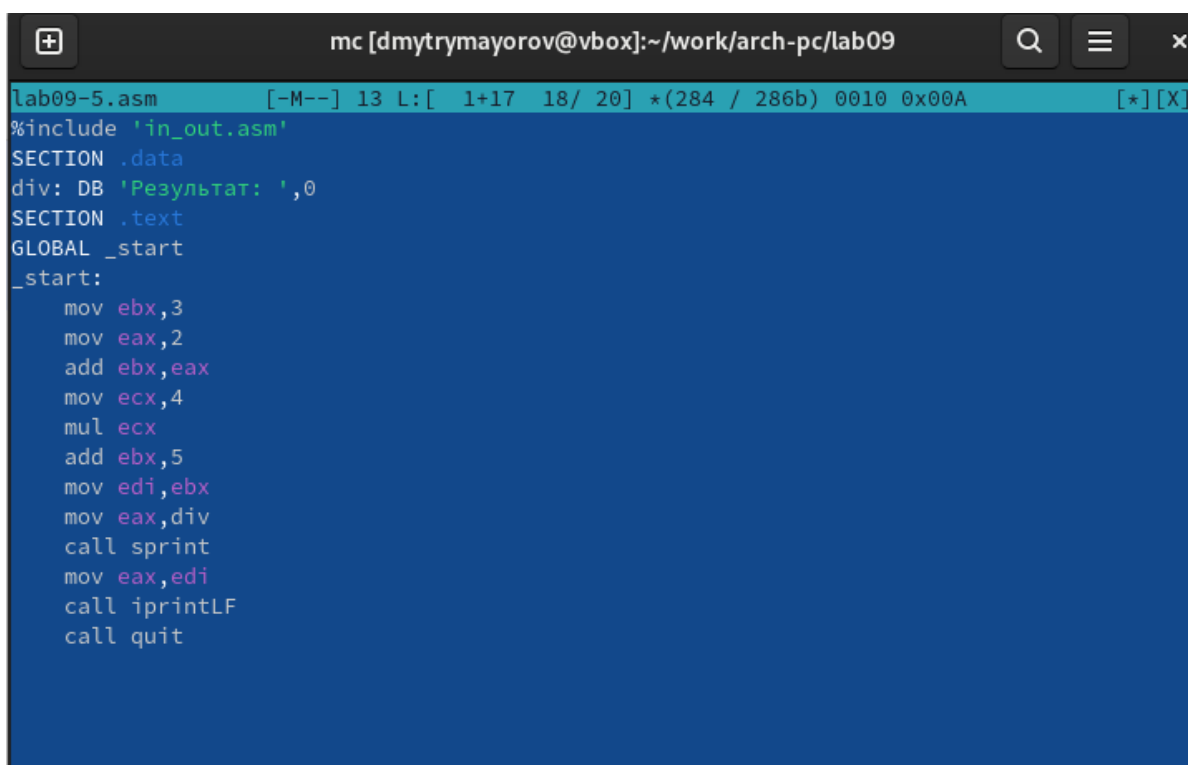
Задание 2

```

dmytrymayorov@vbox:~/work/arch-pc/lab09$ touch lab09-5.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$

```

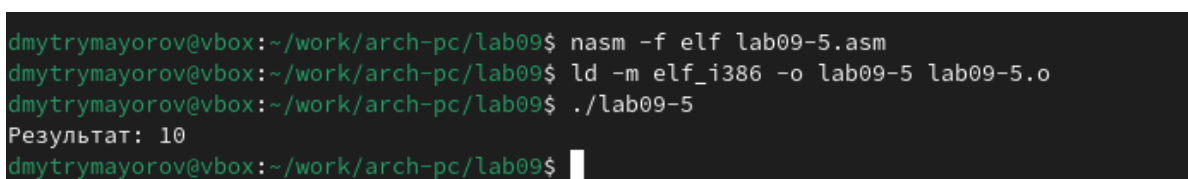
Рис. 3.4: Создаем файл



The screenshot shows a text editor window titled 'mc [dmytrymayorov@vbox]:~/work/arch-pc/lab09'. The file 'lab09-5.asm' is open, and its content is as follows:

```
lab09-5.asm      [-M--] 13 L: [ 1+17 18/ 20] *(284 / 286b) 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,ebx
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис. 3.5: Открываем файл и заполняем его в соответствии с листингом



The screenshot shows a terminal window with the following commands and output:

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

Рис. 3.6: Создаем исполняемый файл и запускаем его. Программа работает неправильно

```

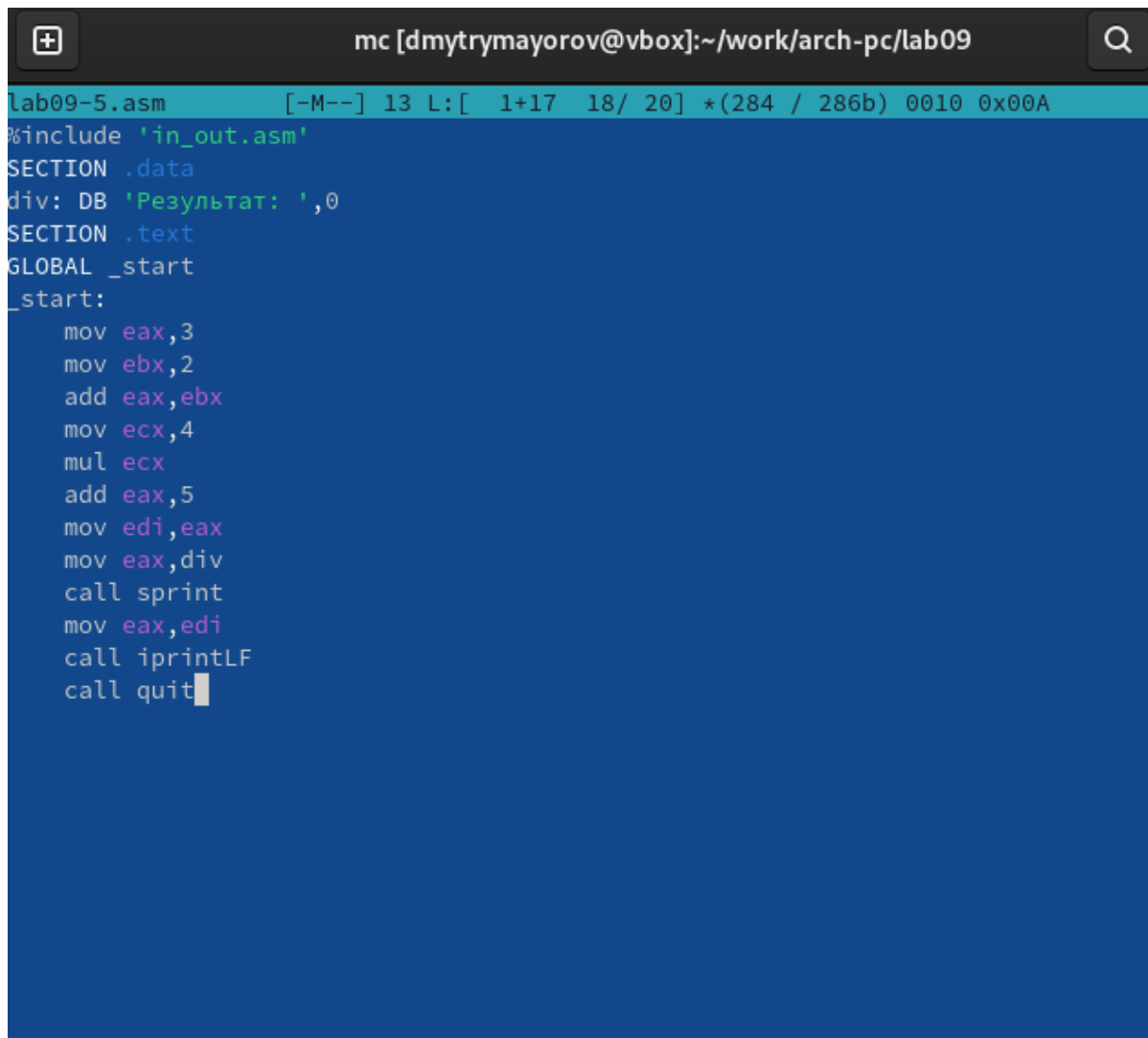
Register group: general
eax      0x2      2
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd0b0 0xffffd0b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f9 0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35
ss       0x2b     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
0x804910c <_start+36>   call   0x8049086 <iprintLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al

native process 8879 In: _start L11 PC: 0x80490f9
esp      0xffffd0b0 0xffffd0b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+12>
eflags   0x206    [ PF IF ]
cs       0x23     35

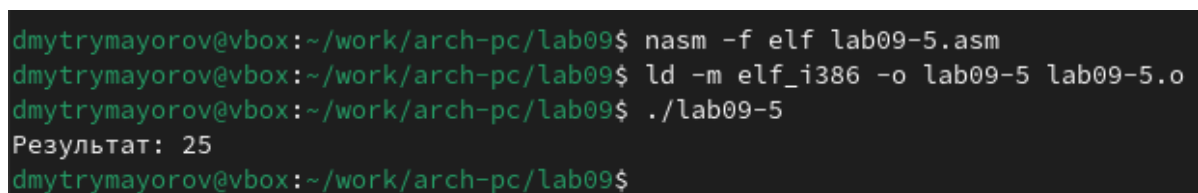
```

Рис. 3.7: Ищем ошибку регистров в отладчике



```
lab09-5.asm [-M--] 13 L:[ 1+17 18/ 20] *(284 / 286b) 0010 0x00A
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,3
    mov ebx,2
    add eax,ebx
    mov ecx,4
    mul ecx
    add eax,5
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис. 3.8: Изменяем файл



```
dmytrymayorov@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
dmytrymayorov@vbox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
dmytrymayorov@vbox:~/work/arch-pc/lab09$
```

Рис. 3.9: Создаем исполняемый файл и запускаем его. Программа работает правильно

4 Выводы

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

Список литературы