

Отчёт по лабораторной работе 9

Архитектура компьютера

Соловьев Серафим

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание для самостоятельной работы	21
3	Выводы	28

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	9
2.5	Код программы lab9-2.asm	10
2.6	Компиляция и запуск программы lab9-2.asm в отладчике	11
2.7	Дизассемблированный код	12
2.8	Дизассемблированный код в режиме интел	13
2.9	Точка остановки	14
2.10	Изменение регистров	15
2.11	Изменение регистров	16
2.12	Изменение значения переменной	17
2.13	Вывод значения регистра	18
2.14	Вывод значения регистра	19
2.15	Вывод значения регистра	21
2.16	Код программы lab9-4.asm	22
2.17	Компиляция и запуск программы lab9-4.asm	23
2.18	Код программы lab9-5.asm с ошибкой	24
2.19	Отладка	25
2.20	Код программы lab9-5.asm исправлен	26
2.21	Проверка работы	27

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

Я организовал папку для проведения лабораторного занятия № 9 и переместился в неё. После этого я создал файл с именем lab9-1.asm.

Давайте рассмотрим в качестве примера программу, задачей которой является расчёт арифметической формулы $f(x) = 2x + 7$, используя для этого вспомогательную функцию `calcul`. В этом случае значение x подаётся через клавиатуру, а расчёт формулы происходит внутри вспомогательной функции.

```
mc [sasoloviev@fedora-VirtualBox]:~/work/arch-pc/la
lab9-1.asm [----] 27 L: [ 1+29 30/ 30] *(462 / 462b) <EOF
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+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
mov eax, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Рис. 2.1: Код программы lab9-1.asm

```
[sasoloviev@fedora-VirtualBox lab09]$ nasm -f elf lab9-1.asm
[sasoloviev@fedora-VirtualBox lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[sasoloviev@fedora-VirtualBox lab09]$ ./lab9-1
Введите x: 3
2x+7=13
[sasoloviev@fedora-VirtualBox lab09]$
[sasoloviev@fedora-VirtualBox lab09]$
```

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

Затем я внес некоторые корректировки в код программы, включив дополнительную функцию `subcalcul` внутри `calcul` для расчёта формулы $f(g(x))$, при этом значение x по-прежнему вводится через клавиатуру, а функции $f(x) = 2x + 7$ и $g(x) = 3x - 1$ обрабатываются внутри этих функций.

```
mc [sasoloviev@fedora-VirtualBox]:~/work/ar
lab9-1.asm  [----]  8 L: [ 3+24 27/ 40] *(380 / 531b
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
mov eax, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 2.3: Код программы lab9-1.asm


```
[sasoloviev@fedora-VirtualBox lab09]$ nasm -f elf lab9-1.asm
[sasoloviev@fedora-VirtualBox lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[sasoloviev@fedora-VirtualBox lab09]$ ./lab9-1
Введите x: 2
2(3x-1)+7=17
[sasoloviev@fedora-VirtualBox lab09]$
```

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

Кроме того, я подготовил файл lab9-2.asm, содержащий код программы из Примера 9.2, который демонстрирует программу для вывода сообщения “Hello world!” на экран.



```
mc [sasoloviev@fedora-Virtual
lab9-2.asm [----] 10 L:[ 1+16 17/ 25]
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. 2.5: Код программы lab9-2.asm

Я добавил отладочную информацию с помощью ключа ‘-g’ для возможности работы с отладчиком GDB.

После этого я загрузил исполняемый файл в отладчик GDB и проверил функционирование программы, активировав её командой ‘run’ (или ‘r’).

```

[sasoloviev@fedora-VirtualBox lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[sasoloviev@fedora-VirtualBox lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[sasoloviev@fedora-VirtualBox lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora 12.1-2.fc36
Copyright (C) 2022 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/sasoloviev/work/arch-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])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 12790) exited normally]
(gdb)

```

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

Для тщательного анализа программы я установил точку останова на метке 'start', с которой стартует исполнение любой программы на ассемблере, и запустил программу для наблюдения. После этого я осмотрел дизассемблированный код программы, чтобы понять её структуру и работу.

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/sasoloviev/work/arch-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])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 12790) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 12.
(gdb) r
Starting program: /home/sasoloviev/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:12
12      mov eax, 4
(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. 2.7: Дизассемблированный код

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
12      mov eax, 4
(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) 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. 2.8: Дизассемблированный код в режиме интел

Чтобы проверить наличие брейкпоинта с меткой '_start', я применил команду 'info breakpoints' (или 'i b'). После этого я задал еще один брейкпоинт на адресе предпоследней команды 'mov ebx, 0x0'.

The screenshot shows a GDB terminal window with the title bar 'sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2'. The main window displays '[Register Values Unavailable]'. Below this, a list of assembly instructions is shown, with the first instruction highlighted in blue:

```
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
```

Below the assembly list, the status bar shows 'native process 12796 In: _start' and 'L12 PC: 0x8049000'. The command prompt shows the following commands and output:

```
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 23.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y   0x08049000 lab9-2.asm:12
       breakpoint already hit 1 time
2      breakpoint      keep y   0x08049031 lab9-2.asm:23
(gdb)
```

Рис. 2.9: Точка остановки

Используя отладчик GDB, я мог наблюдать и редактировать содержимое памяти и регистров. Я выполнил пять шагов командой 'stepi' (или 'si'), следя за изменениями в регистрах.

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>

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

native process 12796 In: _start L13 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--ss 0x2b
43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb)
```

Рис. 2.10: Изменение регистров

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 12796 In: _start L17 PC: 0x8049016
--Type <RET> for more, q to quit, c to continue without paging--ss 0x2b
43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) 
```

Рис. 2.11: Изменение регистров

Чтобы просмотреть значение переменной `msg1`, я воспользовался соответствующей командой для извлечения необходимой информации.

The screenshot shows a GDB window titled "sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The "Register group: general" section displays the following values:

Register	Value	Comment
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd200	0xffffd200
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

Below the registers, the assembly code is displayed, with the instruction at address 0x8049016 highlighted:

```
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
```

The status bar at the bottom indicates "native process 12796 In: _start" and "L17 PC: 0x8049016". The GDB command window shows the following commands and output:

```
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n\034"
(gdb)
```

Рис. 2.12: Изменение значения переменной

Я также использовал команду 'set' для модификации значений в регистрах или ячейках памяти, указывая при этом нужный регистр или адрес. Мне удалось изменить первый символ переменной msg1.

The screenshot shows a GDB terminal window with the title bar "sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into two main sections. The top section, titled "Register group: general", displays the current values of the general-purpose registers:

Register	Value	Comment
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd200	0xffffd200
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The bottom section displays a list of assembly instructions starting from address 0x8049000. The instruction at address 0x8049016, "mov eax, 0x4", is highlighted with a blue selection bar. Below the instruction list, the GDB prompt shows the current state: "native process 12796 In: _start L17 PC: 0x8049016". The register dump shows: \$2 = 1000, \$3 = 134520832, \$4 = 0x804a000, \$5 = 8, \$6 = 1000, \$7 = 0x8. The GDB prompt "(gdb)" is at the bottom.

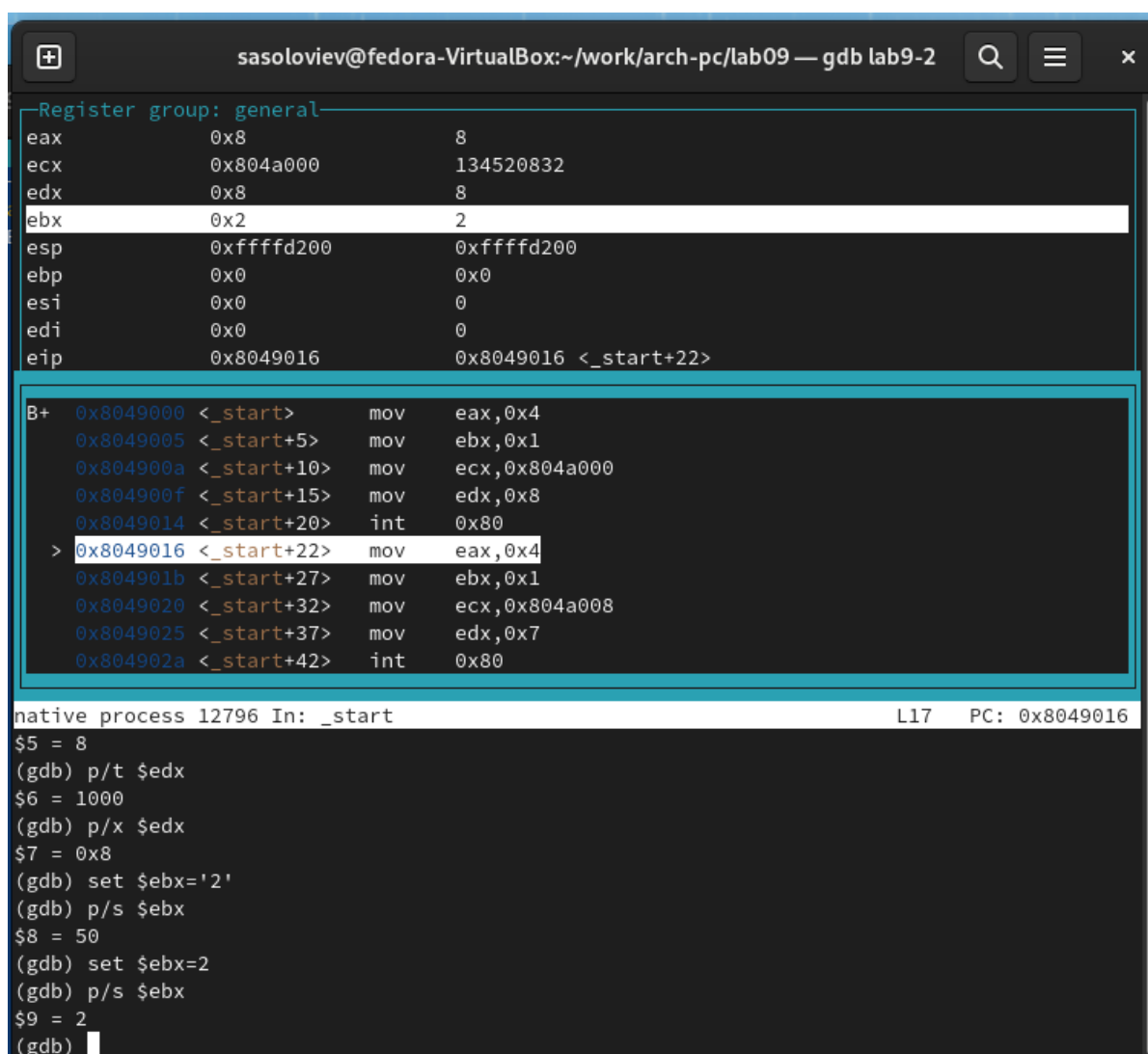
```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 12796 In: _start L17 PC: 0x8049016
$2 = 1000
(gdb) p/s $ecx
$3 = 134520832
(gdb) p/x $ecx
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Вывод значения регистра

С помощью команды 'set' я изменил значение регистра ebx на требуемое.



```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 12796 In: _start L17 PC: 0x8049016
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb)
```

Рис. 2.14: Вывод значения регистра

Я скопировал файл lab8-2.asm, созданный в рамках лабораторной работы №8, который содержит код программы для вывода аргументов командной строки, и сформировал из него исполняемый файл.

Для запуска программы с аргументами в GDB я использовал опцию `-args`, загрузив исполняемый файл с заданными аргументами в отладчик.

Я установил брейкпоинт перед выполнением первой команды программы и начал ее выполнение.

Адрес вершины стека, содержащий количество аргументов командной стро-

ки (включая название программы), находится в регистре ESP. По этому адресу расположено число, показывающее количество аргументов. В моем случае было видно, что их пять, включая название программы lab9-3 и аргументы: аргумент1, аргумент2 и 'аргумент 3'.

Я также осмотрел другие записи стека. По адресу [ESP+4] расположен указатель на имя программы в памяти. Адреса первого, второго и последующих аргументов находятся по адресам [ESP+8], [ESP+12] и так далее, с шагом в 4 байта, поскольку каждый следующий адрес отстоит на 4 байта от предыдущего ([ESP+4], [ESP+8], [ESP+12]).

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb --args lab9-3 arg...
(gdb) r
Starting program: /home/sasoloviev/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\
3

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1d0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd38d: "/home/sasoloviev/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3b8: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3c1: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3c3: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3cc: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3ce: "argument 3"
(gdb) c
Continuing.
argument
1
argument
2
argument 3
[Inferior 1 (process 12872) exited normally]
(gdb)
```

Рис. 2.15: Вывод значения регистра

2.1 Задание для самостоятельной работы

Модифицировал код из восьмой лабораторной работы (Первое задание для индивидуального выполнения), создав подпрограмму для расчета значения функции $f(x)$.

```
mc [sasoloviev@fedora-VirtualBox]:~/work/arch-pc/lab09
lab09-4.asm [----] 8 L: [ 1+20 21/ 38] *(247 / 379b) 0010 0x0
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 17 + 5x',0

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

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

loop next

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

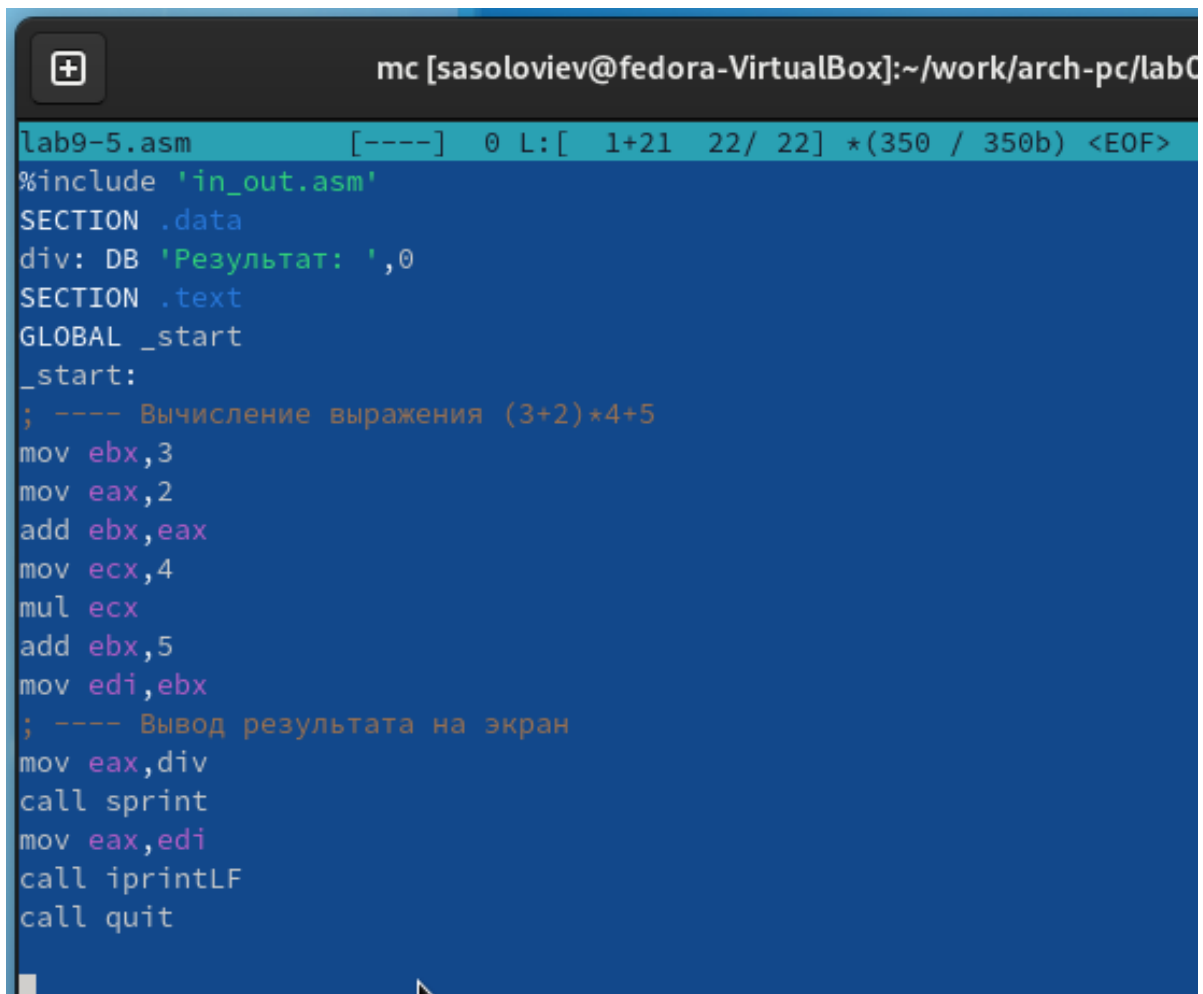
_fx:
mov ebx,5
mul ebx
add eax,17
ret
```

Рис. 2.16: Код программы lab9-4.asm

```
[sasoloviev@fedora-VirtualBox lab09]$  
[sasoloviev@fedora-VirtualBox lab09]$ nasm -f elf lab09-4.asm  
[sasoloviev@fedora-VirtualBox lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o  
[sasoloviev@fedora-VirtualBox lab09]$ ./lab09-4  
f(x)= 17 + 5x  
Результат: 0  
[sasoloviev@fedora-VirtualBox lab09]$ ./lab09-4 1  
f(x)= 17 + 5x  
Результат: 22  
[sasoloviev@fedora-VirtualBox lab09]$ ./lab09-4 1 3 6 4 7 9 1  
f(x)= 17 + 5x  
Результат: 274  
[sasoloviev@fedora-VirtualBox lab09]$
```

Рис. 2.17: Компиляция и запуск программы lab9-4.asm

В представленном коде описан алгоритм для расчета формулы $(3 + 2) * 4 + 5$. Однако его исполнение приводит к некорректному итогу. Я выявил это, наблюдая за изменениями в регистрах при помощи отладчика GDB.



```
mc [sasoloviev@fedora-VirtualBox]:~/work/arch-pc/lab0
lab9-5.asm [----] 0 L: [ 1+21 22/ 22] *(350 / 350b) <EOF>
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 2.18: Код программы lab9-5.asm с ошибкой


```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-5
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd200 0xffffd200
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>

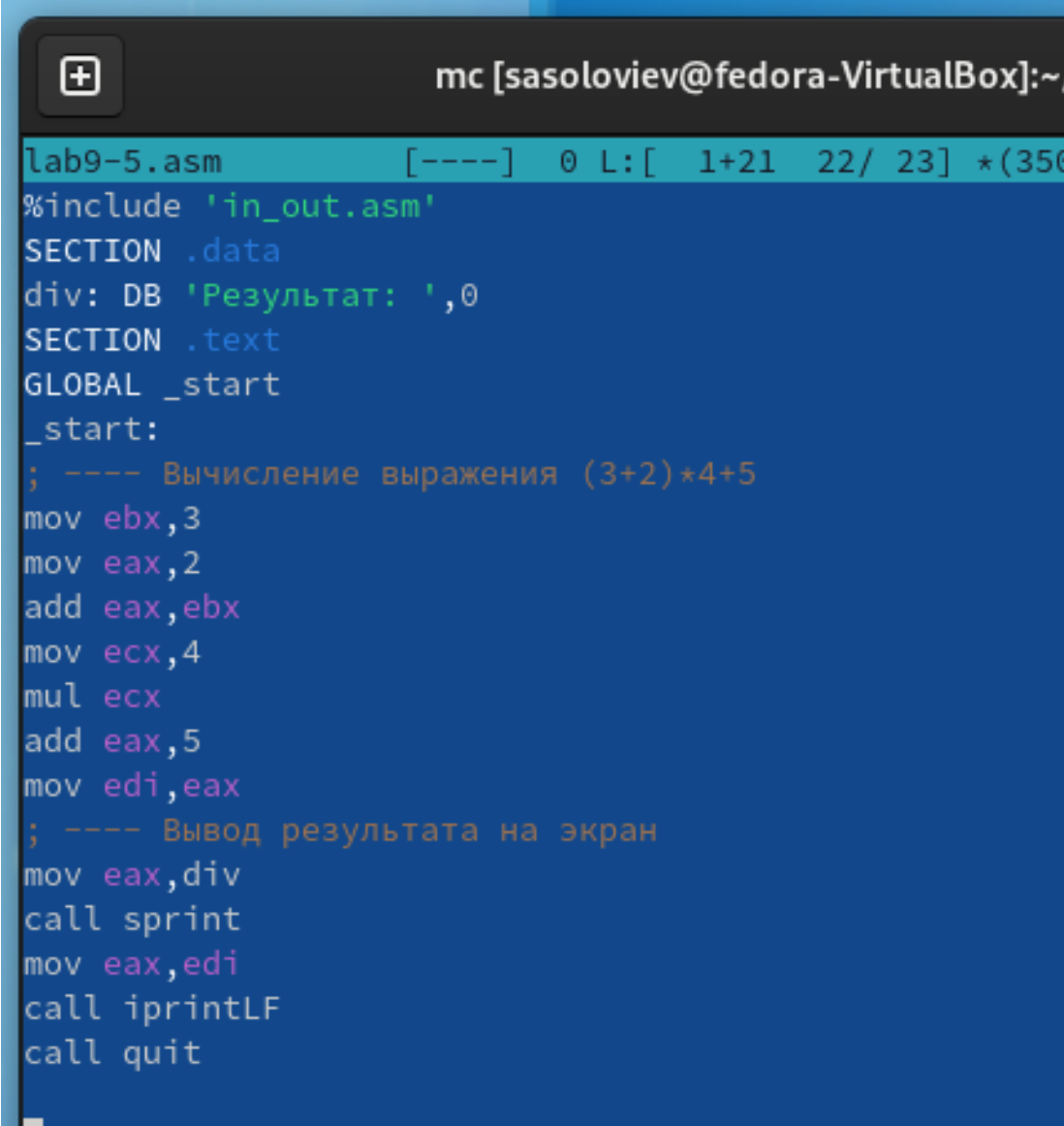
>                                04a000
                                rint>

native process 13093 In: _start L16 PC: 0x8049100
No process In: L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 13093) exited normally]
(gdb) 
```

Рис. 2.19: Отладка

Ошибка заключалась в неверном порядке аргументов команды add и в том, что в конце исполнения программы значение ebx переносится в edi вместо eax.

Исправленный код программы



```
mc [sasoloviev@fedora-VirtualBox]:~  
lab9-5.asm [----] 0 L: [ 1+21 22/ 23] *(356  
%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 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 программы lab9-5.asm исправлен

```
sasoloviev@fedora-VirtualBox:~/work/arch-pc/lab09 — gdb lab9-5
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd200 0xffffd200
ebp      0x0       0x0
esi      0x0       0
edi      0x19      25
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     ecx,0di
0x804910c <_start+36>  call    0x8049086 <iprintLF>
0x8049111 <_start+41>  call    0x80490db <quit>

>                                04a000
                                rint>

native process 13186 In: _start L16 PC: 0x8049100
No process In: L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 13186) exited normally]
(gdb) █
```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.