

# **Отчёт по лабораторной работе 9**

**Архитектура компьютера**

Саяпин Артём Максимович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	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 . . . . .	8
2.3	Код программы lab9-1.asm . . . . .	9
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 . . . . .	20
2.16	Код программы lab9-4.asm . . . . .	21
2.17	Компиляция и запуск программы lab9-4.asm . . . . .	22
2.18	Код программы lab9-5.asm с ошибкой . . . . .	23
2.19	Отладка . . . . .	24
2.20	Код программы lab9-5.asm исправлен . . . . .	25
2.21	Проверка работы . . . . .	26

## 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.

Открыть ▾ 

lab9-  
~/work/ari

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 rez: RESB 80
8
9 SECTION .text
10 GLOBAL _start
11 _start:|
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax,x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax,result
21 call sprint
22 mov eax,[rez]
23 call iprintLF
24 call quit
25 _calcul:
26 mov ebx,2
27 mul ebx
28 add eax,7
29 mov [rez],eax
30 ret ; выход из подпрограммы
```

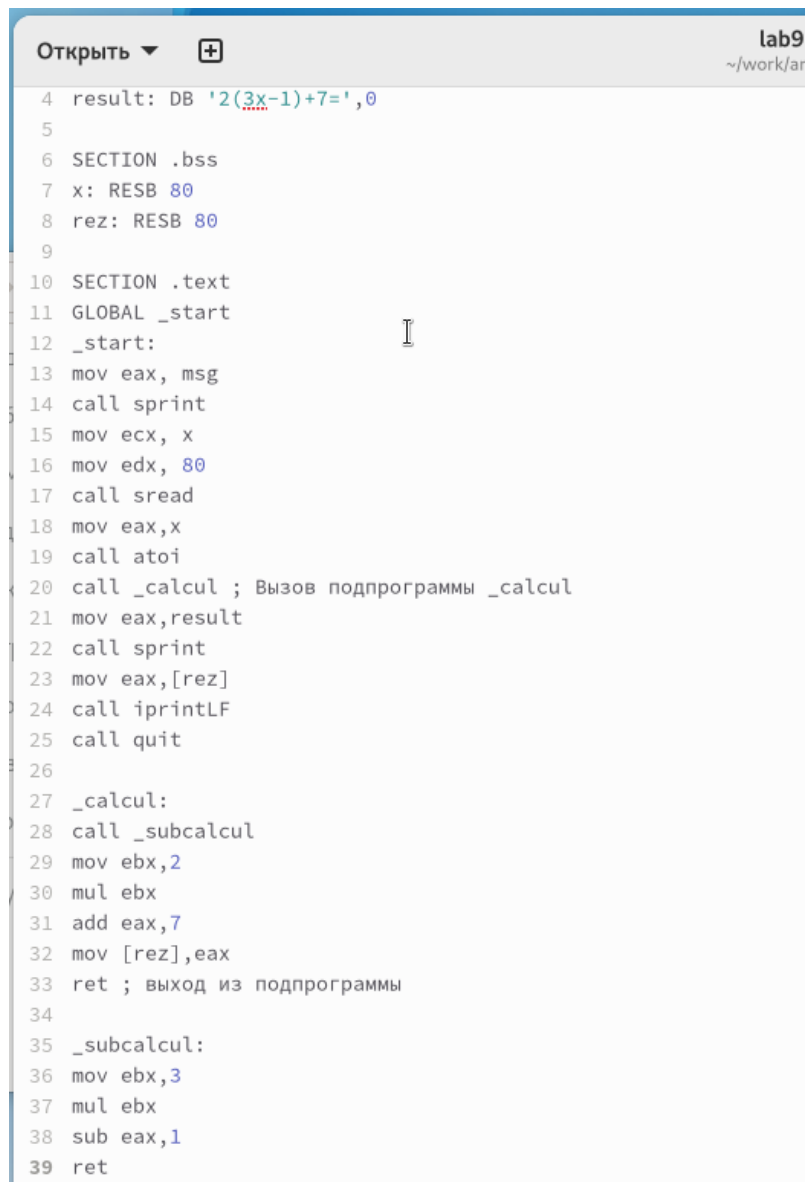
Рис. 2.1: Код программы lab9-1.asm

```
[amsayapin@VirtualBox lab09]$ nasm -f elf lab9-1.asm
[amsayapin@VirtualBox lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[amsayapin@VirtualBox lab09]$ ./lab9-1
Введите x: 2
2x+7=11
[amsayapin@VirtualBox lab09]$
```

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

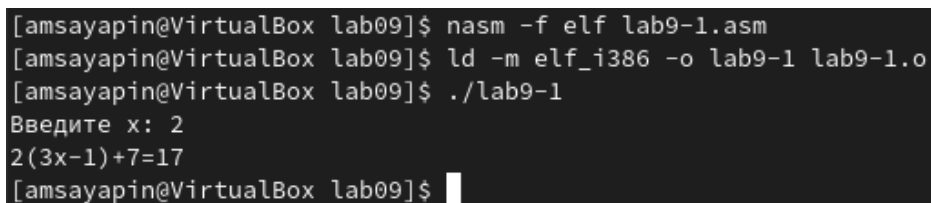
Затем я внес некоторые корректировки в код программы, включив дополнительную функцию `subcalcul` внутри `calcul` для расчёта формулы  $f(g(x))$ , при этом значение  $x$  по-прежнему вводится через клавиатуру, а функции  $f(x) = 2x + 7$  и  $g(x) = 3x - 1$  обрабатываются внутри этих функций.





```
Открыть ▾ + lab9
~/work/ar
4 result: DB '2(3x-1)+7=',0
5
6 SECTION .bss
7 x: RESB 80
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx, 2
30 mul ebx
31 add eax, 7
32 mov [rez], eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 ret
```

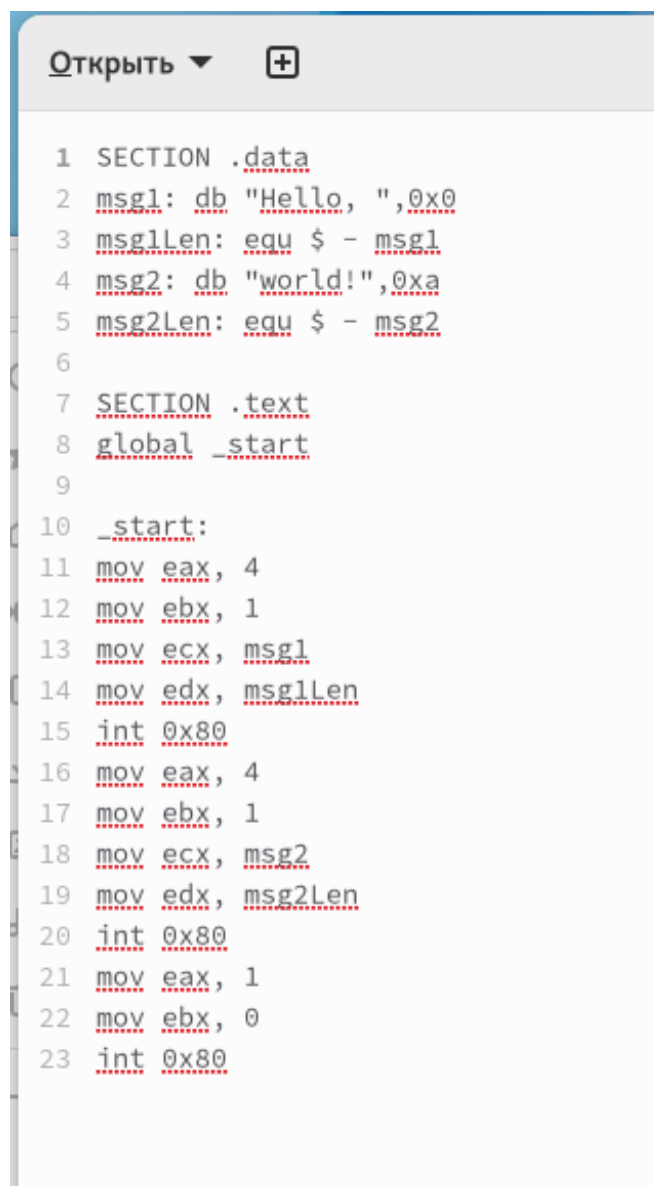
Рис. 2.3: Код программы lab9-1.asm



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



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
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
22 mov ebx, 0
23 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 функ-

ционирование программы, активировав её командой 'run' (или 'r').

```
[amsayapin@VirtualBox lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[amsayapin@VirtualBox lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[amsayapin@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/amsayapin/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 4440) 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.

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/amsayapin/work/arch-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:
=> 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

```
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 'amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2'. The main area displays '[ Register Values Unavailable ]'. A list of assembly instructions is shown, starting from address 0x8049000. Below this, the status bar indicates 'native process 4444 In: \_start' and 'L11 PC: 0x8049000'. The command history shows '(gdb) layout regs', '(gdb) b \*0x8049031', and 'Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.'. A table of breakpoints is displayed, showing two breakpoints at addresses 0x8049000 and 0x8049031. The command '(gdb) i b' is entered at the prompt.

```
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

native process 4444 In: _start                                L11  PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint        keep y   0x08049000 lab9-2.asm:11
      breakpoint already hit 1 time
2     breakpoint        keep y   0x08049031 lab9-2.asm:22
(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.

```

amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd210 0xffffd210 ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049005 0x8049005 <_start+5> eflags   0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1

native process 4444 In: _start L12 PC: 0x8049005
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--ds      0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb)

```

Рис. 2.10: Изменение регистров

```

amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd210      0xffffd210      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      es       0x2b      43
fs       0x0      0      gs       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
0x8049020 <_start+32>     mov     ecx,0x804a008
0x8049025 <_start+37>     mov     edx,0x7
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1

native process 4444 In: _start      L16      PC: 0x8049016
eflags      0x202      [ IF ]
cs          0x23      35
ss          0x2b      43
--Type <RET> for more, q to quit, c to continue without paging--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 посмотреть значение переменной `msg1`, я воспользовался соответствующей командой для извлечения необходимой информации.



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

Register group: general
eax      0x8      8      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x1           1
esp      0xffffd210 0xffffd210  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      es       0x2b          43
fs       0x0       0      gs       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
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov     eax,0x1

native process 4444 In: _start L16 PC: 0x8049016
(gdb) si
(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>: "Lor!d!\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.

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

Register group: general
eax    0x8      8      ecx    0x804a000    134520832
edx    0x8      8      ebx    0x1      1
esp    0xffffd210 0xffffd210  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     es     0x2b     43
fs     0x0      0      gs     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
0x8049020 <_start+32>   mov    ecx,0x804a008
0x8049025 <_start+37>   mov    edx,0x7
0x804902a <_start+42>   int    0x80
0x804902c <_start+44>   mov    eax,0x1

native process 4444 In: _start L16 PC: 0x8049016
(gdb) p/t $eax
$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.

```

amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x2          2
esp      0xffffd210 0xffffd210  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     es       0x2b         43
fs       0x0      0      gs       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
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov     eax,0x1

native process 4444 In: _start L16 PC: 0x8049016
(gdb) p/s $edx
$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.

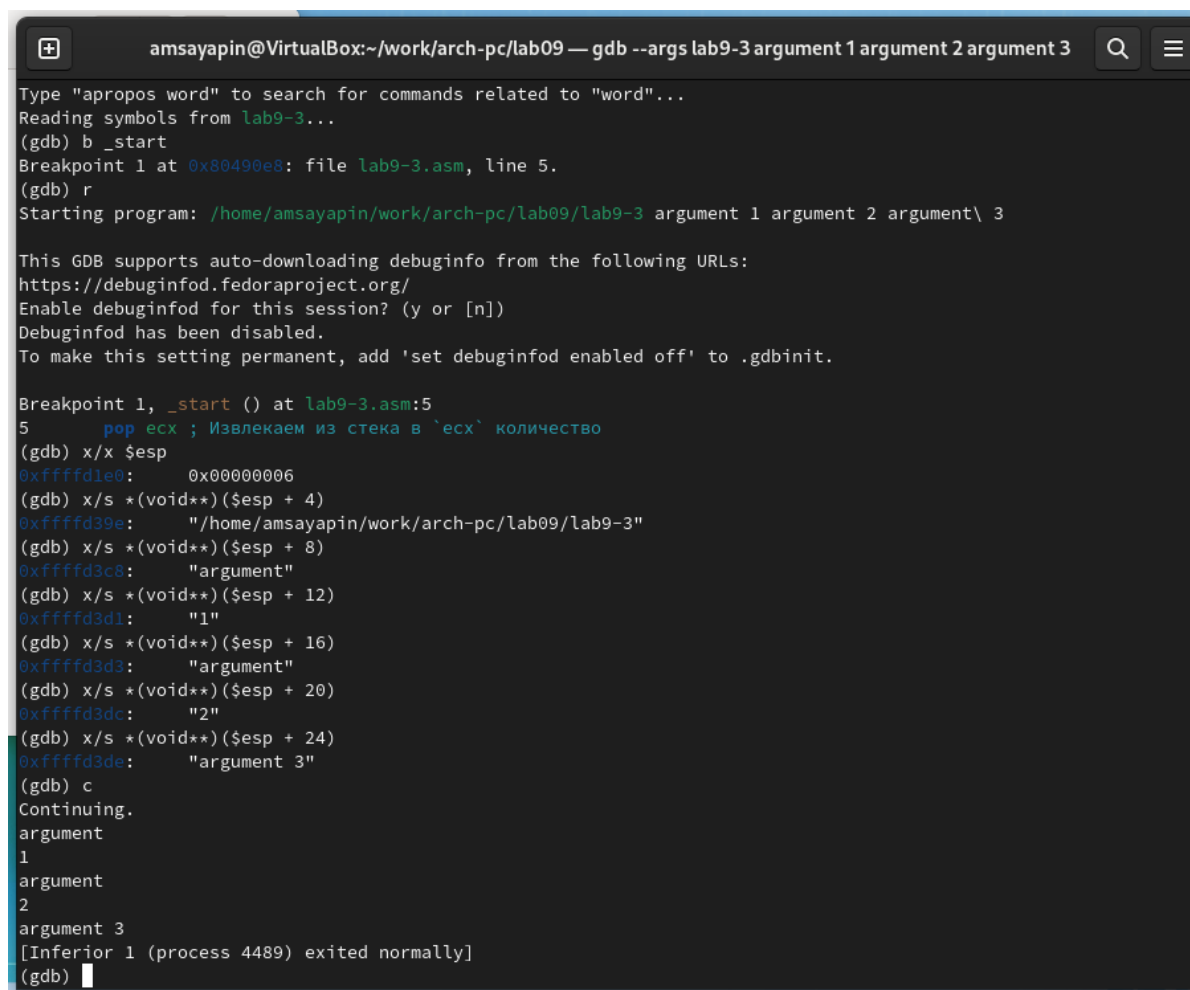
Для запуска программы с аргументами в GDB я использовал опцию `-args`, загрузив исполняемый файл с заданными аргументами в отладчик.

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

Адрес вершины стека, содержащий количество аргументов командной строки (включая название программы), находится в регистре ESP. По этому адресу расположено число, показывающее количество аргументов. В моем случае бы-

ло 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]).



```
amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 argument 3
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/amsayapin/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

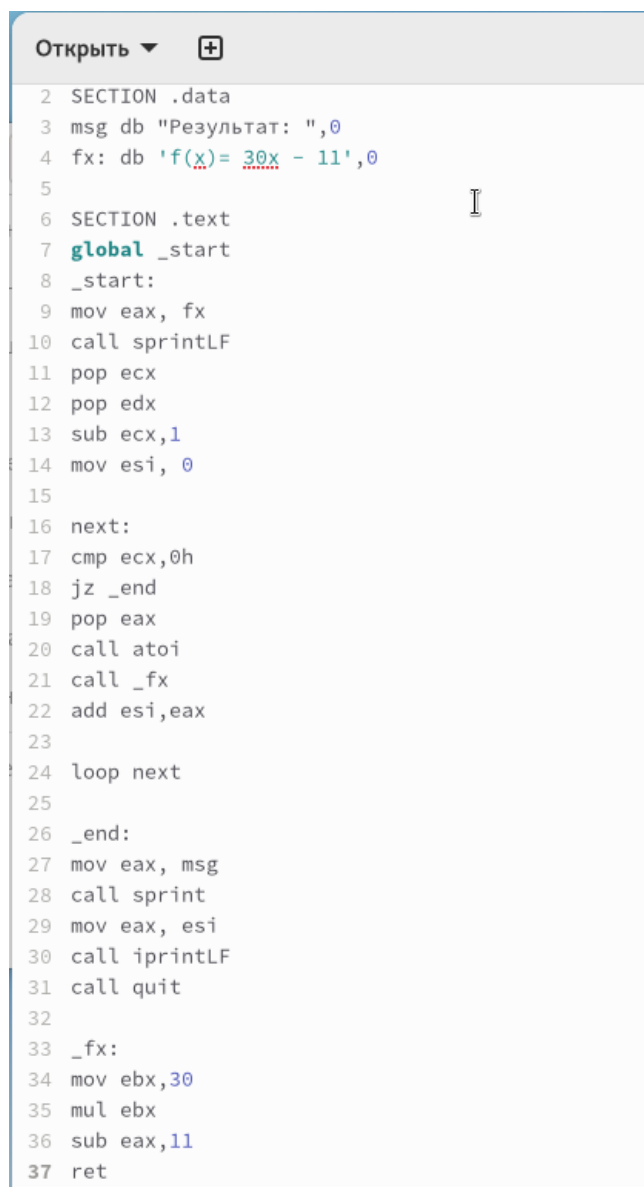
This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1e0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd39e: "/home/amsayapin/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3c8: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3d1: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3d3: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3dc: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3de: "argument 3"
(gdb) c
Continuing.
argument
1
argument
2
argument 3
[Inferior 1 (process 4489) 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  $f(x)$ .




```
Открыть ▾ +
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 30x - 11',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call _fx
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 _fx:
34 mov ebx,30
35 mul ebx
36 sub eax,11
37 ret
```

Рис. 2.16: Код программы lab9-4.asm

```
[amsayarin@VirtualBox lab09]$  
[amsayarin@VirtualBox lab09]$ nasm -f elf lab9-4.asm  
[amsayarin@VirtualBox lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o  
[amsayarin@VirtualBox lab09]$ ./lab9-4  
f(x)= 30x - 11  
Результат: 0  
[amsayarin@VirtualBox lab09]$ ./lab9-4 3 5 7 9  
f(x)= 30x - 11  
Результат: 676  
[amsayarin@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.

Открыть ▾ 

```
1  %include 'in_out.asm'
2  SECTION .data
3  div: DB 'Результат: ',0
4  SECTION .text
5  GLOBAL _start
6  _start:
7  ; ---- Вычисление выражения (3+2)*4+5
8  mov ebx,3
9  mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx|
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

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

```
amsayapin@VirtualBox:~/work/arch-pc/lab09 — gdb lab9-5

eax      0x804a000      134520832      ecx      0x4          4
edx      0x0           0              ebx      0xa          10
esp      0xffffd210    0xffffd210    ebp      0x0          0x0
esi      0x0           0              edi      0xa          10
eip      0x8049105     0x8049105 <_start+29> eflags   0x206         [ PF IF ]
cs       0x23          35             ss       0x2b          43
ds       0x2b          43             es       0x2b          43
fs       0x0           0              gs       0x0          0

B+ 0x80490e8 <_start>      mov     ebx,0x3
0x8049105 <_start+29>      call    0x804900f <sprint>
0x804910a <_start+34>      add     ebx,edi
0x804910c <_start+36>      call    0x8049086 <iprintLF>
0x8049111 <_start+41>      call    0x80490db <quit>
                                ,0x5

                                04a000
>                                rint>

                                86 <iprintLF>

native process 4597 In: _start      L17  PC: 0x8049105
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: 10
[Inferior 1 (process 4597) 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



Открыть ▾ 

```
1  %include 'in_out.asm'
2  SECTION .data
3  div: DB 'Результат: ',0
4  SECTION .text
5  GLOBAL _start
6  _start:
7  ; ---- Вычисление выражения (3+2)*4+5
8  mov ebx,3
9  mov eax,2
10 add eax,ebx
11 mov ecx,4
12 mul ecx
13 add eax,5
14 mov edi,eax
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

Рис. 2.20: Код программы lab9-5.asm исправлен

```
amsayapin@VirtualBox: ~/work/arch-pc/lab09 — gdb lab9-5
eax      0x19      25      ecx      0x4        4
edx      0x0        0      ebx      0x3        3
esp      0xffffd210 0xffffd210 ebp      0x0        0x0
esi      0x0        0      edi      0x19      25
eip      0x8049100 0x8049100 <_start+24> eflags  0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0        0      gs       0x0        0

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>
                                86 <iprintLF>

native process 4645 In: _start L16 PC: 0x8049100
To makeNo process In: , add 'set debuginfod enabled off' to .gdbinit. L?? PC: ??
Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 4645) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.