

# **Отчет по лабораторной работе №10**

**Дисциплина: Архитектура компьютеров**

Лобанова Полина Иннокентьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
<b>3</b>	<b>Выполнение самостоятельной работы</b>	<b>15</b>
<b>4</b>	<b>Вывод</b>	<b>22</b>

# 1 Цель работы

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

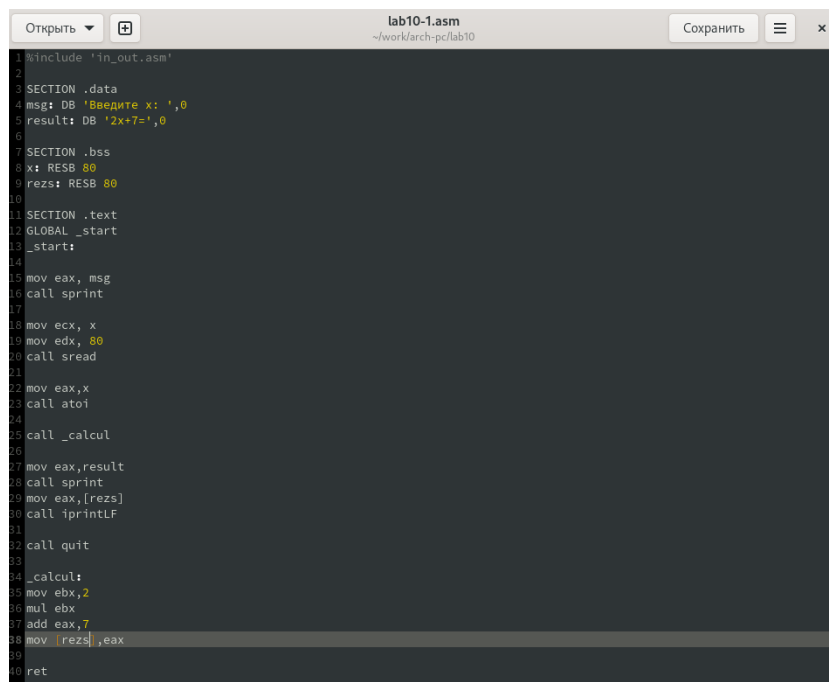
## 2 Выполнение лабораторной работы

1. Создадим каталог lab10 и текстовый файл lab10-1.asm.

```
[pilobanova@i0 ~]$ mkdir ~/work/arch-pc/lab10  
[pilobanova@i0 ~]$ cd ~/work/arch-pc/lab10  
[pilobanova@i0 lab10]$ touch lab10-1.asm
```

Рис. 2.1: Создание каталога lab10 и файла lab10-1.asm.

2. Заполним текстовый файл lab10-1.asm.



```
1 %include "in_out.asm"  
2  
3 SECTION .data  
4 msg: DB 'Введите x: ',0  
5 result: DB '2x+7=',0  
6  
7 SECTION .bss  
8 x: RESB 80  
9 rezs: RESB 80  
10  
11 SECTION .text  
12 GLOBAL _start  
13 _start:  
14  
15 mov eax, msg  
16 call sprint  
17  
18 mov ecx, x  
19 mov edx, 80  
20 call sread  
21  
22 mov eax, x  
23 call atoi  
24  
25 call _calcul  
26  
27 mov eax, result  
28 call sprint  
29 mov eax, [rezs]  
30 call iprintf  
31  
32 call quit  
33  
34 _calcul:  
35 mov ebx, 2  
36 mul ebx  
37 add eax, 7  
38 mov [rezs], eax  
39  
40 ret
```

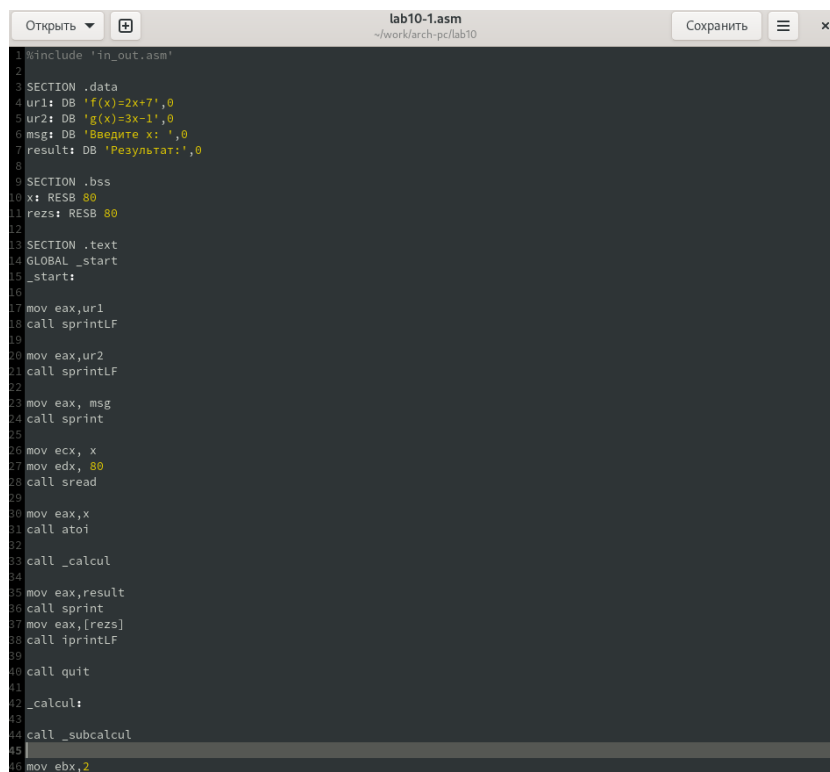
Рис. 2.2: Текст программы в файле lab10-1.asm.

3. Создадим исполняемый файл и проверим его работу.

```
[pilobanova@10 lab10]$ nasm -f elf lab10-1.asm
[pilobanova@10 lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[pilobanova@10 lab10]$ ./lab10-1
Введите x: 5
2x+7=17
[pilobanova@10 lab10]$
```

Рис. 2.3: Создание исполняемого файла и его запуск.

4. Изменим текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ .



```
1 %include "in_out.asm"
2
3 SECTION .data
4 ur1: DB 'f(x)=2x+7',0
5 ur2: DB 'g(x)=3x-1',0
6 msg: DB 'Введите x: ',0
7 result: DB 'Результат:',0
8
9 SECTION .bss
10 x: RESB 80
11 rezs: RESB 80
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax,ur1
18 call sprintf
19
20 mov eax,ur2
21 call sprintf
22
23 mov eax,msg
24 call sprintf
25
26 mov ecx,x
27 mov edx,80
28 call sread
29
30 mov eax,x
31 call atoi
32
33 call _calcul
34
35 mov eax,result
36 call sprintf
37 mov eax,[rezs]
38 call iprintf
39
40 call quit
41
42 _calcul:
43
44 call _subcalcul
45
46 mov ebx,2
```

Рис. 2.4: Измененный текст программы в файле `lab10-1.asm`.

5. Создадим исполняемый файл и проверим его работу.

```
[pilobanova@10 lab10]$ nasm -f elf lab10-1.asm
[pilobanova@10 lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[pilobanova@10 lab10]$ ./lab10-1
f(x)=2x+7
g(x)=3x-1
Введите x: 5
Результат:35
```

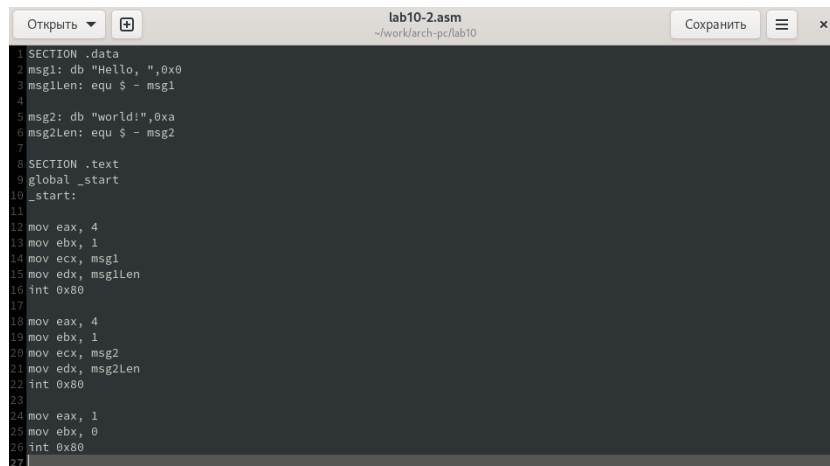
Рис. 2.5: Создание исполняемого файла и его запуск.

6. Создадим текстовый файл lab10-2.asm.

```
[pilobanova@i0 lab10]$ touch lab10-2.asm
```

Рис. 2.6: Создание текстового файла lab10-2.asm.

7. Заполним файл lab10-2.asm.



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4
5 msg2: db "world!",0xa
6 msg2Len: equ $ - msg2
7
8 SECTION .text
9 global _start
10 _start:
11
12 mov eax, 4
13 mov ebx, 1
14 mov ecx, msg1
15 mov edx, msg1Len
16 int 0x80
17
18 mov eax, 4
19 mov ebx, 1
20 mov ecx, msg2
21 mov edx, msg2Len
22 int 0x80
23
24 mov eax, 1
25 mov ebx, 0
26 int 0x80
27
```

Рис. 2.7: Текст программы в файле lab10-2.asm.

8. Создадим исполняемый файл и проверим его работу.

```
[pilobanova@i0 lab10]$ nasm -f elf -g -l lab10-2.lst lab10-2.asm
[pilobanova@i0 lab10]$ ld -m elf_i386 -o lab10-2 lab10-2.o
[pilobanova@i0 lab10]$ ./lab10-2
Hello, world!
```

Рис. 2.8: Создание исполняемого файла и его запуск.

9. Загрузим исполняемый файл в отладчик gdb.

```
[pilobanova@i0 lab10]$ gdb lab10-2
GNU gdb (GDB) Fedora 11.2-3.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 lab10-2...
(gdb) █
```

Рис. 2.9: Загрузка исполняемого файла на gdb.

10. Проверим работу программы, запустив ее в оболочке GDB с помощью команды run.

```
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 2918) exited normally]
```

Рис. 2.10: Запуск программы.

11. Установим брейкпоинт на метку \_start и запустим.

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 12.
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:12
12      mov eax, 4
(gdb) █
```

Рис. 2.11: Установка брейкпоинта и запуск программы.

12. Посмотрим дисассемблированный код программы с помощью команды disassemble начиная с метки \_start.

```
(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. 2.12: Дисассимилированный код.

13. Переключимся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`.

```
(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. 2.13: Переход на отображение команд с Intel'овским синтаксисом.

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel заключаются в том, что у АТТ регистры стоят справа и перед каждым регистром стоит символ %, также перед каждой строкой стоит символ \$, у Intel же регистры стоят слева и нет дополнительных символов.

14. Включим режим псевдографики.



```

[ Register Values Unavailable ]

0x8049000 <_start>    mov    $0x4,%eax
0x8049005 <_start+5>   mov    $0x1,%ebx
0x804900a <_start+10>  mov    $0x804a000,%ecx
0x804900f <_start+15>  mov    $0x8,%edx
0x8049014 <_start+20>  int     $0x80
0x8049016 <_start+22>  mov    $0x4,%eax
0x804901b <_start+27>  mov    $0x1,%ebx

exec No process in:                               L??  PC: ??
(gdb) layout regs
(gdb)

```

Рис. 2.14: Режим псевдографики.

15. Проверим наличие брейкпоинтов.

```

(gdb) info breakpoints
Num   Type       Disp Enb Address      What
1     breakpoint keep  y   0x08049000 lab10-2.asm:12
(gdb)

```

Рис. 2.15: Проверка брейкпоинтов.

16. Установим еще одну точку останова по адресу инструкции.

```

(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 25.
(gdb)

```

Рис. 2.16: Установка точки останова.

17. Посмотрим информацию о всех установленных точках останова.

```

(gdb) i b
Num   Type       Disp Enb Address      What
1     breakpoint keep  y   0x08049000 lab10-2.asm:12
2     breakpoint keep  y   0x08049031 lab10-2.asm:25
(gdb)

```

Рис. 2.17: Проверка точек останова.

18. Выполним 5 инструкций с помощью команды stepi.

eax	0x0	0
eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	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
native process 15270 In: _start L12 PC: 0x8049000
Breakpoint process 15275 In: _start lab10-2.asm, line 25. L12 PC: 0x8049000
(gdb) c
Continuing.
[Inferior 1 (process 15270) exited normally]
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:12
(gdb)

```

Рис. 2.18: Значения регистров до команды *stepi*.

eax	0x0	0
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1

```

    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
native process 15270 In: _start L12 PC: 0x8049000
Breakpoint process 15275 In: _start lab10-2.asm, line 25. L18 PC: 0x8049016
Continuing.
[Inferior 1 (process 15270) exited normally]
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:12
(gdb) si 5
(gdb)

```

Рис. 2.19: Значения регистров после команды *stepi*.

Значение регистра *eax* изменилось с 0 на 8. Значение регистра *ecx* изменилось с 0 на 134520832. Значение регистра *edx* изменилось с 0 на 8. И значение регистра *ebx* изменилось с 0 на 1.

19. Посмотрим содержимое регистров с помощью команды *info registers*.

```

native process 4509 In: _start L18 PC: 0x8049016
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1f0 0xffffd1f0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 2.20: Содержимое регистров.

```

native process 4509 In: _start L18 PC: 0x8049016
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--css
0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb)

```

Рис. 2.21: Содержимое регистров.

20. Посмотрим значение переменной msg1 по имени.

```

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

```

Рис. 2.22: Значение переменной msg1.

21. Посмотрим значение переменной msg2 по адресу.

```

(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 2.23: Значение переменной msg2.

22. Изменим первый символ переменной msg1.

```

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

```

Рис. 2.24: Изменение переменной msg1.

23. Заменим символ в переменной msg2.

```
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "World!\n\034"
(gdb)
```

Рис. 2.25: Изменение переменной msg2.

24. Выведем в различных форматах значение регистра edx.

```
(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb)
```

Рис. 2.26: Значение регистра edx в различных форматах.

25. Изменим значение регистра edx с помощью команды set.

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)
```

Рис. 2.27: Изменение значение регистра edx.

Разница вывода команд p/s \$ebx заключается в том, что в первом случае мы вносим значение 2, а во втором регистр приравнивается к 2.

26. Завершим выполнение программы с помощью команды continue.

```
(gdb) c
Continuing.
World!

Breakpoint 2, _start () at lab10-2.asm:25
(gdb) q
A debugging session is active.

    Inferior 1 [process 4509] will be killed.

Quit anyway? (y or n) y
```

Рис. 2.28: Завершение выполнения программы.

27. Скопируем файл lab9-2.asm в каталог lab10 с названием lab10-3.asm.

```
[pilobanova@10 lab10]$ cd ..
[pilobanova@10 arch-pc]$ cd lab09
[pilobanova@10 lab09]$ ls
in_out.asm lab9-1 lab9-1.o lab9-2.asm lab9-3 lab9-3.o lab9.o
lab9 lab9-1.asm lab9-2 lab9-2.o lab9-3.asm lab9.asm
[pilobanova@10 lab09]$ cp lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm
[pilobanova@10 lab09]$ cd ..
[pilobanova@10 arch-pc]$ cd lab10
[pilobanova@10 lab10]$ ls
a.out lab10-1 lab10-1.o lab10-2.asm lab10-2.o
in_out.asm lab10-1.asm lab10-2 lab10-2.lst lab10-3.asm
[pilobanova@10 lab10]$
```

Рис. 2.29: Копирование файла lab9-2.asm.

28. Создадим исполняемый файл.

```
[pilobanova@10 lab10]$ nasm -f elf -g -l lab10-3.lst lab10-3.asm
[pilobanova@10 lab10]$ ld -m elf_i386 -o lab10-3 lab10-3.o
```

Рис. 2.30: Создание исполняемого файла.

29. Загрузим исполняемый файл в отладчик, указав аргументы.

```
[pilobanova@10 lab10]$ gdb --args lab10-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora 11.2-3.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 lab10-3...
(gdb)
```

Рис. 2.31: Загрузка исполняемого файла в отладчик.

30. Установим точку останова перед первой инструкцией в программе и запустим ее.

```
(gdb) b _start
Breakpoint 2 at 0x80490e8: file lab10-3.asm, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10-3 аргумент1 аргумент
2 аргумент\ 3

Breakpoint 2, _start () at lab10-3.asm:7
7      pop ecx
(gdb)
```

Рис. 2.32: Установка точки останова.

31. Посмотрим число аргументов командной строки.

```
(gdb) x/x $esp
0xffffd1a0: 0x00000005
(gdb)
```

Рис. 2.33: Число аргументов командной строки.

32. Посмотрим остальные позиции стека.

```
(gdb) x/s *(void**)(esp + 4)
0xffffd358: "/home/pilobanova/work/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd384: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd39c: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd3a7: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd3a9: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.34: Позиции стека.

Шаг изменения адреса равен 4, потому что стек может хранить до 4 байт.

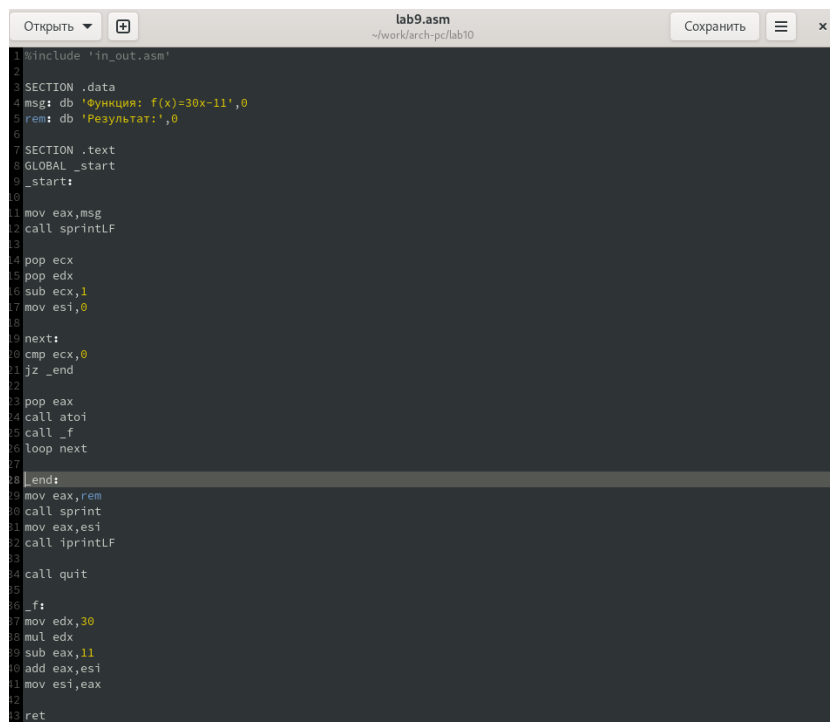
### 3 Выполнение самостоятельной работы

1. Перенесем файл lab9.asm в каталог lab10.

```
[pilobanova@i0 lab10]$ cp ~/work/arch-pc/lab09/lab9.asm ~/work/arch-pc/lab10
[pilobanova@i0 lab10]$ ls
a.out      lab10-1.asm  lab10-2.asm  lab10-3      lab10-3.o
in_out.asm lab10-1.o    lab10-2.lst  lab10-3.asm  lab9.asm
lab10-1    lab10-2      lab10-2.o    lab10-3.lst
```

Рис. 3.1: Копирование файла lab9.asm в каталог lab10.

2. Преобразуем программу из лабораторной работы №9, реализовав вычисление значения функции  $f(x)$ , как подпрограмму.



```
1 #include "in_out.asm"
2
3 SECTION .data
4 msg: db 'Функция: f(x)=30x-11',0
5 rem: db 'Результат:',0
6
7 SECTION .text
8 GLOBAL _start
9 _start:
10
11 mov eax,msg
12 call sprintf
13
14 pop ecx
15 pop edx
16 sub ecx,1
17 mov esi,0
18
19 next:
20 cmp ecx,0
21 jz _end
22
23 pop eax
24 call atoi
25 call _f
26 loop next
27
28 _end:
29 mov eax,rem
30 call sprint
31 mov eax,esi
32 call iprintf
33
34 call quit
35
36 _f:
37 mov edx,30
38 mul edx
39 sub eax,11
40 add eax,esi
41 mov esi,eax
42
43 ret
```

Рис. 3.2: Измененный текст программы в файле lab9.asm.

3. Создадим исполняемый файл и проверим его работу.

```
[pilobanova@i10 lab10]$ nasm -f elf lab9.asm
[pilobanova@i10 lab10]$ ld -m elf_i386 -o lab9 lab9.o
[pilobanova@i10 lab10]$ ./lab9 1 2 3 4
Функция: f(x)=30x-11
Результат:256
```

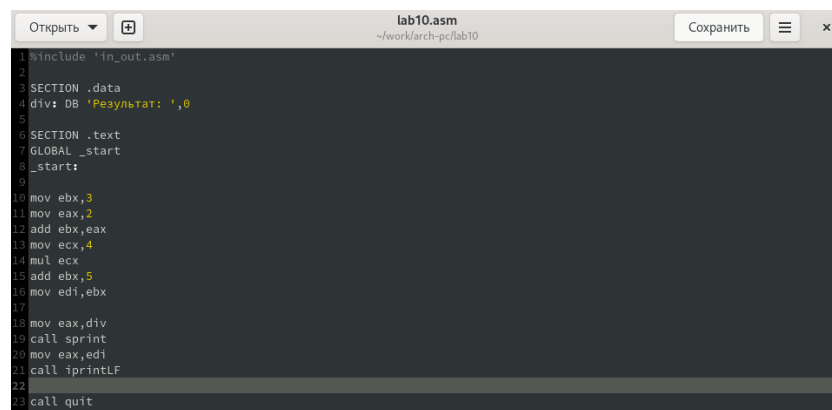
Рис. 3.3: Создание исполняемого файла и его запуск.

4. Создадим текстовый файл lab10.asm.

```
[pilobanova@i10 lab10]$ touch lab10.asm
```

Рис. 3.4: Создание текстового файла lab10.asm.

5. Запишем программу вычисления выражения  $(3 + 2) * 4 + 5$ .



```
1 %include "in_out.asm"
2
3 SECTION .data
4 div: DB 'Результат: ',0
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov ebx,3
11 mov eax,2
12 add ebx,eax
13 mov ecx,4
14 mul ecx
15 add ebx,5
16 mov edi,ebx
17
18 mov eax,div
19 call sprint
20 mov eax,edi
21 call iprintLF
22
23 call quit
```

Рис. 3.5: Текст программы в файле lab10.asm.

6. Создадим исполняемый файл и проверим его работу. Как и ожидалось, программа выдает неверный результат.

```
[pilobanova@i10 lab10]$ nasm -f elf -g -l lab10.lst lab10.asm
[pilobanova@i10 lab10]$ ld -m elf_i386 -o lab10 lab10.o
[pilobanova@i10 lab10]$ ./lab10
Результат: 10
```

Рис. 3.6: Создание исполняемого файла и его запуск.

7. Используем отладчик для определения ошибки.



```
[pilobanova@fedora lab10]$ gdb lab10
GNU gdb (GDB) Fedora 11.2-3.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 lab10...
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Результат: 10
[Inferior 1 (process 15504) exited normally]
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10.asm, line 10.
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10

Breakpoint 1, _start () at lab10.asm:10
10      mov ebx,3
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 15512) exited normally]
```

Рис. 3.7: Использование отладчика.

```
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>: mov     $0x3,%ebx
   0x080490ed <+5>: mov     $0x2,%eax
   0x080490f2 <+10>: add     %eax,%ebx
   0x080490f4 <+12>: mov     $0x4,%ecx
   0x080490f9 <+17>: mul     %ecx
   0x080490fb <+19>: add     $0x5,%ebx
   0x080490fe <+22>: mov     %ebx,%edi
   0x08049100 <+24>: mov     $0x804a000,%eax
   0x08049105 <+29>: call    0x804909f <sprint>
   0x0804910a <+34>: mov     %edi,%eax
   0x0804910c <+36>: call    0x8049086 <printLF>
   0x08049111 <+41>: call    0x80490db <quit>
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>: mov     ebx,0x3
   0x080490ed <+5>: mov     eax,0x2
   0x080490f2 <+10>: add     ebx,eax
   0x080490f4 <+12>: mov     ecx,0x4
   0x080490f9 <+17>: mul     ecx
   0x080490fb <+19>: add     ebx,0x5
   0x080490fe <+22>: mov     edi,ebx
   0x08049100 <+24>: mov     eax,0x804a000
   0x08049105 <+29>: call    0x804909f <sprint>
   0x0804910a <+34>: mov     eax,edi
   0x0804910c <+36>: call    0x8049086 <printLF>
   0x08049111 <+41>: call    0x80490db <quit>
End of assembler dump.
```

Рис. 3.8: Использование отладчика.

```

pilobanova@10:~/work/arch-pc/lab10 — gdb lab10

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd200 0xffffd200 ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490e8 0x80490e8 <_start> 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
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
0x8049118 add BYTE PTR [eax],al

native process 15529 In: _start L10 PC: 0x80490e8
(gdb) layout regs
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10
Breakpoint 1, _start () at lab10.asm:10
(gdb)

```

Рис. 3.9: Использование отладчика.

8. Пошагово проверим каждое действие и проследим изменения регистров.

```

pilobanova@10:~/work/arch-pc/lab10 — gdb lab10

--Register group: general--
eax      0x2      2      ecx      0x0      0
edx      0x0      0      ebx      0x3      3
esp      0xffffd200 0xffffd200 ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490f2 0x80490f2 <_start+1> 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
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
0x8049118 add BYTE PTR [eax],al

native process 15529 In: _start L12 PC: 0x80490f2
(gdb) layout regs
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10
Breakpoint 1, _start () at lab10.asm:10
(gdb) si 2
(gdb)

```

Рис. 3.10: Первые две инструкции.

```

pilobanova@10:~/work/arch-pc/lab10 — gdb lab10

Register group: general
eax      0x2      2      ecx      0x0      0
edx      0x0      0      ebx      0x5      5
esp      0xffffd200 0xffffd200 ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+1> 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
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
0x8049118             add     BYTE PTR [eax],al

native process 15529 In: _start L13 PC: 0x80490f4
(gdb) layout regs
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10

Breakpoint 1, _start () at lab10.asm:10
(gdb) si 2
(gdb) si 1
(gdb)

```

44

Из-за того, что при сложении не соблюден необходимый порядок регистров, результат сложения записывается в регистр ebx.

```

pilobanova@10:~/work/arch-pc/lab10 — gdb lab10

Register group: general
eax      0x8      8      ecx      0x4      4
edx      0x0      0      ebx      0x5      5
esp      0xffffd200 0xffffd200 ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490fb 0x80490fb <_start+1> 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
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
0x8049118             add     BYTE PTR [eax],al

native process 15529 In: _start L15 PC: 0x80490fb
(gdb) layout regs
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10

Breakpoint 1, _start () at lab10.asm:10
(gdb) si 2
(gdb) si 1
(gdb) si 2
(gdb)

```

45

Поскольку результат сложения записан не в eax, умножение выполняется неверно.

```

Register group: general
eax 0x8 8 ecx 0x4 4
edx 0x0 0 ebx 0xa 10
esp 0xffffd200 0xffffd200 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x80490fe 0x80490fe <_start+2> 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
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 <iprintf>
0x8049111 <_start+41> call 0x80490db <quit>
0x8049116 add BYTE PTR [eax],al
0x8049118 add BYTE PTR [eax],al

native process 15529 In: _start L16 PC: 0x80490fe
(gdb) layout regs
(gdb) run
Starting program: /home/pilobanova/work/arch-pc/lab10/lab10
Breakpoint 1, _start () at lab10.asm:10
(gdb) si 2
(gdb) si 1
(gdb) si 2
(gdb) si 1
(gdb)

```

46

Из-за того, что умножение выполнено неверно, прибавление 5 к регистру ebx выдает не тот ответ, который должен быть.

## 9. Исправим ошибки в тексте программы.

```

lab10.asm
~/work/arch-pc/lab10

1 %include 'in_out.asm'
2
3 SECTION .data
4 div: DB 'Результат: ',0
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov ebx,3
11 mov eax,2
12 add eax,ebx
13 mov ecx,4
14 mul ecx
15 add eax,5
16 mov edi,eax
17
18 mov eax,div
19 call sprint
20 mov eax,edi
21 call iprintf
22
23 call quit

```

Рис. 3.11: Измененный текст программы в файле.

10. Создадим исполняемый файл и проверим его работу. После изменений программа начала работать исправно.

```
[pilobanova@fedora lab10]$ nasm -f elf -g -l lab10.lst lab10.asm
[pilobanova@fedora lab10]$ ld -m elf_i386 -o lab10 lab10.o
[pilobanova@fedora lab10]$ ./lab10
Результат: 25
[pilobanova@fedora lab10]$
```

Рис. 3.12: Создание исполняемого файла и его запуск.

## 4 Вывод

Я научилась писать программы с использованием подпрограмм, а также ознакомилась с методами отладки и при помощи GDB и его основными возможностями.