

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютера

Кайнова Екатерина Андреевна НПИбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программы с помощью GDB	9
2.3	Работа с аргументами командной строки	18
2.4	Задание для самостоятельной работы	19
2.5	Выводы	25

Список иллюстраций

2.1	Исходный код программы lab9-1.asm	7
2.2	Результат выполнения программы	7
2.3	Модифицированный код программы	8
2.4	Результат выполнения модифицированной программы	9
2.5	Код программы lab9-2.asm	10
2.6	Запуск программы в отладчике	11
2.7	Дизассемблированный код	12
2.8	Дизассемблированный код в режиме Intel	13
2.9	Установка точки останова	14
2.10	Изменение значений регистров	15
2.11	Отслеживание изменений регистров	16
2.12	Изменение переменной	16
2.13	Отображение измененного регистра	17
2.14	Изменение регистра ebx	18
2.15	Просмотр аргументов командной строки	19
2.16	Код программы prog-1.asm	20
2.17	Результат выполнения программы	21
2.18	Код с ошибками	22
2.19	Результат отладки	23
2.20	Исправленный код программы	24
2.21	Результат проверки	25

Список таблиц

1 Цель работы

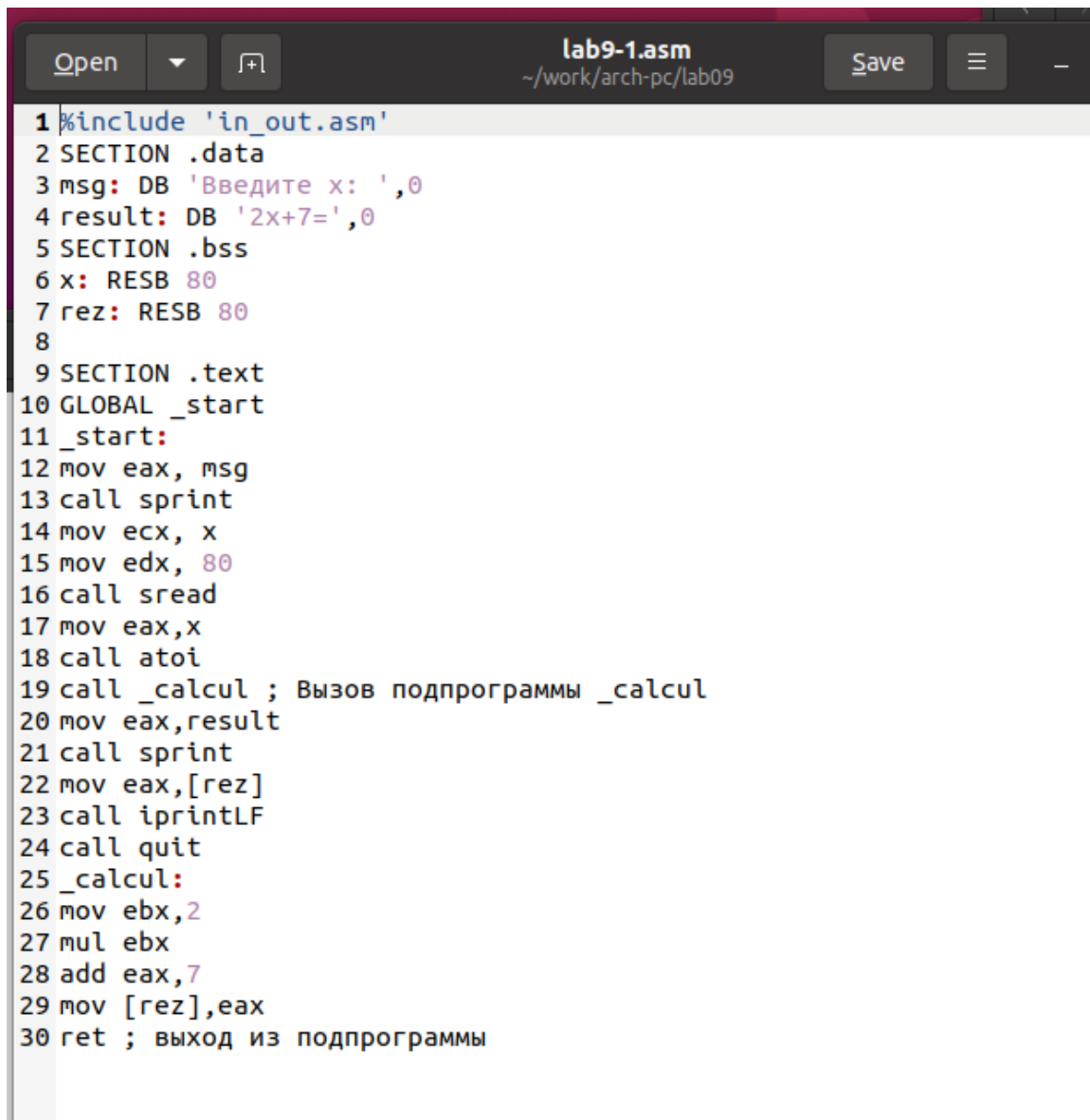
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

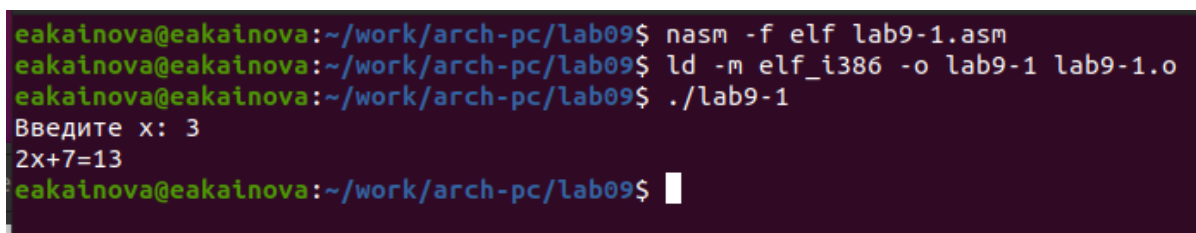
Сначала я создала новую папку для выполнения лабораторной работы №9 и перешла в нее. Затем создала файл с именем lab9-1.asm.

В качестве примера я реализовала программу, вычисляющую арифметическое выражение $f(x) = 2x + 7$ с использованием подпрограммы `calcul`. Значение переменной x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



```
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: Исходный код программы lab9-1.asm

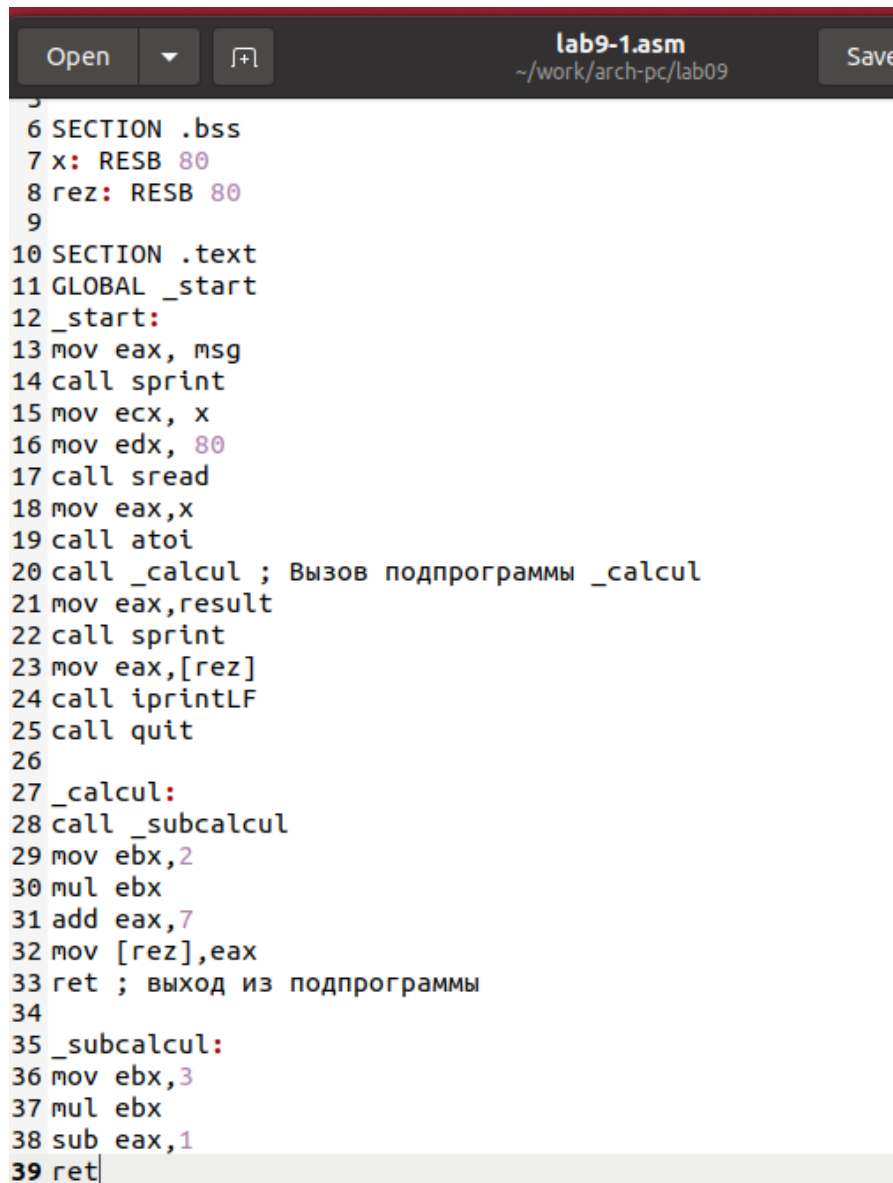


```
eakainova@eakainova:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
eakainova@eakainova:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
eakainova@eakainova:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2x+7=13
eakainova@eakainova:~/work/arch-pc/lab09$
```

Рис. 2.2: Результат выполнения программы

Затем я изменила текст программы, добавив подпрограмму subcalcul внутрь

подпрограммы `calcul`. Это позволило вычислять составное выражение $f(g(x))$, где $f(x) = 2x + 7$, $g(x) = 3x - 1$. Значение x также вводится с клавиатуры.



```
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
```

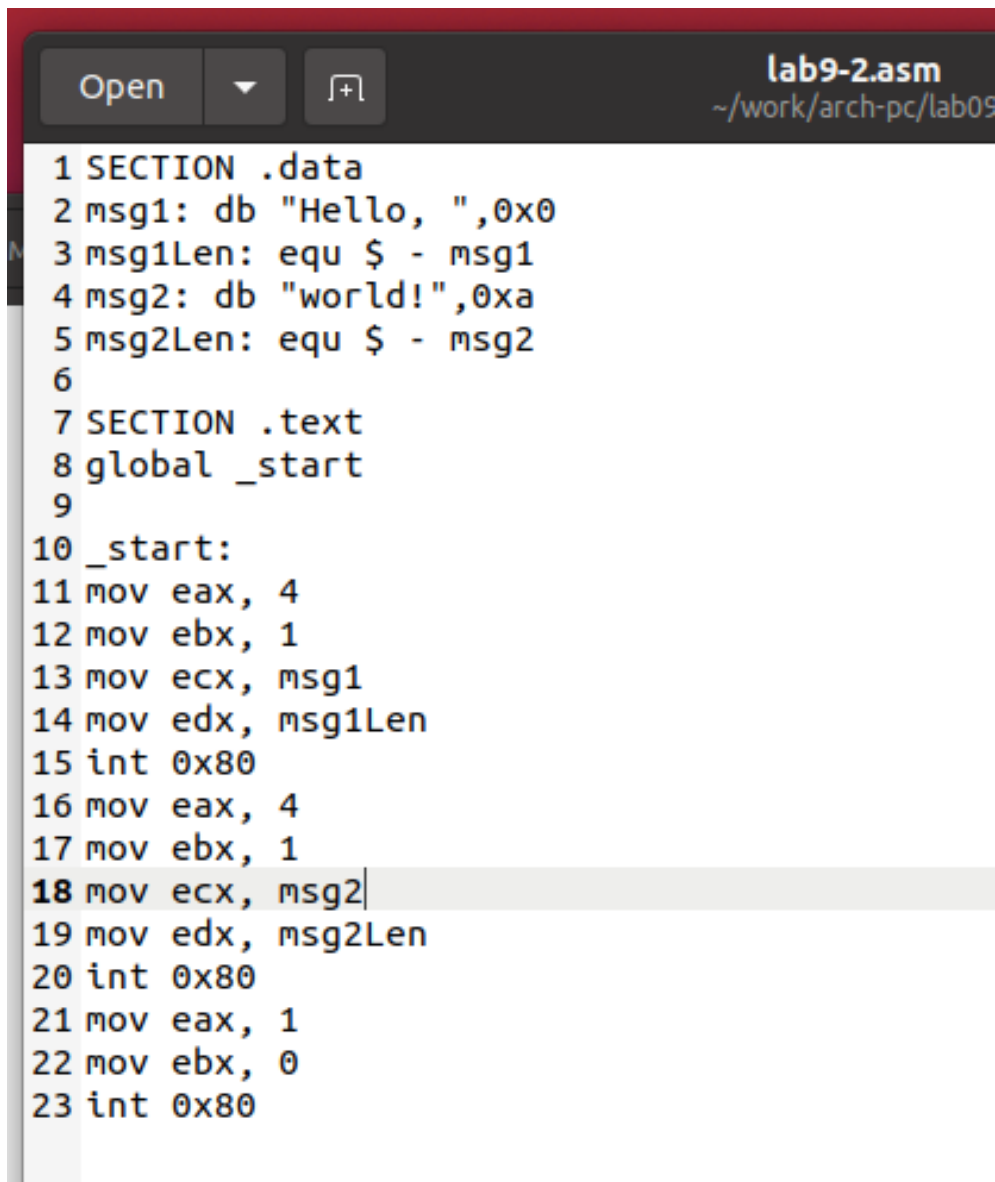
Рис. 2.3: Модифицированный код программы


```
eakainova@eakainova:~/work/arch-pc/lab09$  
eakainova@eakainova:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
eakainova@eakainova:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
eakainova@eakainova:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 3  
2(3x-1)+7=23  
eakainova@eakainova:~/work/arch-pc/lab09$
```

Рис. 2.4: Результат выполнения модифицированной программы

2.2 Отладка программы с помощью GDB

Я создала файл lab9-2.asm, содержащий программу для вывода сообщения “Hello, world!” (Листинг 9.2).



```
lab9-2.asm
~/work/arch-pc/lab09

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

Скомпилировала файл с ключом -g для добавления отладочной информации и загрузила его в GDB. Затем запустила программу командой run.

```
eakainova@eakainova:~/work/arch-pc/lab09$  
eakainova@eakainova:~/work/arch-pc/lab09$ gdb lab9-2  
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2  
Copyright (C) 2020 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-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://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) run  
Starting program: /home/eakainova/work/arch-pc/lab09/lab9-2  
Hello, world!  
[Inferior 1 (process 2761) exited normally]  
(gdb)
```

Рис. 2.6: Запуск программы в отладчике

Установила точку останова на метке `_start`, запустила программу, а затем просмотрела дизассемблированный код.

```
eakainova@eakainova: ~/work/arch-pc/lab09
[Inferior 1 (process 2761) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/eakainova/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 in _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. 2.7: Дизассемблированный код

```
eakainova@eakainova: ~/work/arch-pc/lab09
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 Intel

Установила дополнительные точки останова, используя команды `info breakpoints` и `break`. Например, добавила точку на инструкции `mov ebx, 0x0`.

```
eakainova@eakainova: ~/work/arch-pc/lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

native process 2765 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(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. 2.9: Установка точки останова

С помощью команды `stepi` (или `si`) я пошагово выполняла инструкции, отслеживая изменения регистров.

```
eakainova@eakainova: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

native process 2765 In: _start L?? PC: 0x8049005
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
gs       0x0      0
(gdb) si
0x8049005 in _start ()
(gdb) 
```

Рис. 2.10: Изменение значений регистров

```
eakainova@eakainova: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

native process 2765 In: _start L?? PC: 0x8049016
ds        0x2b      43
es        0x2b      43
fs        0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
gs        0x0      0
(gdb) si
0x8049005 in _start ()
(gdb) si
0x804900a in _start ()
(gdb) si
0x804900f in _start ()
(gdb) si
0x8049014 in _start ()
(gdb) si
0x8049016 in _start ()
(gdb) 
```

Рис. 2.11: Отслеживание изменений регистров

Я также изменила значение переменной `msg1` и регистров, используя команду `set`.

```
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "Hello, "
(gdb)
(gdb) x/1sb 0x804a0080x804a008 <msg2>:  "world!\n"
(gdb)
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "hello, "
(gdb)
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n"
(gdb) 
```

Рис. 2.12: Изменение переменной


```
eakainova@eakainova: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

native process 2765 In: start L?? PC: 0x8049016
0x804a008 <msg2>: "Lor!d!\n"
(gdb)
(gdb) p/s $eax$1 = 8
(gdb)
(gdb) p/t $eax$2 = 1000
(gdb)
(gdb) p/s $ecx$3 = 134520832
(gdb)
(gdb) p/x $ecx$4 = 0x804a000
(gdb)
(gdb) p/s $edx$5 = 8
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Отображение измененного регистра

Используя аналогичные команды, я изменила значение регистра ebx.

```
eakainova@eakainova: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

native process 2765 In: start L?? PC: 0x8049016
(gdb) p/s $ecx$3 = 134520832
(gdb)
(gdb) p/x $ecx$4 = 0x804a000
(gdb)
(gdb) p/s $edx$5 = 8
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
(gdb)
(gdb) p/s $ebx$8 = 50
(gdb)
(gdb) p/s $ebx
$9 = 2
(gdb)
```

Рис. 2.14: Изменение регистра ebx

2.3 Работа с аргументами командной строки

Для работы с аргументами командной строки я использовала файл lab8-2.asm (из лабораторной работы №8), создав из него исполняемый файл. Затем загрузила программу в GDB с аргументами, используя ключ `-args`.

```
eakainova@eakainova: ~/work/arch-pc/lab09
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) run
Starting program: /home/eakainova/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xffffd1a0: 0x00000006
(gdb)
0xffffd1a4: 0xffffd361
(gdb) x/s *(void**)($esp + 4)
0xffffd361: "/home/eakainova/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd38b: "argument"
(gdb) x/s *(void**)($esp + 12)
0xffffd394: "1"
(gdb) x/s *(void**)($esp + 16)
0xffffd396: "argument"
(gdb) x/s *(void**)($esp + 20)
0xffffd39f: "2"
(gdb) x/s *(void**)($esp + 24)
0xffffd3a1: "argument 3"
(gdb) █
```

Рис. 2.15: Просмотр аргументов командной строки

2.4 Задание для самостоятельной работы

В рамках задания я модифицировала программу из лабораторной работы №8, добавив подпрограмму для вычисления функции $f(x)$.

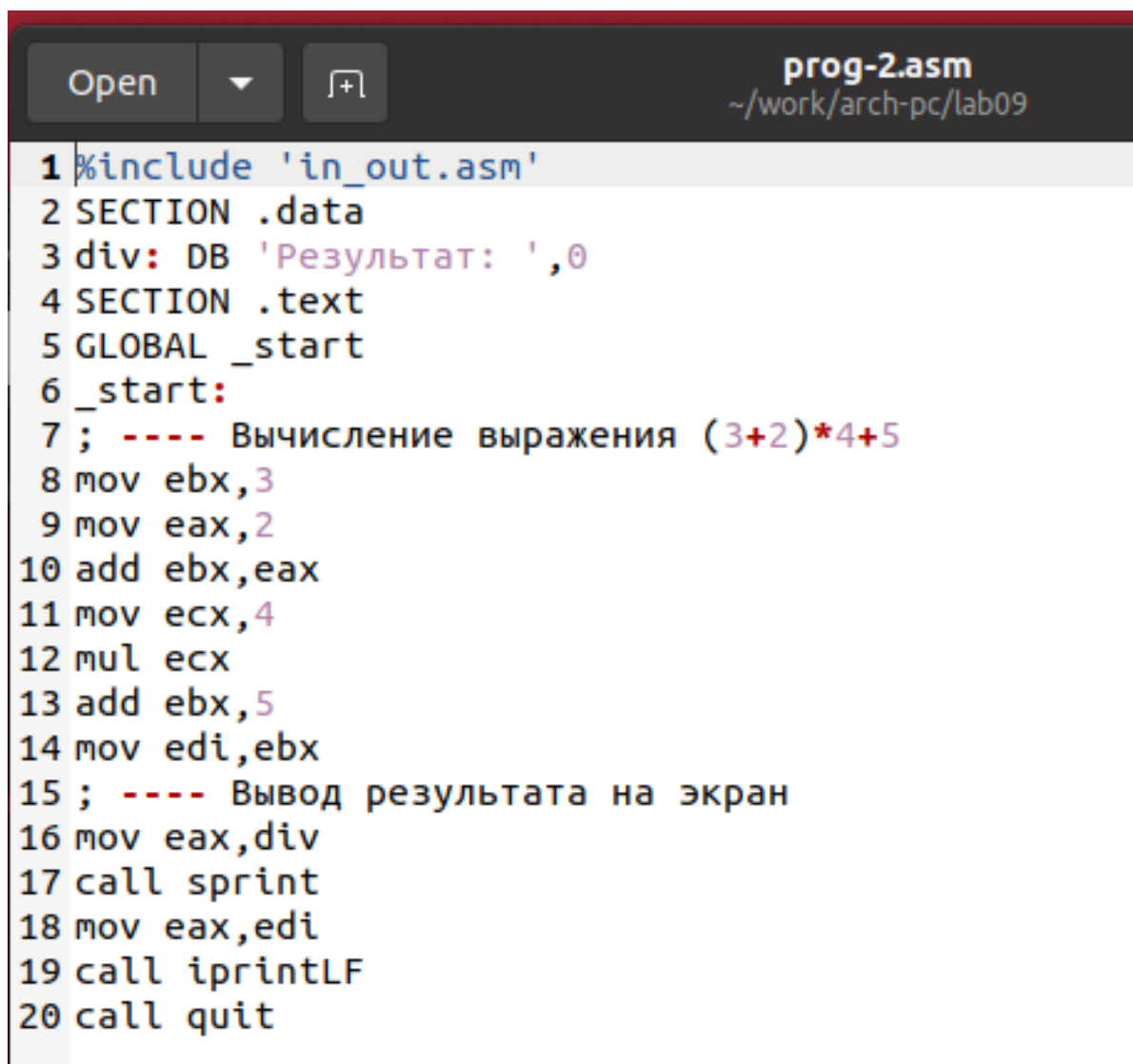
```
1 msg db "Результат: ",0
2
3
4 fx: db 'f(x)= 15x + 2',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call subproc
22 add esi,eax
23
24 loop next
25
26 _end: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

Рис. 2.16: Код программы prog-1.asm

```
eakainova@eakainova:~/work/arch-pc/lab09$  
eakainova@eakainova:~/work/arch-pc/lab09$ nasm -f elf prog-1.asm  
  
eakainova@eakainova:~/work/arch-pc/lab09$ ld -m elf_i386 prog-1.o -o prog-1  
eakainova@eakainova:~/work/arch-pc/lab09$ ./prog-1  
f(x)= 15x + 2  
Результат: 0  
eakainova@eakainova:~/work/arch-pc/lab09$ ./prog-1 3  
f(x)= 15x + 2  
Результат: 47  
eakainova@eakainova:~/work/arch-pc/lab09$ ./prog-1 3 6 7  
f(x)= 15x + 2  
Результат: 246  
eakainova@eakainova:~/work/arch-pc/lab09$
```

Рис. 2.17: Результат выполнения программы

В процессе выполнения программы я обнаружила ошибку: порядок аргументов в инструкции add был перепутан, а регистр ebx вместо eax отправлялся в edi.



```
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
```

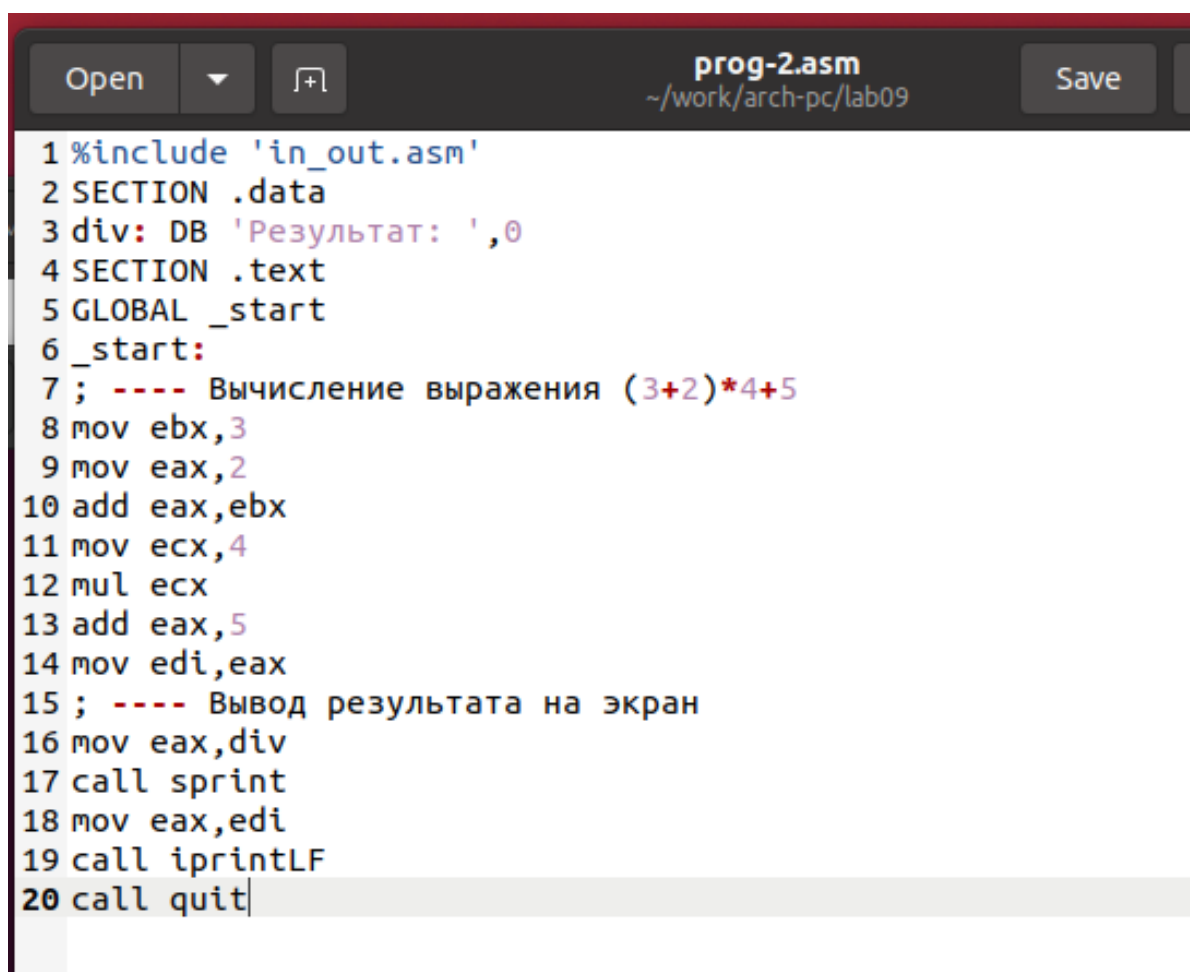
Рис. 2.18: Код с ошибками

```
eakainova@eakainova: ~/work/arch-pc/lab09
eax      0x804a000      134520832
ecx      0x4           4
edx      0x0           0
ebx      0xa           10
esp      0xffffd1dc    0xffffd1dc
ebp      0x0           0x0
esi      0x0           0
edi      0xa           10
eip      0x804900f      0x804900f <sprint>

>0x804900f <sprint> push edx
B+ 0x80490e8 <_start> mov ebx,0x3
0x80490ed <_start+5> mov eax,0x2
0x80490f2 <_start+10> add ebx,eax
0x80490f4 <_start+12> mov ecx,0x400 <slen>
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>
native process 2800 In: sprint L?? PC: 0x804900f
0x08049049 No process In: L?? PC: ??
0x08049100 in _start ()
(gdb) si
0x08049105 in _start ()
(gdb) si
0x0804900f in sprint ()
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 2800) exited normally]
(gdb) █
```

Рис. 2.19: Результат отладки

После исправления ошибок программа заработала корректно.



```
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: Исправленный код программы


```
eakainova@eakainova: ~/work/arch-pc/lab09
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd1e0 0xffffd1e0
ebp      0x0       0
esi      0x0       0
edi      0x19      25
eip      0x8049100 0x8049100 <_start+24>

0x80490f9 <_start+17> mul    ecx
B+ 0x80490e8 <_start>    mov    ebx,0x3
0x80490ed <_start+5>    mov    eax,0x2
>0x80490f2 <_start+10> add    eax,ebx;0x4a000
0x80490f4 <_start+12> mov    ecx,0x4      rint>
0x80490f9 <_start+17> mul    ecx
0x80490fb <_start+19> add    eax,0x586 <iprintLF>
0x80490fe <_start+22> mov    edi,eax
0x8049100 <_start+24> mov    eax,0x804a000 x],al
0x8049105 <_start+29> call   0x804900f <sprint>

native process 2811 In: _start L?? PC: 0x8049100
0x08049No process In: L?? PC: ??
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) si
0x08049100 in _start ()
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 2811) exited normally]
(gdb) 
```

Рис. 2.21: Результат проверки

2.5 Выводы

В ходе лабораторной работы я научилась работать с подпрограммами и отладчиком GDB, а также диагностировать и исправлять ошибки в ассемблерных программах.