

# **Лабораторная работа №9**

**Понятие подпрограммы. Отладчик GDB**

Бунин Арсений Викторович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Самостоятельная работа</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Запуск отладчика . . . . .	8
4.2	Результат работы программы . . . . .	8
4.3	Первая точка останова . . . . .	9
4.4	Дизассемблированный код программы . . . . .	9
4.5	Стиль Интел . . . . .	10
4.6	Псевдографический режим . . . . .	10
4.7	Текст программы . . . . .	11
4.8	Вторая точка останова . . . . .	11
4.9	первая текстовая переменная . . . . .	11
4.10	вторая текстовая переменная . . . . .	12
4.11	Установка значений переменной . . . . .	12
4.12	Запуск второй программы . . . . .	13
4.13	Число переменных . . . . .	13
4.14	Текст программы . . . . .	13
5.1	Код программы . . . . .	14
5.2	Работа программы . . . . .	15
5.3	Неправильная работа программы . . . . .	16
5.4	Нахождение ошибки в отладчике . . . . .	16
5.5	Код программы . . . . .	17
5.6	Неправильная работа программы . . . . .	17
5.7	Нахождение ошибки в отладчике . . . . .	18
5.8	Код программы . . . . .	18
5.9	Корректная работа программы . . . . .	19

## Список таблиц

# 1 Цель работы

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

## 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Выполнить индивидуальное задание по отладке программы на Ассемблере
4. Загрузить файлы на github

### 3 Теоретическое введение

Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия: \* начать выполнение программы, задав всё, что может повлиять на её поведение; \* остановить программу при указанных условиях; \* исследовать, что случилось, когда программа остановилась; \* изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом. Если в программе встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

## 4 Выполнение лабораторной работы

Запустили исполняемый файл в отладчике gdb(рис. 4.1)

```
[arsenii@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb)
```

Рис. 4.1: Запуск отладчика

Проверили работу программы в отладчике(рис. 4.2)

```
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 3023) exited normally]
(gdb) █
```

Рис. 4.2: Результат работы программы

Установили первую точку останова (рис. 4.3).



```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/arsen11/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 4.3: Первая точка останова

Дизассемблирование программы, начиная с точки старта(рис. 4.4).

```

(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. 4.4: Дизассемблированный код программы

Дизассемблированный код программы со стилем Интел(рис. 4.5)

```

(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. 4.5: Стил Intel

Переход в псевдографический режим (рис. 4.6)

```

[ Register Values Unavailable ]

B> 0x08049000 <_start>  mov     eax,0x4
    0x08049005 <_start+5>  mov     ebx,0x1
    0x0804900a <_start+10> mov     ecx,0x804a000
    0x0804900f <_start+15> mov     edx,0x8
    0x08049014 <_start+20> int     0x80
    0x08049016 <_start+22> mov     eax,0x4
    0x0804901b <_start+27> mov     ebx,0x1
    0x08049020 <_start+32> mov     ecx,0x804a008
    0x08049025 <_start+37> mov     edx,0x7
    0x0804902a <_start+42> int     0x80
    0x0804902c <_start+44> mov     eax,0x1
    0x08049031 <_start+49> mov     ebx,0x0
    0x08049036 <_start+54> int     0x80

native process 3084 In: _start
(gdb) layout regs
(gdb)

```

Рис. 4.6: Псевдографический режим

(рис. 4.7)

```
[ Register Values Unavailable ]

b* 0x8049000 <_start> mov $0x4,%eax
0x8049005 <_start+5> mov $0x1,%ebx
0x804900a <_start+10> mov $0x804a000,%ecx
0x804900f <_start+15> mov $0x6,%edx
0x8049014 <_start+20> int $0x80
0x8049016 <_start+22> mov $0x4,%eax
0x804901b <_start+27> mov $0x1,%ebx
0x8049020 <_start+32> mov $0x804a000,%ecx
0x8049025 <_start+37> mov $0x7,%edx
0x804902a <_start+42> int $0x80
0x804902c <_start+44> mov $0x1,%eax
b* 0x8049031 <_start+49> mov $0x7,%ebx
0x8049036 <_start+54> int $0x80

exec No process in:
(gdb) layout regs
(gdb) break 9
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) break 21
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 21.
(gdb) delete breakpoints 2
(gdb) break 20
Breakpoint 3 at 0x8049031: file lab9-2.asm, line 20.
(gdb) █
```

Рис. 4.7: Текст программы

Установили вторую точку останова (рис. 4.8)

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

Рис. 4.8: Вторая точка останова

Вывод значений переменных (рис. 4.9) (рис. 4.10)

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

Рис. 4.9: первая текстовая переменная

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 2
(gdb)
```

Рис. 4.10: вторая текстовая переменная

Установка значений переменных в программе. Вывод переменной отличается из-за формата данных(рис. 4.11).

```
[arsenii@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) layout asm
[arsenii@fedora lab09]$
```

Рис. 4.11: Установка значений переменной

Запуск второй программы с аргументами (рис. 4.12)

```

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) layout asm
[arsenii@fedora lab09]$ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv2+: GNU GPL version 2 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 lab9-3...
(No debugging symbols found in lab9-3)
(gdb) b _start
Breakpoint 1 at 0x00000000
(gdb) ^[[200~gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
Undefined command: "^". Try "help".
(gdb) gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
Undefined command: "gdb". Try "help".
(gdb) run --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
Starting program: /home/arsenii/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/lab9-3 --args lab9-3 аргумент1 аргумент 2 'аргумент 3'

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

Breakpoint 1, 0x00000000 in _start ()
(gdb)

```

Рис. 4.12: Запуск второй программы

Вывод числа аргументов (рис. 4.13)

```

Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xffffcf40: 0x00000007
(gdb)

```

Рис. 4.13: Число переменных

Вывод значений аргументов (рис. 4.14)

```

(gdb) x/s *(void**)(esp + 4)
0xffffd0f1: "/home/arsenii/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd15a: "--args"
(gdb) x/s *(void**)(esp + 12)
0xffffd161: "lab09-3"
(gdb) x/s *(void**)(esp + 16)
0xffffd169: "аргумент1"
(gdb) x/s *(void**)(esp + 20)
0xffffd17b: "аргумент"
(gdb) x/s *(void**)(esp + 24)
0xffffd18c: "2"
(gdb) x/s *(void**)(esp + 28)
0xffffd18e: "аргумент 3"
(gdb)

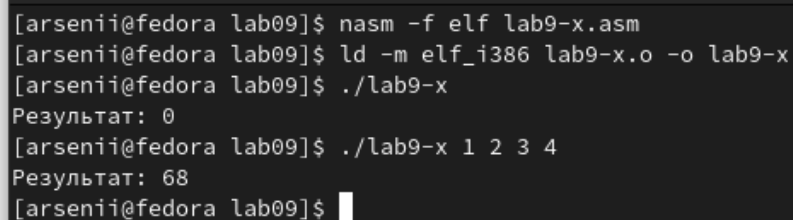
```

Рис. 4.14: Текст программы

## 5 Самостоятельная работа

1. Преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму

Код программы (рис. 5.1)



```
[arsenii@fedora lab09]$ nasm -f elf lab9-x.asm
[arsenii@fedora lab09]$ ld -m elf_i386 lab9-x.o -o lab9-x
[arsenii@fedora lab09]$ ./lab9-x
Результат: 0
[arsenii@fedora lab09]$ ./lab9-x 1 2 3 4
Результат: 68
[arsenii@fedora lab09]$
```

Рис. 5.1: Код программы

Результат работы программы (рис. 5.2)

```

1  %include 'in_out.asm'
2  SECTION .data
3  msg db "Результат: ",0
4  SECTION .text
5  global _start
6  _start:
7  pop ecx ; Извлекаем из стека в ecx количество
8  ; аргументов (первое значение в стеке)
9  pop edx ; Извлекаем из стека в edx имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем ecx на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем esi для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку _end)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число  регистр eax
21 call calc; вызов подпрограммы
22 add esi,eax ;
23 ; след. аргумент
24 loop next ; переход к обработке следующего аргумента
25 _end:
26 mov eax, msg ; вывод сообщения "Результат: "
27 call sprint
28 mov eax, esi ; записываем сумму в регистр eax
29 call iprintLF ; печать результата
30 call quit ; завершение программы
31 ;Расчет функции
32 calc:
33 mov ebx,8 ; EBX=8
34 mul ebx ; EAX=EAX*EBX
35 sub eax,3 ; EAX=EAX-3
36 ret

```

Рис. 5.2: Работа программы

## 2. Отладка программы с помощью GDB Linux

Программа выдает ошибочное значение (рис. 5.3)

```
Breakpoint 1 at 0x80490e8: file lab9-4.asm, line 8.
(gdb) layout asm
[arsenii@fedora lab09]$ ./lab9-4
Результат: 10
[arsenii@fedora lab09]$
```

Рис. 5.3: Неправильная работа программы

В отладчике видим, что умножение происходит в регистре `eax`, тогда как мы положили значение суммы в регистр `ebx`. Это видно по значениям регистров `eax` и `ebx` (рис. 5.4)

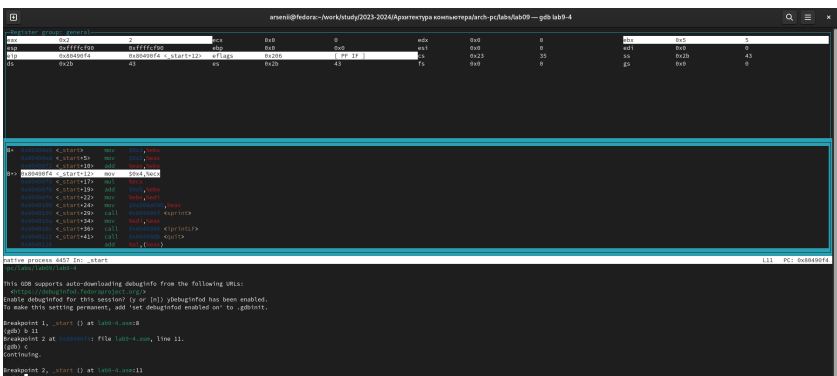


Рис. 5.4: Нахождение ошибки в отладчике

### Исправленный код программы (рис. 5.5)



```

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 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. 5.5: Код программы

Исправленная программа выдает ошибочное значение (рис. 5.3)

```

[arsenii@fedora lab09]$ nasm -f elf -g -l lab9-4.lst lab9-4.asm
[arsenii@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[arsenii@fedora lab09]$ ./lab9-4
Результат: 8

```

Рис. 5.6: Неправильная работа программы

В отладчике видим, что второе сложение происходит в регистре ebx, тогда как мы положили значение суммы в регистр eax (рис. 5.7)

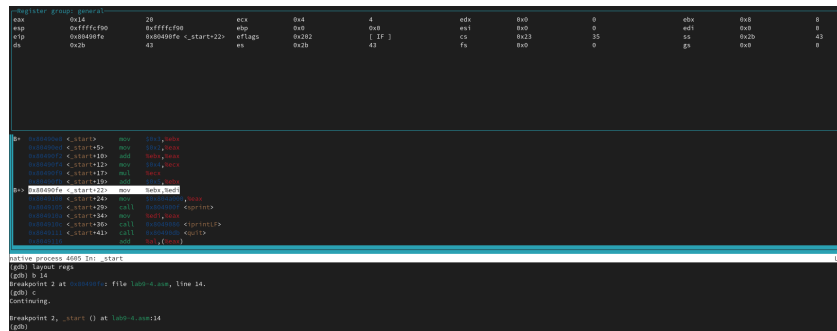


Рис. 5.7: Нахождение ошибки в отладчике

Исправленный код программы (рис. 5.8)

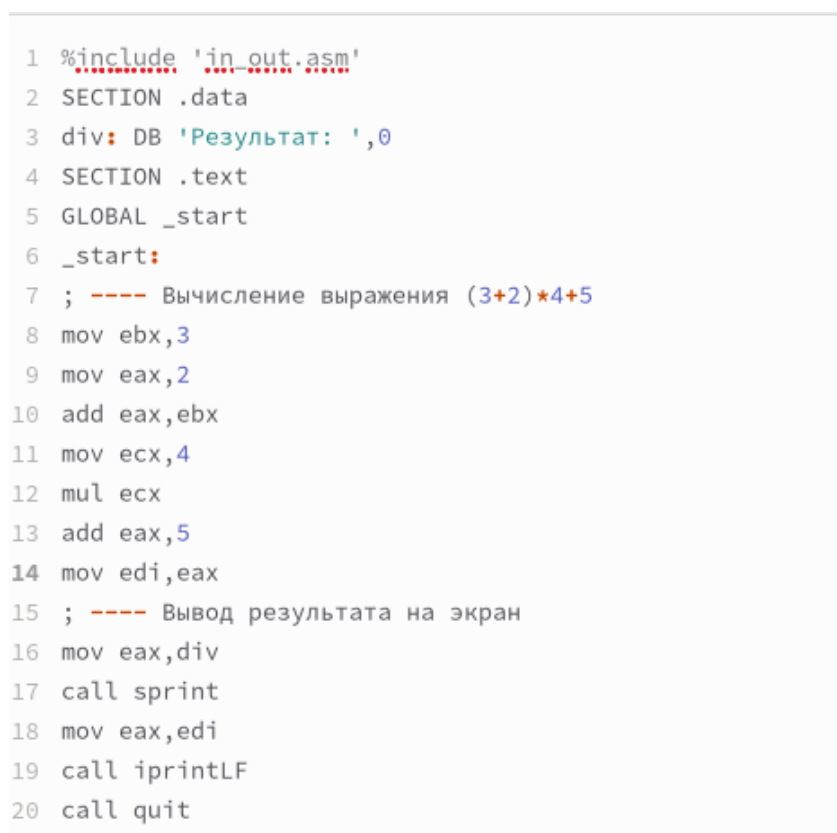


Рис. 5.8: Код программы

Исправленная программа работает корректно (рис. 5.9)

```
(gdb) layout asm
[arsenii@fedora lab09]$ nasm -f elf -g -l lab9-4.lst lab9-4.asm
[arsenii@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[arsenii@fedora lab09]$ ./lab9-4
Результат: 25
[arsenii@fedora lab09]$
```

Рис. 5.9: Корректная работа программы

## 6 Выводы

Приобрели навыки написания программ с использованием подпрограмм. Познакомились с методами отладки при помощи GDB и его основными возможностями.

## **Список литературы**