

# **Отчёт по лабораторной работе №9**

**Архитектура компьютера**

Морозова Мария Вячеславовна

# 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>Выполнение самостоятельной работы</b>	<b>17</b>
<b>6</b>	<b>Выводы</b>	<b>19</b>
<b>7</b>	<b>Листинги</b>	<b>20</b>

## Список иллюстраций

4.1	Создание файла . . . . .	8
4.2	Результат . . . . .	8
4.3	Открываем отладчик . . . . .	9
4.4	Запуск программы . . . . .	9
4.5	Запуск программы с брейкпоинтом . . . . .	9
4.6	Дисассимилированный код программы . . . . .	10
4.7	Отображение команд с Intel'овским синтаксисом . . . . .	11
4.8	Переход в режим псевдографики . . . . .	11
4.9	Команда i b . . . . .	12
4.10	Команда stepi . . . . .	12
4.11	Команда i r . . . . .	12
4.12	Значение переменной msg1 . . . . .	13
4.13	Значение переменной msg2 . . . . .	13
4.14	Изменение msg1 . . . . .	13
4.15	Изменение msg2 . . . . .	14
4.16	Команда set . . . . .	14
4.17	Создание файла lab09-3 . . . . .	15
4.18	Загрузка файла . . . . .	15
4.19	Адрес . . . . .	15
4.20	Остальные позиции стека . . . . .	16
5.1	Запуск программы, результат работы программы . . . . .	17
5.2	Ошибки . . . . .	17
5.3	Результат работы программы . . . . .	18

## **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

Преобразовать программу из лабораторной работы N8, реализовав вычисление значения функции  $f(x)$  как подпрограмму. С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправить ее.

### 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

## 4 Выполнение лабораторной работы

Создала каталог для выполнения лабораторной работы No 9, перешла в него и создала файл lab09-1.asm: (рис. 4.1).

```
mvmorozova@dk8n80 ~ $ mkdir ~/work/arch-pc/lab09
mvmorozova@dk8n80 ~ $ cd ~/work/arch-pc/lab09
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ touch lab09-1.asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 4.1: Создание файла

Запустила файл с изменённым текстом листинга 9.1. (рис. 4.2).

```
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 1
f(g(x))=3
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 4.2: Результат

Провела трансляцию программ с ключом -g, загрузила файл в отладчик. (рис. 4.3).



```

mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ d -m elf_i386 -o lab09-2 lab09-2.o
bash: d: команда не найдена
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
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-pc-linux-gnu".

```

Рис. 4.3: Открываем отладчик

Проверила работу программы, запустив ее в оболочке GDB с помощью команды `run`: (рис. 4.4).

```

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4081) exited normally]
(gdb)

```

Рис. 4.4: Запуск программы

Для более подробного анализа программы установила брейкпоинт на метку `_start`, запустила программу. (рис. 4.5).

```

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4081) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb)

```

Рис. 4.5: Запуск программы с брейкпоинтом

Посмотрела дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`. (рис. 4.6).

```

(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. 4.6: Дисассимилированный код программы

Переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. (рис. 4.7).

```

(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.

```

Рис. 4.7: Отображение команд с Intel'овским синтаксисом

Включила режим псевдографики для более удобного анализа программы. (рис. 4.8).

```

(gdb) layout regs
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab09-2.asm:9
(gdb)

```

Рис. 4.8: Переход в режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверила это с помощью команды info breakpoints (кратко i b): (рис. 4.9).

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08049000 lab09-2.asm:9
2        breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb) █
```

Рис. 4.9: Команда i b

Выполнила 5 инструкций с помощью команды stepi и проследила за изменением значений регистров. (рис. 4.10).

```
the program is not being run.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) █
```

Рис. 4.10: Команда stepi

Посмотрела содержимое регистров с помощью команды info registers. (рис. 4.11).

```
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc310 0xffffc310
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202     [ IF ]
cs       0x23     35
ss       0x2b     43
```

Рис. 4.11: Команда i r

Посмотрела значение переменной msg1 по имени. (рис. 4.12).

```

Type <RET> for more, q to quit, c to continue w
0
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)

```

Рис. 4.12: Значение переменной msg1

Посмотрела значение переменной msg2 по адресу. (рис. 4.13).

```

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

```

Рис. 4.13: Значение переменной msg2

Изменила первый символ переменной msg1. (рис. 4.14).

```

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

```

Рис. 4.14: Изменение msg1

Заменяла символ во второй переменной msg2. (рис. 4.15).

```
(gdb) set {char}0x804a008='L '  
(gdb) set {char}0x804a00b=' '  
(gdb) x/1sb &msg2  
0x804a008 <msg2>: "Lor d!\n\034"  
(gdb) █
```

Рис. 4.15: Изменение msg2

С помощью команды set изменила значение регистра ebx:(рис. 4.16).

```
native process 5127 1  
(gdb) set $ebx='2 '  
(gdb) p/s $ebx  
$4 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$5 = 2  
(gdb) █
```

Рис. 4.16: Команда set

Скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы No8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm: (рис. 4.17).

```

(gdb) layout asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-
pc/lab09/lab09-3.asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $

```

Рис. 4.17: Создание файла lab09-3

Загрузила исполняемый файл в отладчик, указав аргументы: (рис. 4.18).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-3
аргумент1 аргумент 2 аргумент\ 3
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb)

```

Рис. 4.18: Загрузка файла

Адрес вершины стека хранится в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы): (рис. 4.19).

```

5      pop ecx ; Извлекаем из стека в
(gdb) x/x $esp
0xfffffc2c0:      0x00000005
(gdb)

```

Рис. 4.19: Адрес

Посмотрела остальные позиции стека (рис. 4.20).

```
(gdb) x/x $esp
0xffffc2c0:    0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffc55e:    "/afs/.dk.sci.pfu.edu.ru/home/m/v/mvmoro
(gdb) x/s *(void**)(esp + 8)
0xffffc5a5:    "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc5b7:    "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc5c8:    "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc5ca:    "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) █
```

Рис. 4.20: Остальные позиции стека



## 5 Выполнение самостоятельной работы

Преобразовала программу из лабораторной работы No8, реализовав вычисление значения функции  $f(x)$  как подпрограмму. (рис. 5.1).

```
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ./lab09-4
функция: 3(x+2)
результат: 0
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4
функция: 3(x+2)
результат: 204
mvmorozova@dk8n80 ~/work/arch-pc/lab09 $
```

Рис. 5.1: Запуск программы, результат работы программы

С помощью отладчика нашли ошибки в программе. (рис. 5.2).

```
Register group: general
eax    0x2      2
ecx    0x0      0
edx    0x0      0
ebx    0x5      5
esp    0xffffc310 0xffffc310
ebp    0x0      0x0

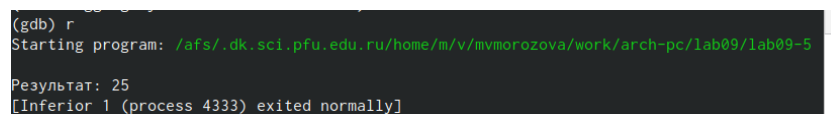
B+ 0x80490e8 <_start>    mov    ebx,0x3
    0x80490ed <_start+5>    mov    eax,0x2
    0x80490f2 <_start+10>   add    ebx,eax
> 0x80490f4 <_start+12>   mov    ecx,0x4
    0x80490f9 <_start+17>   mul    ecx
    0x80490fb <_start+19>   add    ebx,0x5
    0x80490fe <_start+22>   mov    edi,ebx

native process 4098 In: _start L11 PC: 0x80490f4
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:8
(gdb) si
(gdb) si
(gdb) si
```

Рис. 5.2: Ошибки

Проверила работу программы. (рис. 5.3).



```
(gdb) r
Starting program: /afs/.dk.sc1.pfu.edu.ru/home/m/v/mvmorozova/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 4333) exited normally]
```

Рис. 5.3: Результат работы программы

## **6 Выводы**

Были приобретены навыки написания программ с использованием подпрограмм.

## 7 Листинги

```
%include 'in_out.asm'
SECTION .data
f_x db "функция: 3(x+2)",0h
msg db 10,13,'результат: ',0h
SECTION .text
global _start

_f:
push ebx
dec eax
mov ebx, 10
mul ebx
pop ebx
ret

_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
```

```
cmp ecx,0h
jz _end
pop eax
call atoi
call _f
add eax,2
mov ebx,3
mul ebx
add esi, eax
```

```
loop next
```

```
_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintLF
```

```
call quit
```

```
%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
```