

# **Отчёт по лабораторной работе № 9**

**Дисциплина: Архитектура компьютеров**

Хоюгбан Ганчыыр Анатольевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выполнение самостоятельной работы</b>	<b>18</b>
<b>4</b>	<b>Вывод</b>	<b>22</b>

# Список иллюстраций

2.1	Создание файла lab9-1.asm . . . . .	6
2.2	Текст программы файла lab9-1.asm . . . . .	7
2.3	Исполнение программы файла lab9-1.asm . . . . .	8
2.4	Текст программы файла lab9-2.asm . . . . .	9
2.5	Исполнение программы файла lab9-2.asm . . . . .	10
2.6	Текст программы для сообщения Hello world! . . . . .	11
2.7	Открытие отладчика gdb . . . . .	12
2.8	Команда run . . . . .	12
2.9	Установка брейкпоинга . . . . .	12
2.10	Дисассимилированный код . . . . .	13
2.11	Переключение на синтаксис Intel . . . . .	13
2.12	Режим псевдографики . . . . .	14
2.13	Точка останова . . . . .	14
2.14	Информация о точках останова . . . . .	14
2.15	Значение msg2 и изменение первого символа msg1 . . . . .	15
2.16	Изменение первого символа msg2 . . . . .	15
2.17	Команды print . . . . .	15
2.18	Изменение значения регистра . . . . .	16
2.19	Копирование файла и создание исполняемого . . . . .	16
2.20	Запуск программы в оболочке отладки . . . . .	16
2.21	Количество аргументов 4 и просмотр позиций стека . . . . .	17
3.1	Текст программы файла из 8 лабораторной с подпрограммами . . .	19
3.2	Исполнение файла из 8 лабораторной с подпрограммами . . . . .	20
3.3	Исправленный текст программы из 8 лабораторно . . . . .	20
3.4	Исполнение исправленного текста программы из 8 лабораторной	21

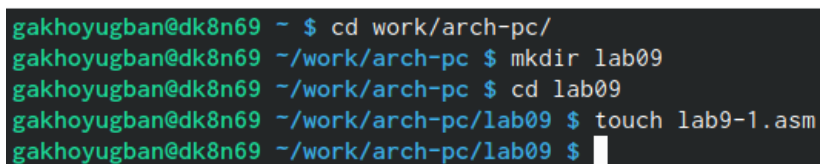
## **Список таблиц**

# 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

Для начала открыл терминал, перехожу на каталог, через которую буду выполнять 9 лабораторную, а затем создаю файл lab9-1.asm(рис. 2.1)

A screenshot of a terminal window with a dark background and light-colored text. It shows a series of five commands being entered at a prompt. The first command is 'cd work/arch-pc/'. The second is 'mkdir lab09'. The third is 'cd lab09'. The fourth is 'touch lab9-1.asm'. The fifth command is partially visible as 'touch lab9-1.asm' followed by a cursor. The prompt changes from '~' to '~/work/arch-pc' and then to '~/work/arch-pc/lab09' as the user navigates through the directory structure.

```
gakhoyugban@dk8n69 ~ $ cd work/arch-pc/  
gakhoyugban@dk8n69 ~/work/arch-pc $ mkdir lab09  
gakhoyugban@dk8n69 ~/work/arch-pc $ cd lab09  
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ touch lab9-1.asm  
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Создание файла lab9-1.asm

Написал текст программы с использованием вызова подпрограммы, что я демонстрирую вам на рисунке(рис. 2.2)

```

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы

```

Рис. 2.2: Текст программы файла lab9-1.asm

Перевел файл lab9-1.asm в объектный, сделал компоновку и отправил на исполнение, куда в итоге ввел значения 1 и 2(рис. 2.3)

```
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ nasm -f elf lab9-2.asm
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ./lab9-2
Введите x: 1
2x+7=9
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ./lab9-2
Введите x: 2
2x+7=11
```

Рис. 2.3: Исполнение программы файла lab9-1.asm

Изменил текст программы, как требует лабораторная, для вновь созданного файла lab9-2.asm, что показываю на рисунке (рис. 2.4)



```

%include "in_out.asm"

SECTION .data
msg: DB 'Введите x: ', 0
result: DB 'f(g(x))=', 0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

call _subcalcul
call _calcul

mov eax, result
call sprint
mov eax, [res]
call iprintLF

call quit

_calcul:
push ebx
mov ebx, 2
mul ebx

add eax, 7

pop ebx
ret

_subcalcul:
push ebx
mov ebx, 3
mul ebx
dec ebx
mov [res], eax

pop ebx
ret

```

Рис. 2.4: Текст программы файла lab9-2.asm

Перевел файл lab9-2.asm в объектный, сделал компоновку и отправил на исполнение, куда в итоге ввел значения 1 и 2(рис. 2.5)

```
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ nasm -f elf lab9-2.asm
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ./lab9-2
Введите x: 1
f(g(x))=3
gakhoyugban@dk8n69 ~/work/arch-pc/lab09 $ ./lab9-2
Введите x: 2
f(g(x))=6
```

Рис. 2.5: Исполнение программы файла lab9-2.asm

Напечатал текст программы для вызова сообщения Hello world!, что показываю на рисунке(рис. 2.6)

```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 2.6: Текст программы для сообщения Hello world!

Перевел файл lab9-3.asm в объектный, сделал компоновку и отправил на исполнение, где открылся отладчик gdb(рис. 2.7)

```
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.asm
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ gdb lab9-3
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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) █
```

Рис. 2.7: Открытие отладчика gdb

Проверил работу программы, запустив ее в оболочке gdb с помощью команды run(рис. 2.8)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gakhoyugban/work/arch-pc/lab09/lab9-3
Hello, world!
[Inferior 1 (process 4931) exited normally]
(gdb) █
```

Рис. 2.8: Команда run

Установил брейкпоинг а метку \_start, с которой начинается выполнение любой ассемблерной программы, а затем запустил ее(рис. 2.9)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-3.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gakhoyugban/work/arch-pc/lab09/lab9-3
Breakpoint 1, _start () at lab9-3.asm:9
(gdb) █
```

Рис. 2.9: Установка брейкпоинга

Посмотрел дисассимилированный код программы с помощью disassemble(рис. 2.10)

```

(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.

```

Рис. 2.10: Дисассимилированный код

Переключился на отображение команд с Intelовским синтаксисом, введя команду set(рис. 2.11)

```

(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. 2.11: Переключение на синтаксис Intel

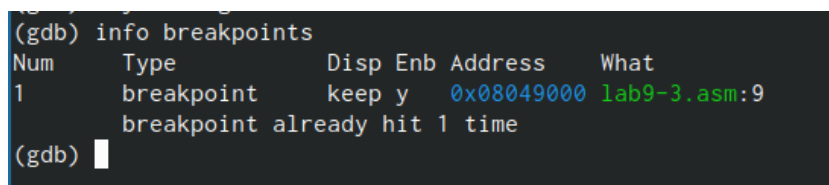
Включил режим псевдографики для удобного анализа программы(рис. 2.12)



```
[ Register Values Unavailable ]
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 4961 In: _start          L9      PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 2.12: Режим псевдографики

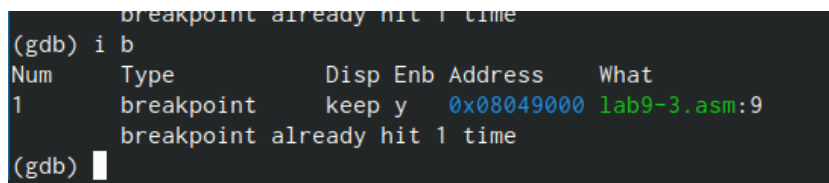
Проверяю точку останова по имени \_start(рис. 2.13)



```
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint       keep y   0x08049000  lab9-3.asm:9
breakpoint already hit 1 time
(gdb)
```

Рис. 2.13: Точка останова

Смотрю информацию о всех установленных точках останова с помощью команды i b (рис. 2.14)



```
breakpoint already hit 1 time
(gdb) i b
Num      Type             Disp Enb Address      What
1        breakpoint       keep y   0x08049000  lab9-3.asm:9
breakpoint already hit 1 time
(gdb)
```

Рис. 2.14: Информация о точках останова

Посмотрел значения переменной msg2 по адресу, а затем изменяю первый символ в переменной msg1(рис. 2.15)

```

0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg="h"
A syntax error in expression, near `]&msg="h"'
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb)

```

Рис. 2.15: Значение msg2 и изменение первого символа msg1

Изменил значение первого символа переменной msg2(рис. 2.16)

```

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

```

Рис. 2.16: Изменение первого символа msg2

Примеры использования команды print(рис. 2.17)

```

(gdb) p/s $edx
$2 = 8
(gdb) p/x
$3 = 0x8
(gdb) p/t
$4 = 1000
(gdb)

```

Рис. 2.17: Команды print

С помощью команды set изменяю значение регистра ebx. Разница вывода из-за того что в первом случае 2 это символ а во втором число(рис. 2.18)

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)

```

Рис. 2.18: Изменение значения регистра

Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, в файл lab9-4.asm (рис. 2.19)

```

gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-4.lst lab9-4.asm
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ gdb --args lab9-4 arg arg2 'arg3'
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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab9-4...
(gdb)

```

Рис. 2.19: Копирование файла и создание исполняемого

Запускаю программу в оболочке gdb (рис. 2.20)

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-4.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gakhoyugban/work/arch-pc/lab09/lab9-4 arg arg2 arg3

Breakpoint 1, _start () at lab9-4.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb)

```

Рис. 2.20: Запуск программы в оболочке отладки



Узнаю количество аргументов, а затем смотрю все позиции стека. Их адреса располагаются в 4 байтах друг от друга(рис. 2.21)

```
(gdb) x/x $esp
0xffffc2f0: 0x00000004
(gdb) x/s *(void**)(esp + 4)
0xffffc58e: "/afs/.dk.sci.pfu.edu.ru/home/g/a/gakhoyugban/work/arch-pc/lab09/lab9-4"
(gdb) x/s *(void**)(esp + 8)
0xffffc5d5: "arg"
(gdb) *(void**)(esp + 12)
Undefined command: "". Try "help".
(gdb) x/s *(void**)(esp + 12)
0xffffc5d9: "arg2"
(gdb) x/s *(void**)(esp + 16)
0xffffc5de: "arg3"
(gdb) x/s *(void**)(esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.21: Количество аргументов 4 и просмотр позиций стека

### **3 Выполнение самостоятельной работы**

Для выполнение самостоятельной работы беру текст файла из лабораторной №8, но с использованием подпрограмм. Текст программы демонстрирую на рисунке(рис. 3.1)

```

%include 'in_out.asm'

SECTION .data
f_x db "функция: f(x)=3(10 + x)",0h
msg db 10,13,'результат: ',0h

SECTION .text
global _start

_f:
push ebx
add eax, 10
mov ebx, 3
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 esi, eax

loop next

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

```

Рис. 3.1: Текст программы файла из 8 лабораторной с подпрограммами

Перевел файл lab9-5.asm в объектный, сделал компоновку и отправил на исполнение, куда в итоге ввел аргументы 1 2 3 4(рис. 3.2)

```
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ./lab9-5
функция: f(x)=3(10 + x)
результат: 0
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ./lab9-5 1 2 3 4
функция: f(x)=3(10 + x)
результат: 150
```

Рис. 3.2: Исполнение файла из 8 лабораторной с подпрограммами

Затем я отредактировал файл, нашел некоторые несостыковки при использовании отладчика и нашел ошибки в строках, а именно: add ebx, eax mov ecx, 4 mul ecx add ebx, 5 mov edi, ebx Я исправил текст программы и демонстрирую вам исправленный код(рис. 3.3)

```
%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
```

Рис. 3.3: Исправленный текст программы из 8 лабораторно

Перевел файл lab9-6.asm в объектный, сделал компоновку и отправил на исполнение, откуда получил в результате 25(рис. 3.4)

```
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ nasm -f elf lab9-6.asm
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-6 lab9-6.o
gakhoyugban@dk8n53 ~/work/arch-pc/lab09 $ ./lab9-6
Результат: 25
```

Рис. 3.4: Исполнение исправленного текста программы из 8 лабораторной

## 4 Вывод

В результате выполнения лабораторной работы, я научился организовывать код в подпрограммы и познакомился с базовыми функциями отладчика gdb