

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

#### «Понятие подпрограммы. Отладчик GDB»

*дисциплина:* Архитектура компьютера

Студент: Бахтиярова Алиса

Группа: НКАбд-01-24

МОСКВА

2024 г.

## Содержание

Цель работы .....	2
Основная часть.....	2
Задание для самостоятельной работы: .....	7
Вывод: я изучила как работать с отладчиком, применила знания на практике. ....	8

### Цель работы

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

### Основная часть

Для работы над этой лабораторной работой создаю текстовый каталог и текстовый файл. Записываю в него текст первого листинга.

```
bakhtiyarovaan@vbox:~$ mkdir ~/work/arch-pc/lab09
bakhtiyarovaan@vbox:~$ cd ~/work/arch-pc/lab09
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ touch lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-1.asm
```

Рис.1

Запускаю объектный и исполняемый файлы и запускаю программу для вычисления уравнения. Все работает верно.

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

Рис.2

Далее моим заданием было исправить код и добавить в него подфункцию. Несколько раз в программе вылезает ошибка.

```

bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
Ошибка сегментирования (образ памяти сброшен на диск)
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
Ошибка сегментирования (образ памяти сброшен на диск)
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
Ошибка сегментирования (образ памяти сброшен на диск)
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ █

```

Рис.3

Создаю еще один текстовый файл. Записываю в него текст листинга 2. Для того, чтобы работать с отладчиком, запускаю gdb.

```

bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ touch lab09-2.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gedit lab09-2.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora 10.0) 7.4.1-5.fc10
Copyright (C) 2009 Free Software Foundation, Inc.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show" for debugging configuration details.
(gdb)

```

Рис.4

Запускаю работу программы в оболочке gdb.

```

(gdb) run
Starting program: /home/bakhtiyarovaan/work/arch-pc/lab09/lab09-2

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 10225) exited normally]
(gdb) █

```

Рис.5

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запускаю ее

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/bakhtiyarovaan/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) █
```

Рис.6

Смотрю дисассимилированный код с помощью программы.

```
(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.7

Переключаюсь на отображение команд с Intel'овским синтаксисом. Сравнив , делаю вывод

Intel: порядок операндов — цель, потом источник; регистры без знака процента; размеры данных определяются инструкцией; немедленные значения без символов; комментарии в конце строки.

AT&T: порядок операндов — источник, потом цель; регистры с знаком процента; размеры данных указываются суффиксами; немедленные значения с долларом; комментарии начинаются с #.

```

bakhtiyarovaan@vbox:~/work/arch-pc/lab09
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
0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov     eax,0x1
0x8049031 <_start+49>   mov     ebx,0x0
0x8049036 <_start+54>   int     0x80

native process 10321 In: _start          L9      PC: 0x8049000
(gdb)

```

Рис.8

Ставлю точки останова по адресам инструкции.

```

bakhtiyarovaan@vbox:~/work/arch-pc/lab09
[ Register Values Unavailable ]

0x8049020 <_start+32>   mov     ecx,0x804a008
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov     eax,0x1
b+ 0x8049031 <_start+49>   mov     ebx,0x0
0x8049036 <_start+54>   int     0x80

native process 10865 In: _start          L9      PC: 0x8049000
(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
          breakpoint already hit 1 time
2        breakpoint    keep y  0x08049031 lab09-2.asm:20
(gdb)

```

Рис.9

С помощью только что изученной команды смотрю содержимое переменной.



The screenshot shows a GDB terminal window with the title bar "bakhtiyarovaan@vbox:~/work/arch-pc/lab09". The main display area shows a list of assembly instructions with their addresses and disassembled forms:

```
[ Register Values Unavailable ]  
  
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
```

Below the instructions, the GDB status bar shows "native process 10865 In: \_start L9 PC: 0x8049000". The command prompt shows the following commands and their outputs:

```
es          0x2b          43  
--Type <RET> for more, q to quit, c to continue without paging--q  
Quit  
(gdb) x/1sb &msg1  
0x804a000 <msg1>:      "Hello, "  
(gdb) x/1sb 0x804a008  
0x804a008 <msg2>:      "world!\n\034"  
(gdb) 
```

Рис.10

Заменяю первые буквы в инструкциях.

The screenshot shows a GDB terminal window with the following commands and outputs:

```
(gdb) set {char}0x804a001='h'  
(gdb) x/1sb &msg1  
0x804a000 <msg1>:      "hhlllo, "  
(gdb) set {char}0x804a008='L'  
(gdb) set {char}0x804a00b=' '  
(gdb) x/1sb &msg2  
0x804a008 <msg2>:      "Lor d!\n\034"  
(gdb) 
```

Рис.11

Пробую изменить значение регистра

```
0x8049025 <_start+37> int    0x80  7
0x804902a <_start+42> mov    eax,0x4
0x804902c <_start+44> mov    ebx,0x1
B->0x8049031 <_start+49> mov    ebx,0x014a008
0x8049036 <_start+54> int    0x80
0x8049038 <_start+56> add    BYTE PTR [eax],al
native process 10865 In: _start L9 PC: 0x8049000
(gdb) x/1sb &msg2 20 31
(gdb) p/s $ebx
$1 = 50
(gdb) c
Continuing.
hhllo, Lor d!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb)
```

Рис.12

### Задание для самостоятельной работы:

2) Записываю текст листинга в файл, создаю объектный и исполнительный файлы. Проверю программу с помощью отладчика.

```
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ ./lab09-3
Результат: 25
bakhtiyarovaan@vbox:~/work/arch-pc/lab09$ gdb ./lab09-3
GNU gdb (Fedora Linux) 14.2-1.fc40
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/>.
```

Рис.13

Далее по шажочку, после точки старта проверяю строчки кода программы. Отладчик не выдал мне ни одной ошибки.

```

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./lab09-3...
(No debugging symbols found in ./lab09-3)
(gdb) break _start
Breakpoint 1 at 0x80490e8:
(gdb) run
Starting program: /home/bakhtiyarovaan/work/arch-pc/lab09/lab09-3

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, 0x080490e8 in _start ()
(gdb) step
Single stepping until exit from function _start,
which has no line number information.
0x0804900f in sprint ()

```

```

Dump of assembler code for function _start:
   0x080490e8 <+0>:    mov     $0x3,%ebx
   0x080490ed <+5>:    add     $0x2,%ebx
   0x080490f0 <+8>:    mov     %ebx,%eax
   0x080490f2 <+10>:   mov     $0x4,%ecx
   0x080490f7 <+15>:   mul     %ecx
   0x080490f9 <+17>:   add     $0x5,%eax
   0x080490fc <+20>:   mov     %eax,%edi
   0x080490fe <+22>:   mov     $0x804a000,%eax
   0x08049103 <+27>:   call    0x804900f <sprint>
   0x08049108 <+32>:   mov     %edi,%eax
   0x0804910a <+34>:   call    0x8049086 <iprintLF>
   0x0804910f <+39>:   call    0x80490db <quit>
End of assembler dump.

```

Рис.14

Программа проверена, при запуске выдает результат вычисления, 25. Все верно

Вывод: я изучила как работать с отладчиком, применила знания на практике.