

Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Корчаги Алексей Павлович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	17
5	Выводы	24

Список иллюстраций

3.1	Создание файла	7
3.2	Редактирование файла	8
3.3	Работа программы	8
3.4	Создание файла	9
3.5	Редактирование файла	10
3.6	Работа файла	11
3.7	Работа с gdb	11
3.8	Установка breakpoint	11
3.9	Дизассемблированный код	12
3.10	Содержание регистров	12
3.11	Команда i b	13
3.12	Взаимодействие с адресом	13
3.13	Значение регистров	13
3.14	Значение переменной msg1	13
3.15	Изменение значения msg1	14
3.16	Изменение значения msg2	14
3.17	Значение регистра	14
3.18	Присваивание значения регистру	14
3.19	Копирование файла	15
3.20	Значение esp	15
3.21	Элементы стека	16
4.1	Редактирование файла	17
4.2	Создание исполняемого файла	18
4.3	Программа из листинга	20
4.4	Создание исполняемого файла	20
4.5	Отладчик	21
4.6	Первая точка остановки	21
4.7	Вторая точка остановки	21
4.8	Третья точка остановки	22
4.9	Редактирование файла	22
4.10	Создание исполняемого файла	23

Список таблиц

1 Цель работы

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

2 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на три этапа: • поиск ошибки; • определение причины ошибки; • исправление ошибки.

GDB — отладчик проекта GNU работает на большинстве UNIX-подобных системах и может проводить процесс отладки для кода написанного на многих языках программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки. 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

3 Выполнение лабораторной работы

Создал директорию и файл для выполнения лабораторной работы(рис. 3.1).

```
apkorchagin@dk5n59 ~ $ mkdir ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~ $ cd ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ touch lab9-1.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $
```

Рис. 3.1: Создание файла

Записал код в файл lab9-1.asm, дописал доп функцию для того чтобы программа работала как сложная функции(рис. 3.2).

```

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. 3.2: Редактирование файла

Проверил работу программы(рис. 3.3).

```

apkorchagin@dk4n65 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm && ld -m elf_i386 -o lab9-1
lab9-1.o && ./lab9-1
Введите x: 1
f(g(x))=3
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $

```

Рис. 3.3: Работа программы

Создал файл lab9-2.asm(рис. 3.4).


```
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ touch lab9-2.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $
```

Рис. 3.4: Создание файла

Записал код в файл(рис. 3.5).

```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6 SECTION .text
7 global _start
8 _start:
9 mov eax, 4
10 mov ebx, 1
11 mov ecx, msg1
12 mov edx, msg1Len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2Len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 3.5: Редактирование файла

Ассемблировал файл lab9-2.asm, чтобы открыть файл через gdb(рис. 3.6).

```

apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
ld: невозможно найти lab09-2.o: Нет такого файла или каталога
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb lab9-2

```

Рис. 3.6: Работа файла

Открыл файл lab9-2.asm через gdb(рис. 3.7).

```

apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb lab9-2
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".
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 lab9-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/
Hello, world!
[Inferior 1 (process 8537) exited normally]

```

Рис. 3.7: Работа с gdb

Поставил breakpoint на _start(рис. 3.8).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-2

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

```

Рис. 3.8: Установка breakpoint

Начал просмотр дизассемблированного кода с синтаксисом intel(рис. 3.9).

```

(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. 3.9: Дизассемблированный код

В синтаксисе АТТ в виде 16-ти ричного числа записаны первые аргументы всех команд, в синтаксисе Intel так записываются адреса вторых аргументов. запустим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. 3.10).



```

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
native process 8540 In: _start          L9      PC: 0x08049000
(gdb) layout regs
(gdb)

```

Рис. 3.10: Содержимое регистров

С помощью `i b` посмотрим информацию о точках остановки(рис. 3.11).

```
(gdb) layout regs
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb) □
```

Рис. 3.11: Команда i b

Добавил ещё одну точку останова по адресу(рис. 3.12).

```
(gdb) b *0x8049036
Breakpoint 2 at 0x8049036: file lab9-2.asm, line 21.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
2        breakpoint keep y  0x08049036 lab9-2.asm:21
(gdb) □
```

Рис. 3.12: Взаимодействие с адресом

С помощью команды i r вывел значение регистров(рис. 3.13).

```
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc480 0xffffc480
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--□
```

Рис. 3.13: Значение регистров

Вывел значение переменной по имени(рис. 3.14).

```
(gdb) x/lb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) □
```

Рис. 3.14: Значение переменной msg1

Изменил значение переменной msg1(рис. 3.15).

```

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

```

Рис. 3.15: Изменение значения msg1

Изменил значение переменной msg2(рис. 3.16).

```

(gdb) set {char}&msg2=8
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "\borld!\n\034"
(gdb) 

```

Рис. 3.16: Изменение значения msg2

Вывел значение регистра ebx, в трёх разных форматах: в строчном(p/s); в 16-ричном(p/x); в двоичном(p/t)(рис. 3.17).

```

(gdb) p/s $edx
$1 = 0
(gdb) p/x
$2 = 0x0
(gdb) p/t
$3 = 0
(gdb) 

```

Рис. 3.17: Значение регистра

Задал значение регистру, при попытке задания регистру строкового значения возникает ошибка(рис. 3.18).

```

(gdb) set $ebx="2"
evaluation of this expression requires the program to have a function "malloc".
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb) 

```

Рис. 3.18: Присваивание значения регистру

Скопировал из 8 лабораторной работы файл переименовал и создал исполняемый файл. Открыл файл создал точку остановки(рис. 3.19).

```
apkorchagin@dk5n59 ~ $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
apkorchagin@dk5n59 ~ $ cd ~/work/arch-pc/lab09
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
apkorchagin@dk5n59 ~/work/arch-pc/lab09 $ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 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) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-3 аргумент1 арг
умент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
.    .    .
```

Рис. 3.19: Копирование файла

Посмотрел на содержимое того, что находится по адресу регистра esp(рис. 3.20).

```
(gdb) x/x $esp
0xffffc440:      0x00000005
(gdb) □
```

Рис. 3.20: Значение esp

Посмотрел на все остальные элементы стека. Их адреса находятся на расстоянии 4 байта, именно столько занимает один элемент стека(рис. 3.21).

```

(gdb) x/s *(void**)(esp + 8)
0xffffc6ec:    "апрымент1"
(gdb) x/s *(void**)(esp + 4)
0xffffc6a5:    "/afs/.dk.sci.pfu.edu.ru/home/a/p/apkorchagin/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 12)
0xffffc6fe:    "апрымент"
(gdb) x/s *(void**)(esp + 16)
0xffffc70f:    "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc711:    "апрымент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) □

```

Рис. 3.21: Элементы стека

4 Выполнение самостоятельной работы

Скопировал и переименовал файл, переписал программу с использованием подпрограмм(рис. ??).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 msg1 db "Функция: f(x)=3x-1"
5 SECTION .text
6 global _start
7 _start:
8 pop ecx
9 pop edx
10 sub ecx,1
11 mov esi, 0
12 next:
13 cmp ecx,0h
14 jz _end ; если аргументов нет выходим из цикла
15 ; (переход на метку '_end')
16 pop eax
17 call atoi
18 call _calcul ; вызов подпрограммы
19 ; след. аргумент 'esi=esi+eax'
20 loop next
21 _end:
22 mov eax,msg1
23 call sprintf
24 mov eax, msg
25 call sprintf
26 mov eax, esi
27 call sprintf
28 call quit
29 ;-----Код подпрограммы
30 _calcul:
31 mov ebx,3 ;ebx=3
32 mul ebx; eax=eax*ebx
33 sub eax,1 ; eax-1
34 add esi,eax
35
36 ;----возврат в основную программу
37
38 ret
```

Рис. 4.1: Редактирование файла

Создал исполняемый файл и проверил работу изменённой программы(рис. 4.2).

```

apkorchagin@dk4n65 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm && ld -m elf_i386 -o lab9-4
lab9-4.o && ./lab9-4 1 2 3 4 5
Функция: f(x)=3x-1
Результат: 40
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $

```

Рис. 4.2: Создание исполняемого файла

Код программы задания 1:

```

#include 'in_out.asm'

SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=3x-1"

SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax
call atoi
call _calcul ; вызов подпрограммы
; след. аргумент `esi=esi+eax`
loop next
_end:
mov eax,msg1
call sprintLF

```

```

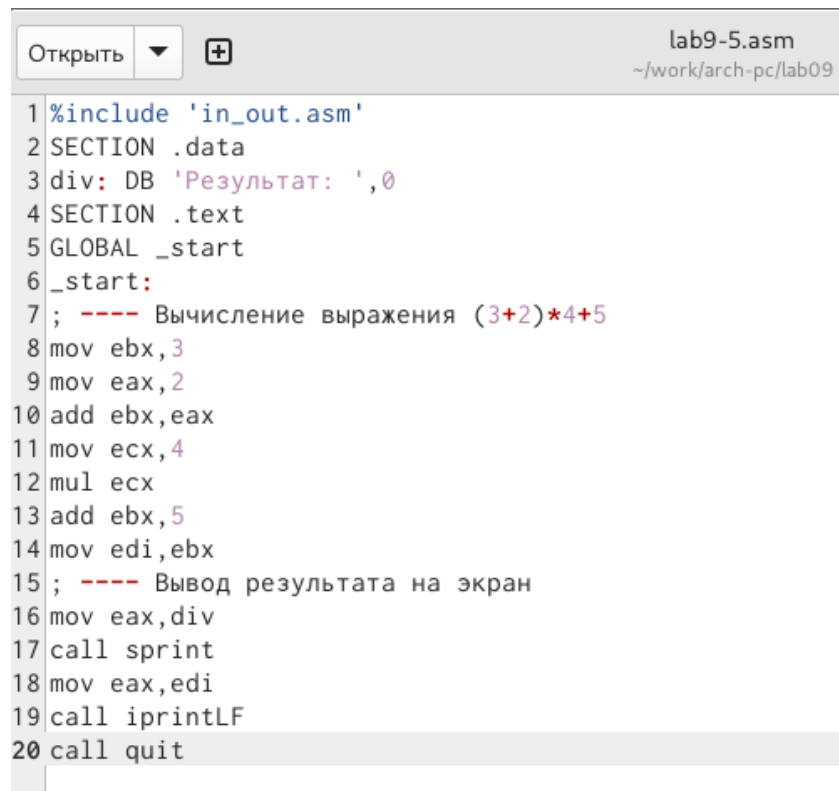
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
;-----Код подпрограммы
_calcul:
mov ebx,3 ;ebx=3
mul ebx; eax=eax*ebx
sub eax,1 ; eax-1
add esi,eax

;----возврат в основную программу

ret

```

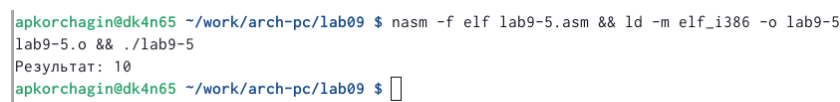
Создал файл lab9-5.asm и ввёл в него код из листинга(рис. 4.3).



```
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. 4.3: Программа из листинга

Создал исполняемый файл и запустил программу, программа выводит не верный ответ(рис. 4.4).



```
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm && ld -m elf_i386 -o lab9-5
lab9-5.o && ./lab9-5
Результат: 10
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $
```

Рис. 4.4: Создание исполняемого файла

Открыл программу в отладчике. Для того, чтобы найти ошибку, смотрю дисассемблирую программу и добавляю брейкпоинты в основные части программы (рис. 4.5).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:    mov     ebx,0x3
    0x080490ed <+5>:    mov     eax,0x2
    0x080490f2 <+10>:   add     ebx,eax
    0x080490f4 <+12>:   mov     ecx,0x4
    0x080490f9 <+17>:   mul     ecx
    0x080490fb <+19>:   add     ebx,0x5
    0x080490fe <+22>:   mov     edi,ebx
    0x08049100 <+24>:   mov     eax,0x804a000
    0x08049105 <+29>:   call    0x804900f <sprint>
    0x0804910a <+34>:   mov     eax,edi
    0x0804910c <+36>:   call    0x8049086 <iprintLF>
    0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb) □
```

Рис. 4.5: Отладчик

Перешёл к первому брекпоинту, в точке `_start`, ошибок не обнаружил(рис. 4.6).

```
Breakpoint 1, _start () at lab9-5.asm:8
8      mov ebx,3
(gdb) p/s $ebx
$1 = 0
(gdb) p/s $eax
$2 = 0
(gdb) □
```

Рис. 4.6: Первая точка остановки

Перешёл к второй точке остановки, заметил, что результат суммирования записывается в `ebx` (рис. 4.7).

```
Breakpoint 2, _start () at lab9-5.asm:11
11     mov ecx,4
(gdb) p/s $ebx
$3 = 5
(gdb) p/s $eax
$4 = 2
(gdb) □
```

Рис. 4.7: Вторая точка остановки

Перешёл к третьему брекпоинту, проверил значение регистров, заметил что умножение значения регистра `ecx` происходит на регистр `eax`($4*2$), а к регистру `ebx` прибавляется 5 ($5+5$) и его значение записывается в результат программы. (рис. 4.8).

```

Breakpoint 3, _start () at lab9-5.asm:14
14      mov edi,ebx
(gdb) p/s $ebx
$5 = 10
(gdb) p/s $eax
$6 = 8
(gdb) p/s $ecx
$7 = 4
(gdb) 

```

Рис. 4.8: Третья точка остановки

Исправил ошибки, которые нашёл благодаря GDB(рис. 4.9).

```

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. 4.9: Редактирование файла

Создал исполняемый файл и проверил программу, теперь программа выводит корректное значение(рис. 4.10).

```
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-5.lst lab9-5.asm && ld -m elf_i386 -o lab9-5 lab9-5.o && ./lab9-5
Результат: 25
apkorchagin@dk4n65 ~/work/arch-pc/lab09 $
```

Рис. 4.10: Создание исполняемого файла

Код программы задания 2:

```
%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
```

5 Выводы

В ходе выполнения работы я приобрёл навыки написания программ с использованием подпрограмм и познакомился с методами отладки при помощи GDB.