

Отчет по выполнению лабораторной работы №9

Дисциплина: архитектура компьютеров

Новиков Никита Владимирович

Содержание

1 Цель работы

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

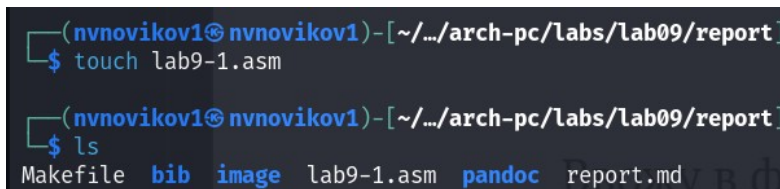
2 Задание

1. Релизация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Задание для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM

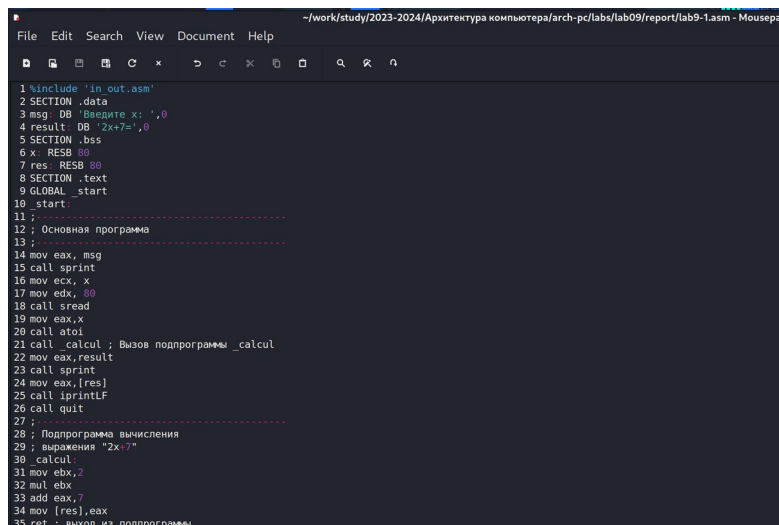
Создаю каталог для выполнения лабораторной работы №9, перехожу в него и создаю файл lab09-1.asm. (рис. [??]).



```
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]  
$ touch lab9-1.asm  
  
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]  
$ ls  
Makefile  bib  image  lab9-1.asm  pandoc  report.md
```

Создание каталога и файла

Ввожу в файл lab09-1.asm текст программы из листинга 9.1.(рис. [??])



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB "Введите x: ",0
4 result: DB "2x+7=",0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; -----
12 ; Основная программа
13 ; -----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul: Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call sprintf
26 call quit
27 ; -----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx, 2
32 mul ebx
33 add eax, 7
34 mov [res], eax
35 ret ; выход из подпрограммы
```

Ввод программы из листинга 9.1

Создаю исполняемый файл и проверяю его работу.(рис. [??])



```
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ nasm -f elf lab9-1.asm

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ./lab9-1
Введите x: 10
2x+7=27
```

Проверка работы программы

Изменяю текст программы, добавляя подпрограмму _subcalcul в подпрограммы _calcul.(рис. [??])

```
File Edit Search View Document Help
1 %include 'in.out.asm'
2 SECTION .data
3 msg: DB "Введите x: ",0
4 result: DB "2(3x-1)+7=",0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; -----
12 ; Основная программа
13 ; -----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintLF
26 call quit
27 ; -----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 calcul:
31 call subcalcul
32 mov ebx, 2
33 mul ebx
34 add eax, 7
35 mov [res], eax
36 ret ; выход из подпрограммы
37 subcalcul:
38 mov ebx, 3
39 mul ebx
40 sub eax, 1
41 ret
```

Изменение текста программы

Создаю исполняемый файл и запускаю его.(рис. [??])

```
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ nasm -f elf lab9-1.asm

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ./lab9-1
Введите x: 10
2(3x-1)+7=65
```

Запуск программы

Программа работает корректно.

3.2 Отладка программ с помощью GDB

Создаю файл lab09-2.asm с текстом программы из Листинга 9.2.(рис. [??])

```
File Edit Search View Document Help
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
```

Создание файла печати сообщения Hello world!

Получаю исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. Далее загружаю исполняемый файл в отладчик gdb и запускаю его с помощью команды run. (рис. [??])

```
(nvnovikov1@nvnovikov1)~/arch-pc/labs/lab09/report
$ nasm -f elf -g -l lab9-2.lst lab9-2.asm

(nvnovikov1@nvnovikov1)~/arch-pc/labs/lab09/report
$ ld -m elf_i386 -o lab9-2 lab9-2.o

(nvnovikov1@nvnovikov1)~/arch-pc/labs/lab09/report
$ gdb lab9-2
GNU gdb (Debian 13.2-1) 13.2
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-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) Hello world
Undefined command: "Hello". Try "help".
(gdb) r
Starting program: /home/nvnovikov1/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/report/lab9-2
Hello, world!
[Inferior 1 (process 466634) exited normally]
(gdb)
```

Получение и загрузка в отладчик исполняемого файла

Устанавливаю брейкпоинт на метку _start и запускаю программу. (рис. [??])

```
(gdb) break _start
Breakpoint 1 at 0x0049000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /home/nvnovikov1/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/report/lab9-2

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

Установка брейкпоинта

Смотрю дисассемблированный код программы с помощью команды disassemble начинаю с метки _start. (рис. [??])

```

(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) █

```

Просмотр дисассимилированного кода программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel.(рис. [??])

```

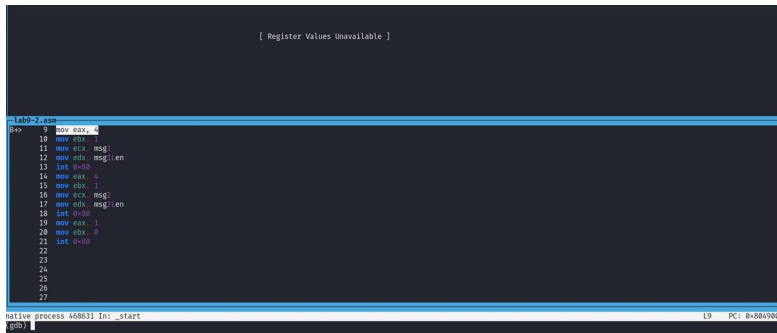
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
    0x08049005 <+5>:      mov     ebx,0x1
    0x0804900a <+10>:     mov     ecx,0x804a000
    0x0804900f <+15>:     mov     edx,0x8
    0x08049014 <+20>:     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) █

```

Переключение на Intel'ский синтаксис

Отличия заключаются в том, что в режиме АТТ используются “%” перед именами регистров и “\$” перед именами операндов, а в режиме Intel используется обычный синтаксис.

Включаю режим псевдографики для более удобного анализа программы.
(рис. [??])



Включение режима псевдографики

3.2.1 Добавление точек останова

Проверяю точку останова по имени метки.(рис. [??])

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

Проверка точки останова

Определяю адрес предпоследней инструкции и устанавливаю точку останова. Далее смотрю информацию о всех установленных точках останова.(рис. [??])

```
native process 468631 In: _start
(gdb) i b
Num      Type      Disp Enb Address  What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
          breakpoint already hit 1 time
(gdb) break *0x08049031
Breakpoint 2 at 0x08049031: file lab9-2.asm, line 20.
(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  0x08049031 lab9-2.asm:20
```

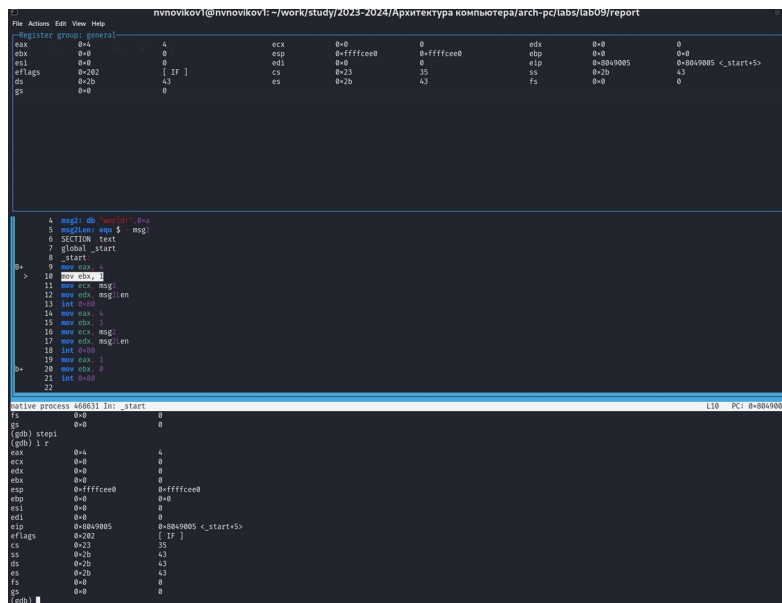
Установка точки останова и её проверка

3.2.2 Работа с данными программы в GDB

Выполняю 5 инструкций с помощью команды stepi и слежу за изменением регистров.(рис. [??]) (рис. [??])

```
(gdb) i r
eax                0x0                0
ecx                0x0                0
edx                0x0                0
ebx                0x0                0
esp                0xffffcee0         0xffffcee0
ebp                0x0                0x0
esi                0x0                0
edi                0x0                0
eip                0x8049000         0x8049000 <_start>
eflags             0x202             [ IF ]
cs                 0x23              35
ss                 0x2b              43
ds                 0x2b              43
es                 0x2b              43
fs                 0x0                0
gs                 0x0                0
(gdb) █
```

До использования команды stepi



После использования команды stepi

Изменились регистры eax,ebx,ecx,edx

Просматриваю значение переменной msg1 по имени.(рис. [??])

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

Просмотр значения переменной msg1

Также просматриваю значение переменной msg2.(рис. [??])

```
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) █
```

Просмотр значения переменной msg2

Изменяю первый символ переменной msg1.(рис. [??])

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

Изменение первого символа msg1

Изменяю первый символ переменной msg2.(рис. [??])

```
(gdb) set {char}0x804a008='t'
(gdb) x/lsb &msg2
0x804a008 <msg2>: "torld!\n\034"
```

Изменение первого символа переменной msg2

Вывожу в различных форматах значение регистра ebx.(рис. [??])

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

Вывод значений регистра ebx

С помощью команды set изменяю значение регистра ebx.(рис. [??])


```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb) █
```

Изменение значения регистра ebx

Разница в том, что в первый раз мы вводим символ, который преобразуется в число, а во второй раз мы вводим само число.

3.2.3 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. И создаю исполняемый файл.(рис. [??])

```
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ nasm -f elf -g -l lab9-3.lst lab9-3.asm

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o lab9-3 lab9-3.o
```

Создание исполняемого файла

Заружаю исполняемый файл в отладчик, указав аргументы.(рис. [??])

```
(nvnovikov1@nvnovikov1)-[~/arch-pc/labs/lab09/report]
$ gdb --args lab9-3 аргумент1 аргумент2 'аргумент 3'
GNU gdb (Debian 13.2-1) 13.2
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-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...
(gdb) |
```

Загрузка исполняемого файла в отладчик

Для начала устанавливаю точку останова перед первой инструкцией в программе и запускаю её.(рис. [??])

```
(gdb) b _start
Breakpoint 1 at 0x00490e0: file lab9-3.asm, line 8.
(gdb) run
Starting program: /home/nvnovikov1/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/report/lab9-3 аргумент1 аргумент2 аргумент\ 3
Breakpoint 1, _start () at lab9-3.asm:8
8   pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) |
```

Установка точки останова

Просматриваю вершину стека, то есть число аргументов строки(включая имя программы).(рис. [??])

```
(gdb) x/x $esp
0xffffce60: 0x00000004
```

Просмотр вершины стека

Просматриваю остальные позиции стека.(рис. [??])

```
(gdb) x/s *(void**)(($esp + 4)
0xffffd04c: "/home/nvnovikov1/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/report/lab9-3"
(gdb) x/s *(void**)(($esp + 8)
0xffffd00f: "аргумент1"
(gdb) x/s *(void**)(($esp + 12)
0xffffd0d1: "аргумент2"
(gdb) x/s *(void**)(($esp + 16)
0xffffd0e3: "аргумент 3"
(gdb) x/s *(void**)(($esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(($esp + 24)
0xffffd0f6: "COLORFGBG=15;0"
```

Просмотр остальных позиций стека

Шаг изменения адреса равен 4, потому что значение регистра esp в стеке увеличивается на 4.

4 Задание для самостоятельной работы.

1. Открываю программу из лабораторной работы №8 и начинаю её редактировать.(рис. [??])

```
File Edit Search View Document Help
lab9-2.asm
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi,0
11 mov edi,3
12 next1:
13 cmp ecx,0h
14 jz end
15 pop eax
16 call atoi
17 call f
18 add esi, eax
19 loop next1
20 push edx
21 end
22 mov eax, msg
23 call sprint
24 mov eax, esi
25 call iprintf
26 call quit
27 f:
28 add eax,10
29 mul edi
30 ret
```

Редактирование программы

Создаю исполняемый файл и запускаю его, чтобы проверить работу программы.(рис. [??])

```
(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ nasm -f elf test.asm

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o test test.o

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ./test 5 2 1 7
Результат: 165

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ./test 1 2
Результат: 69
```

Проверка работы программы

Программа работает верно.

2. Создаю файл test3.asm и ввожу туда текст программы из листинга 9.3.
(рис. [??])

```
File Edit Search View Document Help
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab09/report/test3.asm - Mousepad
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 iprintf
20 call quit
```

Ввод программы из листинга 9.3

Создаю исполняемый файл и запускаю его.(рис. [??])

```

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ nasm -f elf test3.asm

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o test3 test3.o

(nvnovikov1@nvnovikov1)-[~/.../arch-pc/labs/lab09/report]
$ ./test3
Результат: 10

```

Запуск программы

Убеждаюсь, что программа работает неверно.

Создаю исполняемый файл для работы с GDB и запускаю его через режим отладки. Создаю брейкпойнт и пошагово просматриваю программу.(рис. [??])

```

Register dump:
eax 0x804a000 134520832 ecx 0x4 4
edx 0x0 ebx 0xa 10
esp 0xffffc218 0xffffc218 ebp 0x0 0x0
esi 0x0 edi 0xa 10
eip 0x8049010 0x8049010 <sprint+1> eflags 0x206 [ PF IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0x804900f <sprint> push %edx
> 0x8049010 <sprint+1> push %ecx
0x8049011 <sprint+2> push %ebx
0x8049012 <sprint+3> push %eax
0x8049013 <sprint+4> call 0x8049000 <slen>
0x8049018 <sprint+9> mov %eax,%edx
0x804901a <sprint+11> pop %eax
0x804901b <sprint+12> mov %eax,%ecx
0x804901d <sprint+14> mov $0x1,%ebx
0x8049022 <sprint+19> mov $0x4,%eax
0x8049027 <sprint+24> int $0x80

native process 24231 In: sprint
(gdb) si
Breakpoint 3, _start () at test3.asm:10
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si

```

Просмотр программы

Замечаю, что после выполнения инструкции mul программы умножит 4 на 2 на не на 5, как должно быть.Из-за этого программа выдает неверный результат.

Далее открываю файл с программой и исправляю ошибку.(рис. [??])

```

File Edit Search View Document Help
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 iprintfLF
20 call quit

```

Исправление ошибки

Создаю исполняемый файл и запускаю его.(рис. [??])

```
(nvnovikov1@nvnovikov1)-[~/../arch-pc/labs/lab09/report]
$ nasm -f elf test3.asm

(nvnovikov1@nvnovikov1)-[~/../arch-pc/labs/lab09/report]
$ ld -m elf_i386 -o test3 test3.o

(nvnovikov1@nvnovikov1)-[~/../arch-pc/labs/lab09/report]
$ ./test3
Результат: 25
```

Просмотр программы

Теперь программа работает верно.

5 Выводы

После выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм и познакомился с методами отладки при помощи GDB и его основными возможностями.

Список литературы