

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Трофимов Владислав Алексеевич

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1 Цель работы	3
2 Задание.....	4
3 Теоретическое введение	5
4 Выполнение лабораторной работы	7
4.1 Релазиация подпрограмм в NASM	7
4.1.1 Отладка программ с помощью GDB	9
4.1.2 Добавление точек останова	12
4.1.3 Работа с данными программы в GDB	12
4.1.4 Обработка аргументов командной строки в GDB	15
4.2 Задание для самостоятельной работы.....	16
5 Выводы	20
6 Список литературы	21

Список иллюстраций

- Рис. 4.1: Создание рабочего каталога
- Рис. 4.2: Запуск программы из листинга
- Рис. 4.3: Изменение программы первого листинга
- Рис. 4.4: Запуск программы в отладчике
- Рис. 4.5: Проверка программы отладчиком
- Рис. 4.6: Запуск отладчика с брейкпоинтом
- Рис. 4.7: Дисассимилирование программы
- Рис. 4.8: Режим псевдографики
- Рис. 4.9: Список брейкпоинтов
- Рис. 4.10: Добавление второй точки останова
- Рис. 4.11: Просмотр содержимого регистров
- Рис. 4.12: Просмотр содержимого переменных двумя способами
- Рис. 4.13: Изменение содержимого переменных двумя способами
- Рис. 4.14: Просмотр значения регистра разными представлениями
- Рис. 4.15: Примеры использования команды set
- Рис. 4.16: Подготовка новой программы
- Рис. 4.17: Проверка работы стека
- Рис. 4.18: Измененная программа предыдущей лабораторной работы
- Рис. 4.19: Поиск ошибки в программе через пошаговую отладку
- Рис. 4.20: Проверка корректировок в программме

1 Цель работы

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

2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

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-таки трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

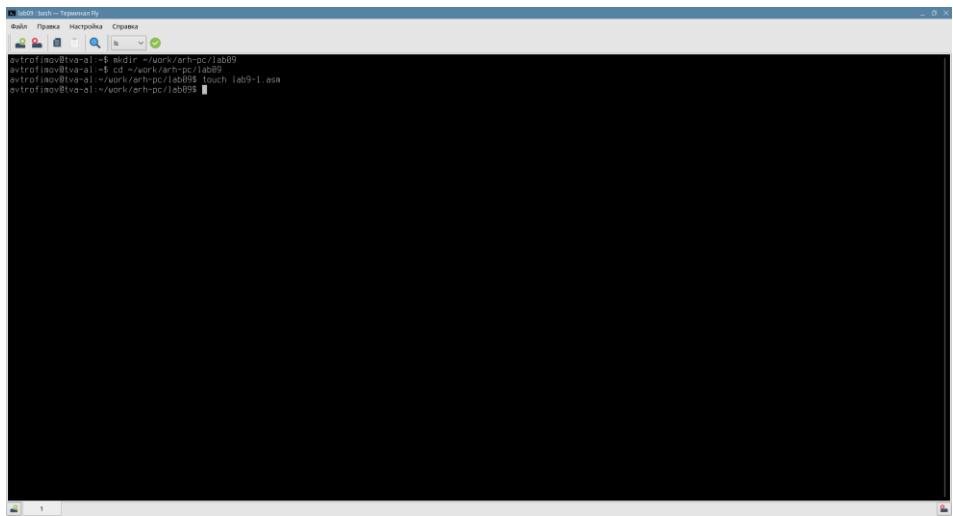
Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

4.1 Релазиация подпрограмм в NASM

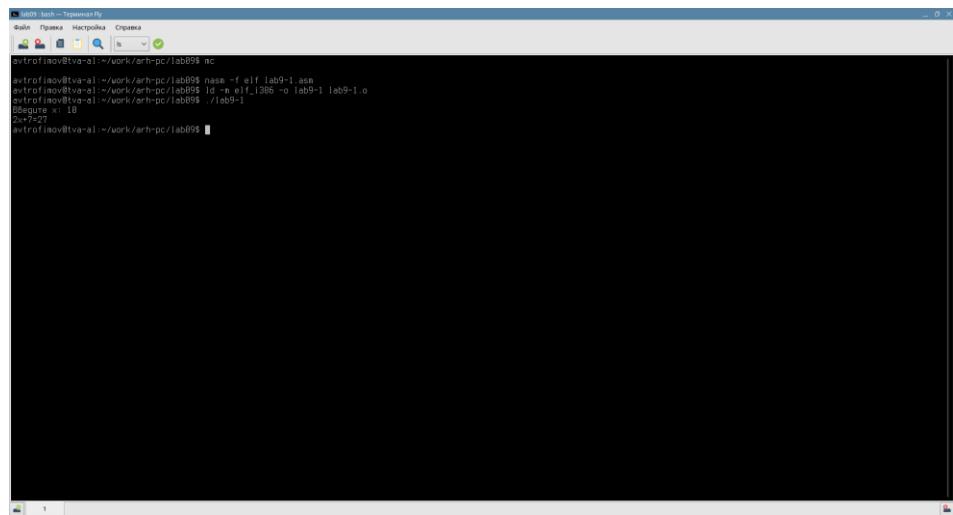
Создаю каталог для выполнения лабораторной работы №9 (рис. 1).



```
lab09 bash - Терминал
Файл Трекка Настройка Справка
ls
avtrolinov@Viva-ai: ~$ mkdir ~/work/arth-pc/lab09
avtrolinov@Viva-ai: ~$ cd ~/work/arth-pc/lab09
avtrolinov@Viva-ai: ~$ touch lab9-1.asm
avtrolinov@Viva-ai: ~$
```

Рис. 1: Создание рабочего каталога

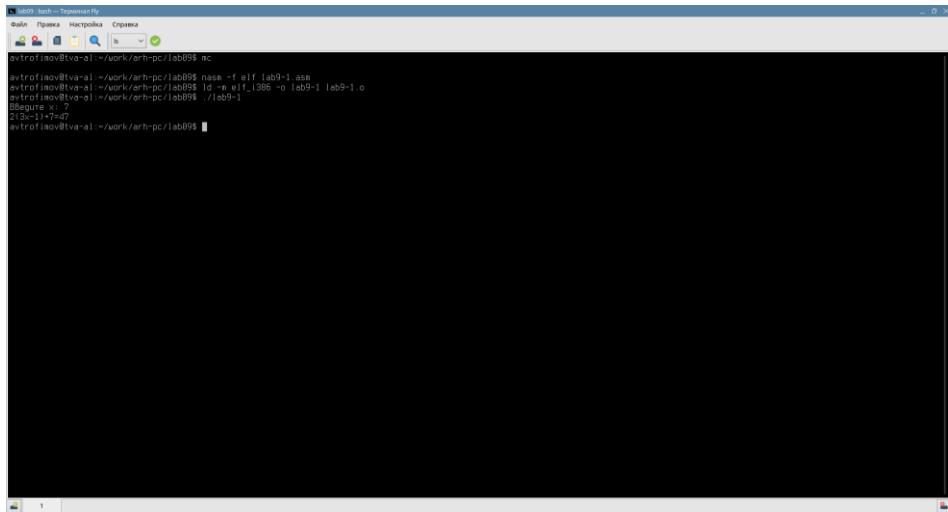
Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 2).



```
lab09 bash - Терминал
Файл Трекка Настройка Справка
ls
avtrolinov@Viva-ai: ~$ work/arth-pc/lab09$ nc
avtrolinov@Viva-ai: ~$ work/arth-pc/lab09$ nasm -f elf lab9-1.asm
avtrolinov@Viva-ai: ~$ work/arth-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
avtrolinov@Viva-ai: ~$ work/arth-pc/lab09$ ./lab9-1
Сбрасывай x18
Сбрасывай x18
avtrolinov@Viva-ai: ~$ work/arth-pc/lab09$
```

Рис. 2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 3).



```
lab09: ~ /work/erh-pc/lab09$ nc
avtrolimov@tva-s1:~/work/erh-pc/lab09$ nasm -f elf lab9-1.asm
avtrolimov@tva-s1:~/work/erh-pc/lab09$ ld -n elf1386 -o lab9-1 lab9-1.o
avtrolimov@tva-s1:~/work/erh-pc/lab09$ ./lab9-1
Debugfile: ./lab9-1.d
avtrolimov@tva-s1:~/work/erh-pc/lab09$
```

Рис. 3: Изменение программы первого листинга

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ', 0
result: DB '2(3x-1)+7=', 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 _calcul
```

```
mov eax, result
```

```
call sprint
```

```
mov eax, [res]
```

```
call iprintLF
```

```
call quit
```

```
_calcul:
```

```
push eax
```

```
call _subcalcul
```

```
mov ebx, 2
```

```
mul ebx
```

```
add eax, 7
```

```
mov [res], eax
```

```
pop eax
```

```
ret
```

```
_subcalcul:
```

```
mov ebx, 3
```

```
mul ebx
```

```
sub eax, 1
```

```
ret
```

4.1.1 Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 4).

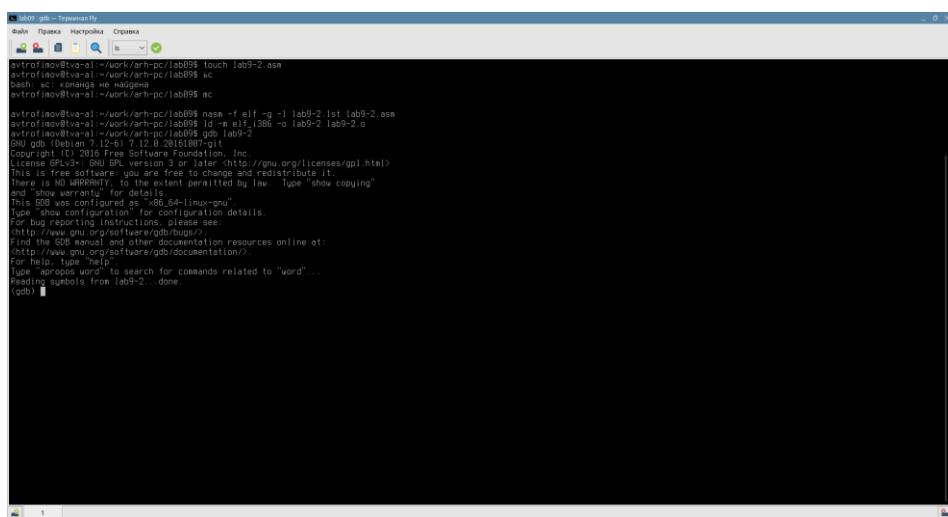
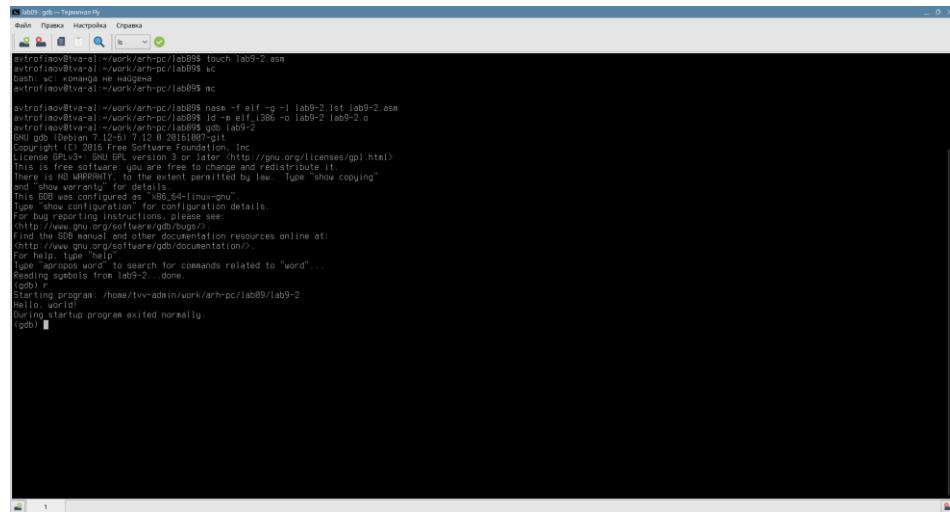


Рис. 4: Запуск программы в отладчике

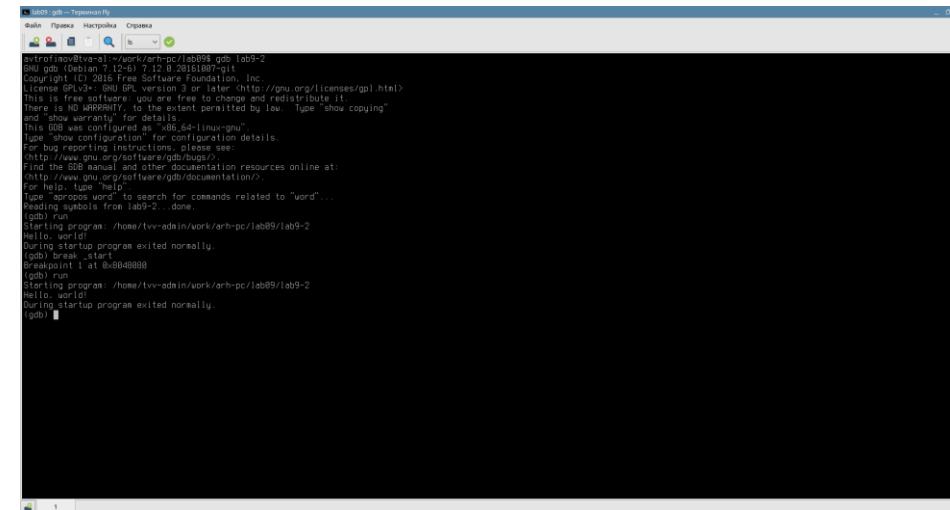
Запустив программу командой run, я убедился в том, что она работает исправно (рис. 5).



```
avtrolimovtva-ai ~ /work/arh-pc/lab09$ touch lab9-2.asm
avtrolimovtva-ai ~ /work/arh-pc/lab09$ nasm -f elf -g -l lab9-2.lab9-2.o
avtrolimovtva-ai ~ /work/arh-pc/lab09$ ld -n elf_i386 -o lab9-2 lab9-2.o
avtrolimovtva-ai ~ /work/arh-pc/lab09$ gdb lab9-2
GNU gdb (GDB) 7.12.1-20160701-git
Copyright (C) 2016 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:
  http://www.gnu.org/software/gdb/bug.html
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...done.
(gdb) r
Starting program: /home/lvv-admin/work/arh-pc/lab09/lab9-2
Hello, world!
During startup program exited normally.
(gdb)
```

Рис. 5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку `_start` и снова запускаю отладку (рис. 6).



```
avtrolimovtva-ai ~ /work/arh-pc/lab09$ gdb lab9-2
GNU gdb (GDB) 7.12.1-20160701-git
Copyright (C) 2016 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:
  http://www.gnu.org/software/gdb/bug.html
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...done.
(gdb) run
Starting program: /home/lvv-admin/work/arh-pc/lab09/lab9-2
Breakpoint 1 at 0x00000000
(gdb) break *start
Breakpoint 1 at 0x00000000
(gdb) run
Starting program: /home/lvv-admin/work/arh-pc/lab09/lab9-2
Breakpoint 1, 0x00000000 in _start ()
(gdb) s
During startup program exited normally.
(gdb)
```

Рис. 6: Запуск отладчика с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команды с синтаксисом Intel амд топчик (рис. 7).

Различия между синтаксисом ATT и Intel заключаются в

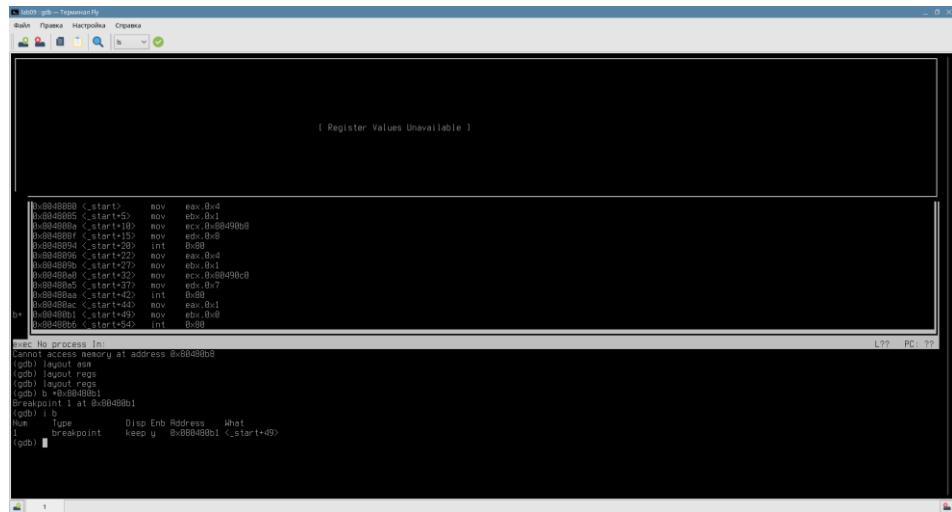
порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```

lab01:~/lab$ gdb --Terminal Nu
Файл Правка Настройка Справка
[...]
Reading symbols from lab9-2... done.
(gdb) run
Starting program: /home/tvt-admin/work/arch-pc/lab09/lab9-2
Hello, world!
[...]
breakpoint 1 at 0x00400000
(gdb) break start
Breakpoint 1 at 0x00400000
(gdb) run
Starting program: /home/tvt-admin/work/arch-pc/lab09/lab9-2
Hello, world!
[...]
Breakpoint 1, 0x00400000 in _start ()
(gdb) disassembly start
Dump of assembler code for function _start:
0x00400000 <_start>:    mov    eax,0xd
0x00400004 <_start+4>:    mov    eax,0x1
0x00400008 <_start+8>:    mov    ecx,0xbfbff0b0
0x0040000c <_start+12>:   mov    edx,0x8
0x00400010 <_start+16>:   int    $0x80
0x00400014 <_start+20>:   mov    ebx,0xd
0x00400018 <_start+24>:   mov    ebx,0x1
0x0040001c <_start+28>:   mov    edx,0x7
0x00400020 <_start+32>:   mov    eax,0x1
0x00400024 <_start+36>:   mov    eax,0xbfbff0b0
0x00400028 <_start+40>:   mov    edx,0x8
0x0040002c <_start+44>:   mov    ebx,0xd
0x00400030 <_start+48>:   mov    ebx,0x1
0x00400034 <_start+52>:   mov    edx,0x7
0x00400038 <_start+56>:   mov    eax,0x1
0x0040003c <_start+60>:   mov    eax,0xbfbff0b0
0x00400040 <_start+64>:   mov    edx,0x8
0x00400044 <_start+68>:   mov    ebx,0xd
0x00400048 <_start+72>:   mov    ebx,0x1
0x0040004c <_start+76>:   mov    edx,0x7
0x00400050 <_start+80>:   mov    eax,0x1
0x00400054 <_start+84>:   mov    eax,0xbfbff0b0
0x00400058 <_start+88>:   mov    edx,0x8
0x0040005c <_start+92>:   mov    ebx,0xd
0x00400060 <_start+96>:   mov    ebx,0x1
0x00400064 <_start+100>:  mov    edx,0x7
0x00400068 <_start+104>:  mov    eax,0x1
0x00400072 <_start+108>:  mov    eax,0xbfbff0b0
0x00400076 <_start+112>:  int    $0x80
0x00400080 <_start+116>:  mov    ebx,0xd
0x00400084 <_start+120>:  mov    ebx,0x1
0x00400088 <_start+124>:  mov    edx,0x7
0x00400092 <_start+128>:  mov    eax,0x1
0x00400096 <_start+132>:  mov    eax,0xbfbff0b0
0x0040009a <_start+136>:  mov    edx,0x8
0x0040009e <_start+140>:  mov    ebx,0xd
0x004000a2 <_start+144>:  mov    ebx,0x1
0x004000a6 <_start+148>:  mov    edx,0x7
0x004000a8 <_start+152>:  mov    eax,0x1
0x004000b2 <_start+156>:  mov    eax,0xbfbff0b0
0x004000b6 <_start+160>:  mov    edx,0x8
0x004000b8 <_start+164>:  mov    ebx,0xd
0x004000bc <_start+168>:  mov    ebx,0x1
0x004000c0 <_start+172>:  mov    edx,0x7
0x004000c4 <_start+176>:  mov    eax,0x1
0x004000c8 <_start+180>:  mov    eax,0xbfbff0b0
0x004000cc <_start+184>:  mov    edx,0x8
0x004000d0 <_start+188>:  mov    ebx,0xd
0x004000d4 <_start+192>:  mov    ebx,0x1
0x004000d8 <_start+196>:  mov    edx,0x7
0x004000dc <_start+200>:  mov    eax,0x1
0x004000e0 <_start+204>:  mov    eax,0xbfbff0b0
0x004000e4 <_start+208>:  mov    edx,0x8
0x004000e8 <_start+212>:  mov    ebx,0xd
0x004000f2 <_start+216>:  mov    ebx,0x1
0x004000f6 <_start+220>:  mov    edx,0x7
0x004000f8 <_start+224>:  mov    eax,0x1
0x004000fc <_start+228>:  mov    eax,0xbfbff0b0
0x004000f8 <_start+232>:  int    $0x80
0x00400100 <_start+236>:  mov    ebx,0xd
0x00400104 <_start+240>:  mov    ebx,0x1
0x00400108 <_start+244>:  mov    edx,0x7
0x0040010c <_start+248>:  mov    eax,0x1
0x00400110 <_start+252>:  mov    eax,0xbfbff0b0
0x00400114 <_start+256>:  mov    edx,0x8
0x00400118 <_start+260>:  mov    ebx,0xd
0x0040011c <_start+264>:  mov    ebx,0x1
0x00400120 <_start+268>:  mov    edx,0x7
0x00400124 <_start+272>:  mov    eax,0x1
0x00400128 <_start+276>:  mov    eax,0xbfbff0b0
0x0040012c <_start+280>:  mov    edx,0x8
0x00400130 <_start+284>:  mov    ebx,0xd
0x00400134 <_start+288>:  mov    ebx,0x1
0x00400138 <_start+292>:  mov    edx,0x7
0x0040013c <_start+296>:  mov    eax,0x1
0x00400140 <_start+300>:  mov    eax,0xbfbff0b0
0x00400144 <_start+304>:  mov    edx,0x8
0x00400148 <_start+308>:  mov    ebx,0xd
0x00400152 <_start+312>:  mov    ebx,0x1
0x00400156 <_start+316>:  mov    edx,0x7
0x0040015a <_start+320>:  mov    eax,0x1
0x00400164 <_start+324>:  mov    eax,0xbfbff0b0
0x00400168 <_start+328>:  mov    edx,0x8
0x00400172 <_start+332>:  mov    ebx,0xd
0x00400176 <_start+336>:  mov    ebx,0x1
0x00400180 <_start+340>:  mov    edx,0x7
0x00400184 <_start+344>:  mov    eax,0x1
0x00400188 <_start+348>:  mov    eax,0xbfbff0b0
0x00400192 <_start+352>:  mov    edx,0x8
0x00400196 <_start+356>:  mov    ebx,0xd
0x004001a0 <_start+360>:  mov    ebx,0x1
0x004001a4 <_start+364>:  mov    edx,0x7
0x004001a8 <_start+368>:  mov    eax,0x1
0x004001b2 <_start+372>:  mov    eax,0xbfbff0b0
0x004001b6 <_start+376>:  mov    edx,0x8
0x004001b8 <_start+380>:  mov    ebx,0xd
0x004001bc <_start+384>:  mov    ebx,0x1
0x004001c0 <_start+388>:  mov    edx,0x7
0x004001c4 <_start+392>:  mov    eax,0x1
0x004001c8 <_start+396>:  mov    eax,0xbfbff0b0
0x004001cc <_start+400>:  mov    edx,0x8
0x004001d0 <_start+404>:  mov    ebx,0xd
0x004001d4 <_start+408>:  mov    ebx,0x1
0x004001d8 <_start+412>:  mov    edx,0x7
0x004001dc <_start+416>:  mov    eax,0x1
0x004001e0 <_start+420>:  mov    eax,0xbfbff0b0
0x004001e4 <_start+424>:  mov    edx,0x8
0x004001e8 <_start+428>:  mov    ebx,0xd
0x004001f2 <_start+432>:  mov    ebx,0x1
0x004001f6 <_start+436>:  mov    edx,0x7
0x004001f8 <_start+440>:  mov    eax,0x1
0x004001fc <_start+444>:  mov    eax,0xbfbff0b0
0x004001f8 <_start+448>:  int    $0x80
0x00400200 <_start+452>:  mov    ebx,0xd
0x00400204 <_start+456>:  mov    ebx,0x1
0x00400208 <_start+460>:  mov    edx,0x7
0x0040020c <_start+464>:  mov    eax,0x1
0x00400210 <_start+468>:  mov    eax,0xbfbff0b0
0x00400214 <_start+472>:  mov    edx,0x8
0x00400218 <_start+476>:  mov    ebx,0xd
0x0040021c <_start+480>:  mov    ebx,0x1
0x00400220 <_start+484>:  mov    edx,0x7
0x00400224 <_start+488>:  mov    eax,0x1
0x00400228 <_start+492>:  mov    eax,0xbfbff0b0
0x0040022c <_start+496>:  mov    edx,0x8
0x00400230 <_start+500>:  mov    ebx,0xd
0x00400234 <_start+504>:  mov    ebx,0x1
0x00400238 <_start+508>:  mov    edx,0x7
0x0040023c <_start+512>:  mov    eax,0x1
0x00400240 <_start+516>:  mov    eax,0xbfbff0b0
0x00400244 <_start+520>:  mov    edx,0x8
0x00400248 <_start+524>:  mov    ebx,0xd
0x0040024c <_start+528>:  mov    ebx,0x1
0x00400250 <_start+532>:  mov    edx,0x7
0x00400254 <_start+536>:  mov    eax,0x1
0x00400258 <_start+540>:  mov    eax,0xbfbff0b0
0x0040025c <_start+544>:  mov    edx,0x8
0x00400260 <_start+548>:  mov    ebx,0xd
0x00400264 <_start+552>:  mov    ebx,0x1
0x00400268 <_start+556>:  mov    edx,0x7
0x00400272 <_start+560>:  mov    eax,0x1
0x00400276 <_start+564>:  mov    eax,0xbfbff0b0
0x0040027a <_start+568>:  mov    edx,0x8
0x00400284 <_start+572>:  mov    ebx,0xd
0x00400288 <_start+576>:  mov    ebx,0x1
0x00400292 <_start+580>:  mov    edx,0x7
0x00400296 <_start+584>:  mov    eax,0x1
0x0040029a <_start+588>:  mov    eax,0xbfbff0b0
0x004002a4 <_start+592>:  mov    edx,0x8
0x004002a8 <_start+596>:  mov    ebx,0xd
0x004002b2 <_start+600>:  mov    ebx,0x1
0x004002b6 <_start+604>:  mov    edx,0x7
0x004002b8 <_start+608>:  mov    eax,0x1
0x004002bc <_start+612>:  mov    eax,0xbfbff0b0
0x004002c0 <_start+616>:  mov    edx,0x8
0x004002c4 <_start+620>:  mov    ebx,0xd
0x004002c8 <_start+624>:  mov    ebx,0x1
0x004002d2 <_start+628>:  mov    edx,0x7
0x004002d6 <_start+632>:  mov    eax,0x1
0x004002d8 <_start+636>:  mov    eax,0xbfbff0b0
0x004002e2 <_start+640>:  mov    edx,0x8
0x004002e6 <_start+644>:  mov    ebx,0xd
0x004002e8 <_start+648>:  mov    ebx,0x1
0x004002f2 <_start+652>:  mov    edx,0x7
0x004002f6 <_start+656>:  mov    eax,0x1
0x004002f8 <_start+660>:  mov    eax,0xbfbff0b0
0x004002f8 <_start+664>:  int    $0x80
0x00400300 <_start+668>:  mov    ebx,0xd
0x00400304 <_start+672>:  mov    ebx,0x1
0x00400308 <_start+676>:  mov    edx,0x7
0x0040030c <_start+680>:  mov    eax,0x1
0x00400310 <_start+684>:  mov    eax,0xbfbff0b0
0x00400314 <_start+688>:  mov    edx,0x8
0x00400318 <_start+692>:  mov    ebx,0xd
0x0040031c <_start+696>:  mov    ebx,0x1
0x00400320 <_start+700>:  mov    edx,0x7
0x00400324 <_start+704>:  mov    eax,0x1
0x00400328 <_start+708>:  mov    eax,0xbfbff0b0
0x0040032c <_start+712>:  mov    edx,0x8
0x00400330 <_start+716>:  mov    ebx,0xd
0x00400334 <_start+720>:  mov    ebx,0x1
0x00400338 <_start+724>:  mov    edx,0x7
0x0040033c <_start+728>:  mov    eax,0x1
0x00400340 <_start+732>:  mov    eax,0xbfbff0b0
0x00400344 <_start+736>:  mov    edx,0x8
0x00400348 <_start+740>:  mov    ebx,0xd
0x0040034c <_start+744>:  mov    ebx,0x1
0x00400350 <_start+748>:  mov    edx,0x7
0x00400354 <_start+752>:  mov    eax,0x1
0x00400358 <_start+756>:  mov    eax,0xbfbff0b0
0x0040035c <_start+760>:  mov    edx,0x8
0x00400360 <_start+764>:  mov    ebx,0xd
0x00400364 <_start+768>:  mov    ebx,0x1
0x00400368 <_start+772>:  mov    edx,0x7
0x00400372 <_start+776>:  mov    eax,0x1
0x00400376 <_start+780>:  mov    eax,0xbfbff0b0
0x0040037a <_start+784>:  mov    edx,0x8
0x00400384 <_start+788>:  mov    ebx,0xd
0x00400388 <_start+792>:  mov    ebx,0x1
0x0040038c <_start+796>:  mov    edx,0x7
0x00400390 <_start+800>:  mov    eax,0x1
0x00400394 <_start+804>:  mov    eax,0xbfbff0b0
0x00400398 <_start+808>:  mov    edx,0x8
0x004003a0 <_start+812>:  mov    ebx,0xd
0x004003a4 <_start+816>:  mov    ebx,0x1
0x004003a8 <_start+820>:  mov    edx,0x7
0x004003a8 <_start+824>:  mov    eax,0x1
0x004003b2 <_start+828>:  mov    eax,0xbfbff0b0
0x004003b6 <_start+832>:  mov    edx,0x8
0x004003b8 <_start+836>:  mov    ebx,0xd
0x004003bc <_start+840>:  mov    ebx,0x1
0x004003c0 <_start+844>:  mov    edx,0x7
0x004003c4 <_start+848>:  mov    eax,0x1
0x004003c8 <_start+852>:  mov    eax,0xbfbff0b0
0x004003cc <_start+856>:  mov    edx,0x8
0x004003d0 <_start+860>:  mov    ebx,0xd
0x004003d4 <_start+864>:  mov    ebx,0x1
0x004003d8 <_start+868>:  mov    edx,0x7
0x004003dc <_start+872>:  mov    eax,0x1
0x004003e0 <_start+876>:  mov    eax,0xbfbff0b0
0x004003e4 <_start+880>:  mov    edx,0x8
0x004003e8 <_start+884>:  mov    ebx,0xd
0x004003f2 <_start+888>:  mov    ebx,0x1
0x004003f6 <_start+892>:  mov    edx,0x7
0x004003f8 <_start+896>:  mov    eax,0x1
0x004003fc <_start+900>:  mov    eax,0xbfbff0b0
0x00400400 <_start+904>:  mov    edx,0x8
0x00400404 <_start+908>:  mov    ebx,0xd
0x00400408 <_start+912>:  mov    ebx,0x1
0x00400412 <_start+916>:  mov    edx,0x7
0x00400414 <_start+918>:  mov    eax,0x1
0x00400418 <_start+922>:  mov    eax,0xbfbff0b0
0x0040041c <_start+926>:  mov    edx,0x8
0x00400420 <_start+930>:  mov    ebx,0xd
0x00400424 <_start+934>:  mov    ebx,0x1
0x00400428 <_start+938>:  mov    edx,0x7
0x0040042c <_start+942>:  mov    eax,0x1
0x00400430 <_start+946>:  mov    eax,0xbfbff0b0
0x00400434 <_start+950>:  mov    edx,0x8
0x00400438 <_start+954>:  mov    ebx,0xd
0x00400442 <_start+958>:  mov    ebx,0x1
0x00400446 <_start+962>:  mov    edx,0x7
0x00400448 <_start+966>:  mov    eax,0x1
0x00400450 <_start+970>:  mov    eax,0xbfbff0b0
0x00400454 <_start+974>:  mov    edx,0x8
0x00400458 <_start+978>:  mov    ebx,0xd
0x00400462 <_start+982>:  mov    ebx,0x1
0x00400466 <_start+986>:  mov    edx,0x7
0x00400468 <_start+990>:  mov    eax,0x1
0x00400470 <_start+994>:  mov    eax,0xbfbff0b0
0x00400474 <_start+998>:  mov    edx,0x8
0x00400478 <_start+1002>:  mov    ebx,0xd
0x00400482 <_start+1006>:  mov    ebx,0x1
0x00400486 <_start+1010>:  mov    edx,0x7
0x00400488 <_start+1012>:  mov    eax,0x1
0x0040048c <_start+1016>:  mov    eax,0xbfbff0b0
0x00400490 <_start+1020>:  mov    edx,0x8
0x00400494 <_start+1024>:  mov    ebx,0xd
0x00400498 <_start+1028>:  mov    ebx,0x1
0x0040049c <_start+1032>:  mov    edx,0x7
0x0040049e <_start+1036>:  mov    eax,0x1
0x004004a0 <_start+1040>:  mov    eax,0xbfbff0b0
0x004004a4 <_start+1044>:  mov    edx,0x8
0x004004a8 <_start+1048>:  mov    ebx,0xd
0x004004b2 <_start+1052>:  mov    ebx,0x1
0x004004b6 <_start+1056>:  mov    edx,0x7
0x004004b8 <_start+1060>:  mov    eax,0x1
0x004004bc <_start+1064>:  mov    eax,0xbfbff0b0
0x004004c0 <_start+1068>:  mov    edx,0x8
0x004004c4 <_start+1072>:  mov    ebx,0xd
0x004004c8 <_start+1076>:  mov    ebx,0x1
0x004004d2 <_start+1080>:  mov    edx,0x7
0x004004d4 <_start+1084>:  mov    eax,0x1
0x004004d8 <_start+1088>:  mov    eax,0xbfbff0b0
0x004004dc <_start+1092>:  mov    edx,0x8
0x004004e0 <_start+1096>:  mov    ebx,0xd
0x004004e4 <_start+1098>:  mov    ebx,0x1
0x004004e8 <_start+1102>:  mov    edx,0x7
0x004004f0 <_start+1106>:  mov    eax,0x1
0x004004f4 <_start+1110>:  mov    eax,0xbfbff0b0
0x004004f8 <_start+1114>:  mov    edx,0x8
0x004004f8 <_start+1118>:  int    $0x80
0x00400500 <_start+1122>:  mov    ebx,0xd
0x00400504 <_start+1126>:  mov    ebx,0x1
0x00400508 <_start+1130>:  mov    edx,0x7
0x0040050c <_start+1134>:  mov    eax,0x1
0x00400510 <_start+1136>:  mov    eax,0xbfbff0b0
0x00400514 <_start+1140>:  mov    edx,0x8
0x00400518 <_start+1144>:  mov    ebx,0xd
0x00400522 <_start+1148>:  mov    ebx,0x1
0x00400526 <_start+1152>:  mov    edx,0x7
0x00400528 <_start+1156>:  mov    eax,0x1
0x00400530 <_start+1160>:  mov    eax,0xbfbff0b0
0x00400534 <_start+1164>:  mov    edx,0x8
0x00400538 <_start+1168>:  mov    ebx,0xd
0x00400542 <_start+1172>:  mov    ebx,0x1
0x00400546 <_start+1176>:  mov    edx,0x7
0x00400548 <_start+1180>:  mov    eax,0x1
0x00400550 <_start+1184>:  mov    eax,0xbfbff0b0
0x00400554 <_start+1188>:  mov    edx,0x8
0x00400558 <_start+1192>:  mov    ebx,0xd
0x00400562 <_start+1196>:  mov    ebx,0x1
0x00400566 <_start+1200>:  mov    edx,0x7
0x00400568 <_start+1204>:  mov    eax,0x1
0x00400570 <_start+1208>:  mov    eax,0xbfbff0b0
0x00400574 <_start+1212>:  mov    edx,0x8
0x00400578 <_start+1216>:  mov    ebx,0xd
0x00400582 <_start+1220>:  mov    ebx,0x1
0x00400586 <_start+1224>:  mov    edx,0x7
0x00400588 <_start+1228>:  mov    eax,0x1
0x00400590 <_start+1232>:  mov    eax,0xbfbff0b0
0x00400594 <_start+1236>:  mov    edx,0x8
0x00400598 <_start+1240>:  mov    ebx,0xd
0x004005a2 <_start+1244>:  mov    ebx,0x1
0x004005a6 <_start+1248>:  mov    edx,0x7
0x004005a8 <_start+1252>:  mov    eax,0x1
0x004005b0 <_start+1256>:  mov    eax,0xbfbff0b0
0x004005b4 <_start+1260>:  mov    edx,0x8
0x004005b8 <_start+1264>:  mov    ebx,0xd
0x004005bc <_start+1268>:  mov    ebx,0x1
0x004005c0 <_start+1272>:  mov    edx,0x7
0x004005c4 <_start+1276>:  mov    eax,0x1
0x004005c8 <_start+1280>:  mov    eax,0xbfbff0b0
0x004005cc <_start+1284>:  mov    edx,0x8
0x004005d0 <_start+1288>:  mov    ebx,0xd
0x004005d4 <_start+1292>:  mov    ebx,0x1
0x004005d8 <_start+1296>:  mov    edx,0x7
0x004005dc <_start+1300>:  mov    eax,0x1
0x004005e0 <_start+1304>:  mov    eax,0xbfbff0b0
0x004005e4 <_start+1308>:  mov    edx,0x8
0x004005e8 <_start+1312>:  mov    ebx,0xd
0x004005f2 <_start+1316>:  mov    ebx,0x1
0x004005f6 <_start+1320>:  mov    edx,0x7
0x004005f8 <_start+1324>:  mov    eax,0x1
0x004005fc <_start+1328>:  mov    eax,0xbfbff0b0
0x00400600 <_start+1332>:  mov    edx,0x8
0x00400604 <_start+1336>:  mov    ebx,0xd
0x00400608 <_start+1340>:  mov    ebx,0x1
0x00400612 <_start+1344>:  mov    edx,0x7
0x00400616 <_start+1348>:  mov    eax,0x1
0x0040061a <_start+1352>:  mov    eax,0xbfb
```

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).



The screenshot shows the GDB terminal window with the following content:

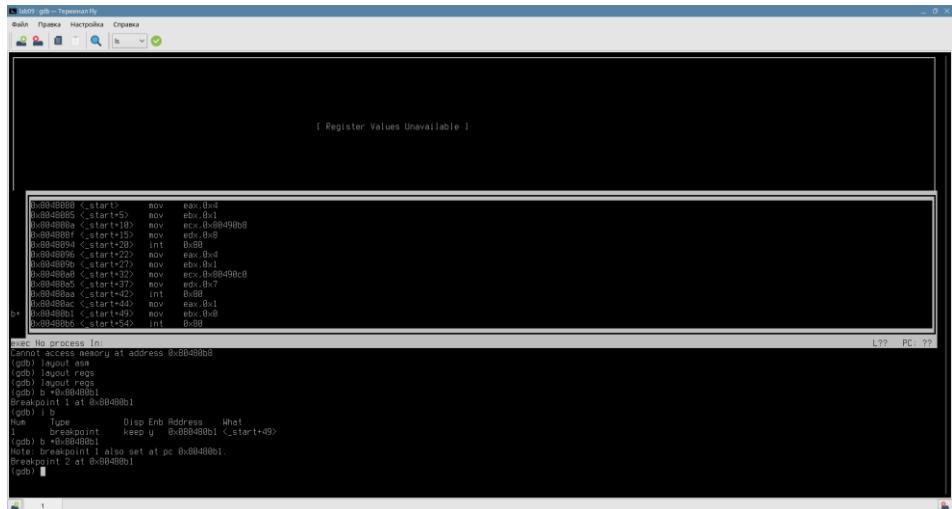
```
[Register Values Unavailable]

0x00400000 <_start>    mov    eax,0x0
0x00400005 <_start+5>   mov    ebx,0x1
0x0040000a <_start+10>  mov    ecx,0x004000b9
0x0040000f <_start+15>  mov    edx,0x8
0x00400014 <_start+20>  mov    esp,0x0
0x00400019 <_start+22>  mov    eax,0x4
0x0040001e <_start+27>  mov    ebx,0x1
0x00400023 <_start+32>  mov    ecx,0x004000c0
0x00400028 <_start+37>  mov    edx,0x7
0x00400033 <_start+42>  int    0x80
0x00400038 <_start+47>  mov    eax,0x1
0x0040003d <_start+52>  mov    ebx,0x8
0x00400042 <_start+54>  int    0x80

Breakpoint 1 at 0x004000b1
(gdb) b *0x004000b1
Breakpoint 1 at 0x004000b1
(gdb) i br
Num  Type      Disp Enb Address    What
1*   breakpoint  keep y  0x004000b1 <_start+49>
(gdb) b *0x004000b1
Breakpoint 2 at 0x004000b1
(gdb)
```

Рис. 9: Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. 10).



The screenshot shows the GDB terminal window with the following content:

```
[Register Values Unavailable]

0x00400000 <_start>    mov    eax,0x0
0x00400005 <_start+5>   mov    ebx,0x1
0x0040000a <_start+10>  mov    ecx,0x004000b9
0x0040000f <_start+15>  mov    edx,0x8
0x00400014 <_start+20>  mov    esp,0x0
0x00400019 <_start+22>  mov    eax,0x4
0x0040001e <_start+27>  mov    ebx,0x1
0x00400023 <_start+32>  mov    ecx,0x004000c0
0x00400028 <_start+37>  mov    edx,0x7
0x00400033 <_start+42>  int    0x80
0x00400038 <_start+47>  mov    eax,0x1
0x0040003d <_start+52>  mov    ebx,0x8
0x00400042 <_start+54>  int    0x80

Breakpoint 1 at 0x004000b1
(gdb) b *0x004000b1
Breakpoint 1 at 0x004000b1
(gdb) i br
Num  Type      Disp Enb Address    What
1*   breakpoint  keep y  0x004000b1 <_start+49>
(gdb) b *0x004000b1
Breakpoint 2 at 0x004000b1
Breakpoint 2 at 0x004000b1
(gdb) b *0x004000b1
Breakpoint 3 at 0x004000b1
(gdb)
```

Рис. 10: Добавление второй точки останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. 11).

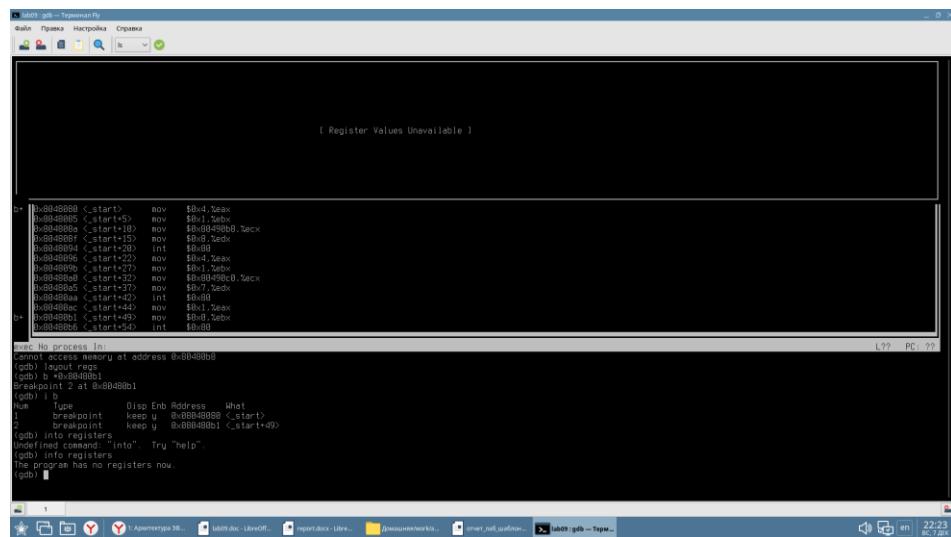


Рис. 11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 12).

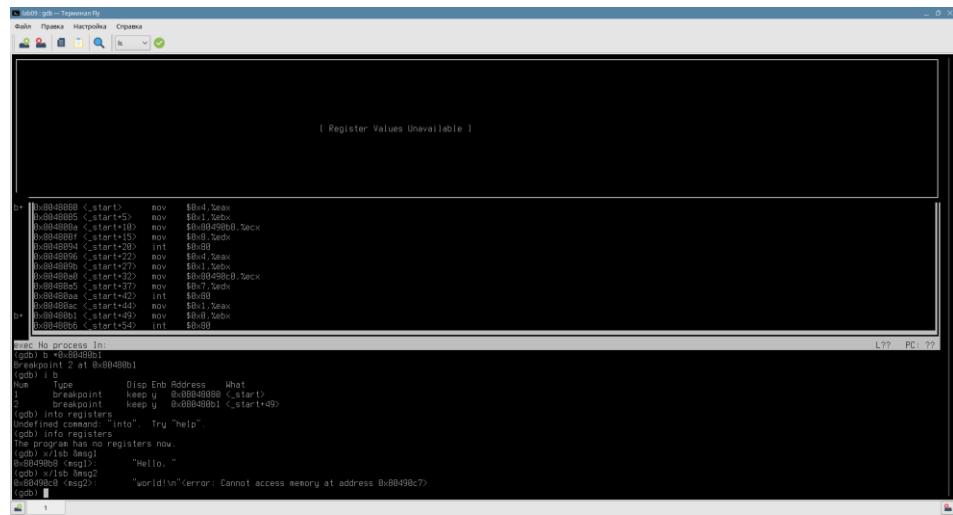


Рис. 12: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. 13).

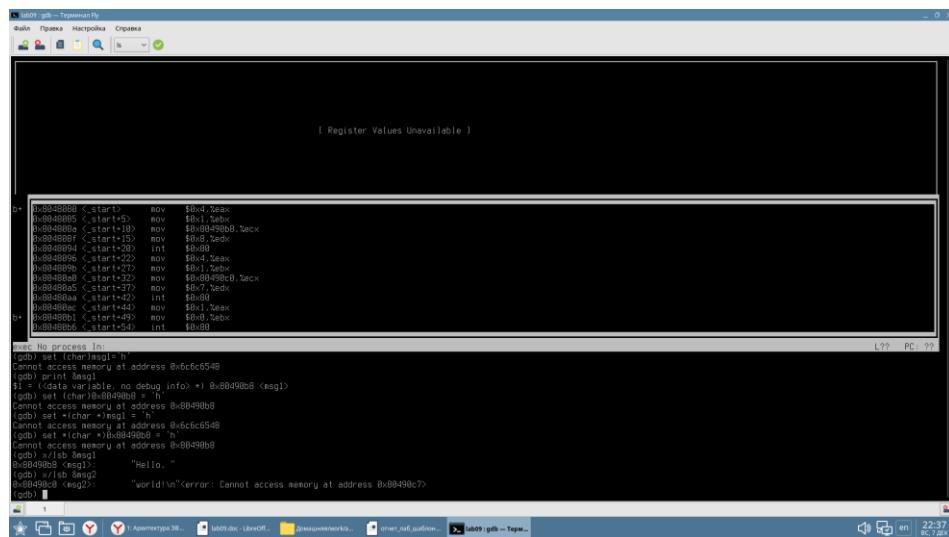


Рис. 13: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. 14).

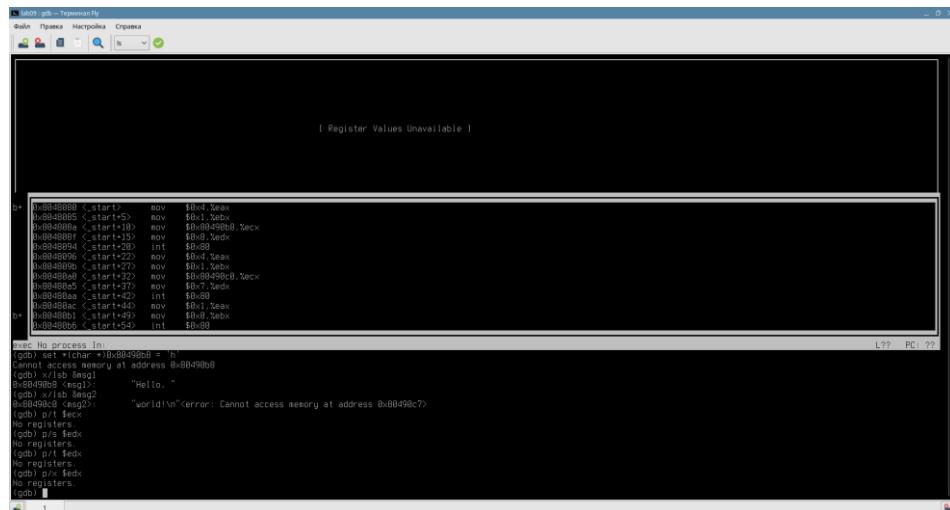


Рис. 14: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. 15).

The screenshot shows the GDB interface with assembly code and command history. The assembly code is as follows:

```

b* 0x00400000 <start>: mov    $0x4, %eax
0x00400004 <start+4>: mov    $0x1, %ebx
0x00400008 <start+8>: mov    $0x00490b0, %ecx
0x0040000c <start+12>: mov    $0x0, %edx
0x00400010 <start+16>: int    $0x80
0x00400014 <start+20>: mov    $0x4, %eax
0x00400018 <start+24>: mov    $0x1, %ebx
0x00400022 <start+28>: mov    $0x00490b0, %ecx
0x00400026 <start+32>: mov    $0x7, %edx
0x00400030 <start+36>: int    $0x80
0x00400034 <start+40>: mov    $0x0, %eax
0x00400038 <start+44>: mov    $0x1, %ebx
0x0040003c <start+48>: int    $0xb8

```

The command history shows:

```

(gdb) set *char *0x00490b0 = 'h'
Cannot access memory at address 0x00490b0
(gdb) p/s $esp
$0x00490b0 <msg1>: "Hello, "
(gdb) x/1b $msg2
$0x00490b0 <msg2>: "world!\n"error: Cannot access memory at address 0x00490c7>
(gdb) p/t $esp
No registers
(gdb) p/s $esp
$0x00490b0 <msg1>: "Hello, "
(gdb) p/t $esp
No registers
(gdb) p/s $esp
$0x00490b0 <msg1>: "Hello, "
(gdb) p/t $esp
No registers
(gdb) 

```

Рис. 15: Примеры использования команды set

4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. 16).

The terminal window shows the following commands being run:

```

retrofisnov@tx-d1:~/work/arm-pc/lab09$ cp ~/work/arm-pc/lab08/lab08-2.asm ~/work/arm-pc/lab09/lab09-2.asm
retrofisnov@tx-d1:~/work/arm-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
retrofisnov@tx-d1:~/work/arm-pc/lab09$ ld -n elf_i386 -o lab9-3 lab9-3.o
retrofisnov@tx-d1:~/work/arm-pc/lab09$ 

```

Рис. 16: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились.

(рис. 17).

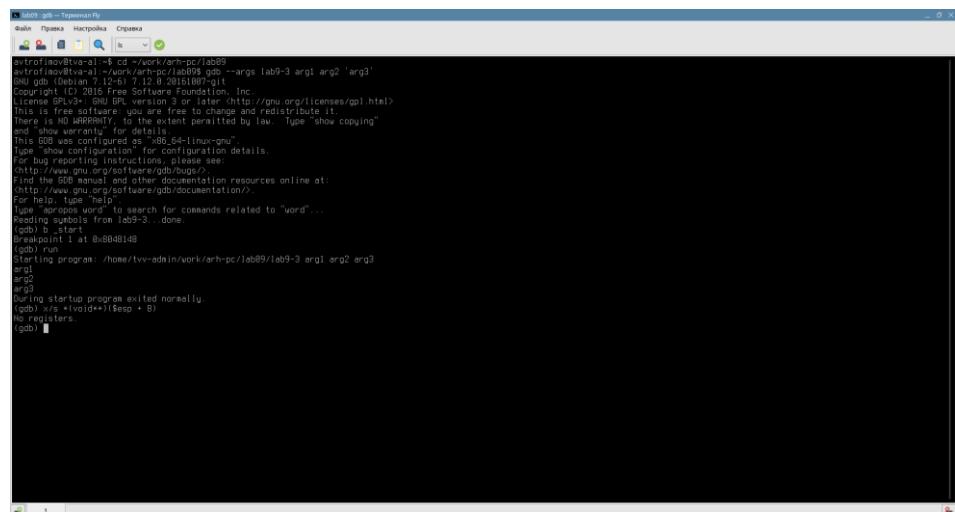


Рис. 17: Проверка работы стека

4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 18).

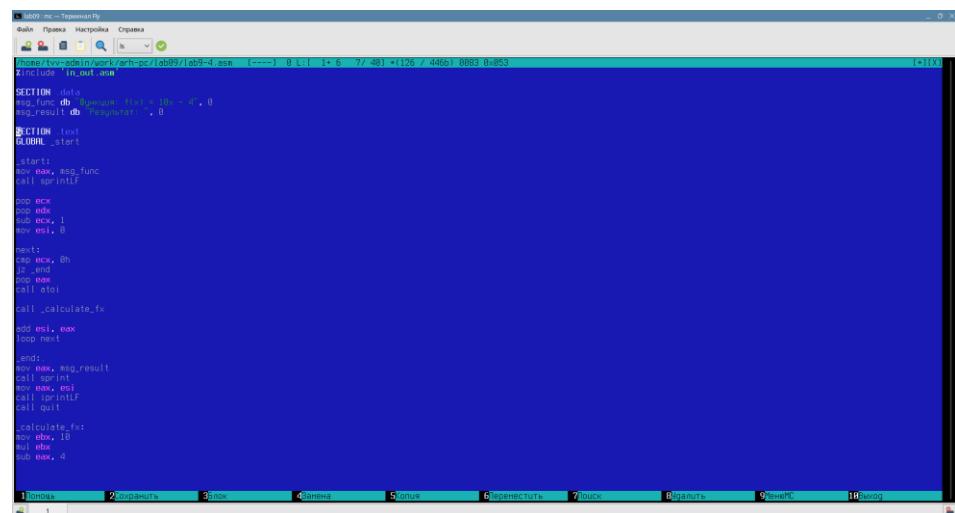


Рис. 18: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data  
msg_func db "Функция: f(x) = 10x - 4", 0
```

```
msg_result db "Результат: ", 0
```

```
SECTION .text  
GLOBAL _start
```

```
_start:
```

```
    mov eax, msg_func  
    call sprintLF
```

```
    pop ecx  
    pop edx  
    sub ecx, 1  
    mov esi, 0
```

```
next:
```

```
    cmp ecx, 0h  
    jz _end  
    pop eax  
    call atoi
```

```
    call _calculate_fx
```

```
    add esi, eax  
    loop next
```

```
_end:
```

```
    mov eax, msg_result  
    call sprint  
    mov eax, esi  
    call iprintLF  
    call quit
```

```
_calculate_fx:
```

```
    mov ebx, 10  
    mul ebx  
    sub eax, 4
```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через і г. При выполнении инструкции mul есх можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программы неверно подсчитывает функцию (рис. 19).

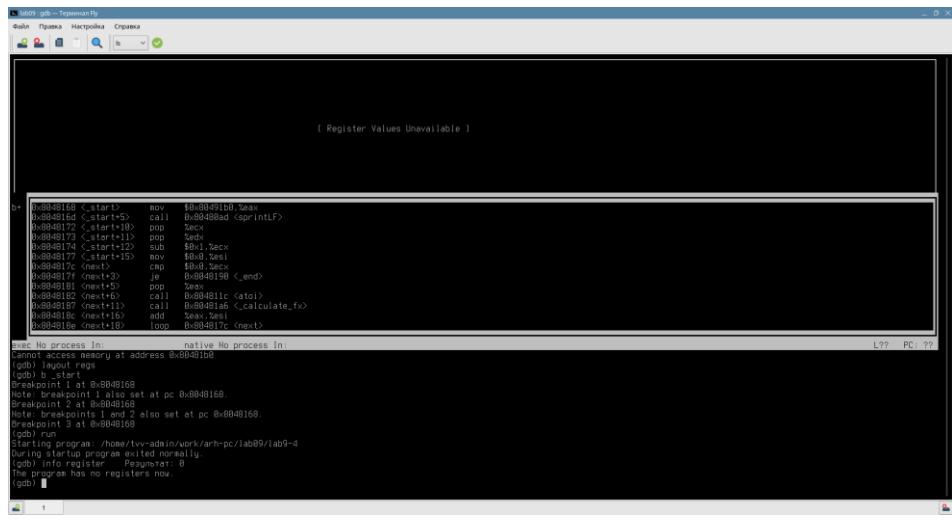


Рис. 19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 20).

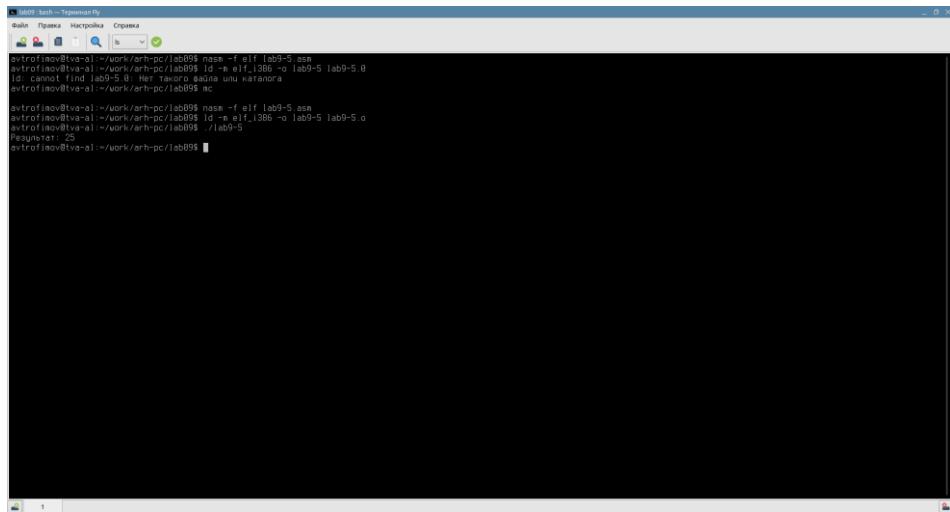


Рис. 20: Проверка корректировок в программме

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
div: DB 'Результат: ', 0
```

```
SECTION .text
GLOBAL _start
_start:
```

```
mov ebx, 3
mov eax, 2
add ebx, eax
mov eax, ebx
mov ecx, 4
mul ecx
add eax, 5
mov edi, eax
```

```
mov eax, div
call sprint
mov eax, edi
call iprintLF
```

```
call quit
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №9
3. Программирование на языке ассемблера NASM Столяров А. В.