Лабораторная работа №9

Понятие подпрограммы.Отладчик GDB.

Казначеев Сергей Ильич

Содержание

Список иллюстраций

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 создадим папку и файл lab9-1.asm

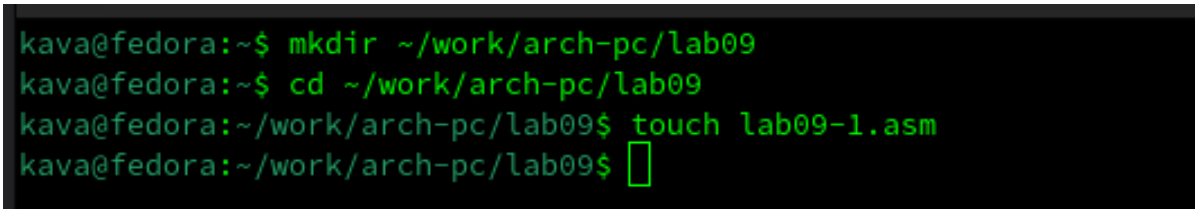

Рис. 2.1: 01

Далее, запустим Midnight commander и копируем файл in_out.asm.

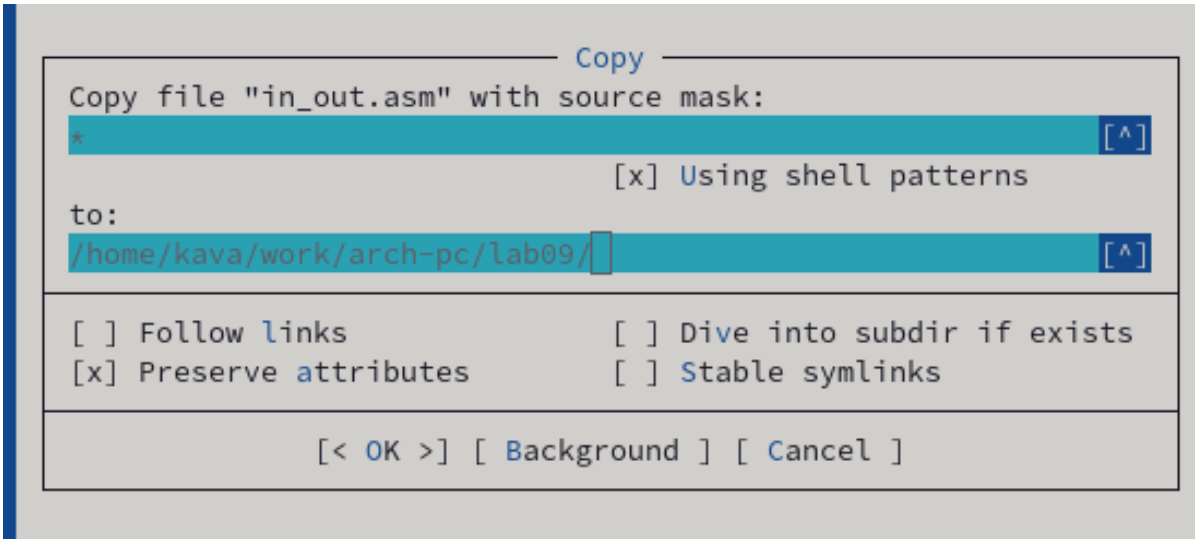

Рис. 2.2: 02

Вставляем в файл lab9-1.asm код из листинга 9.1

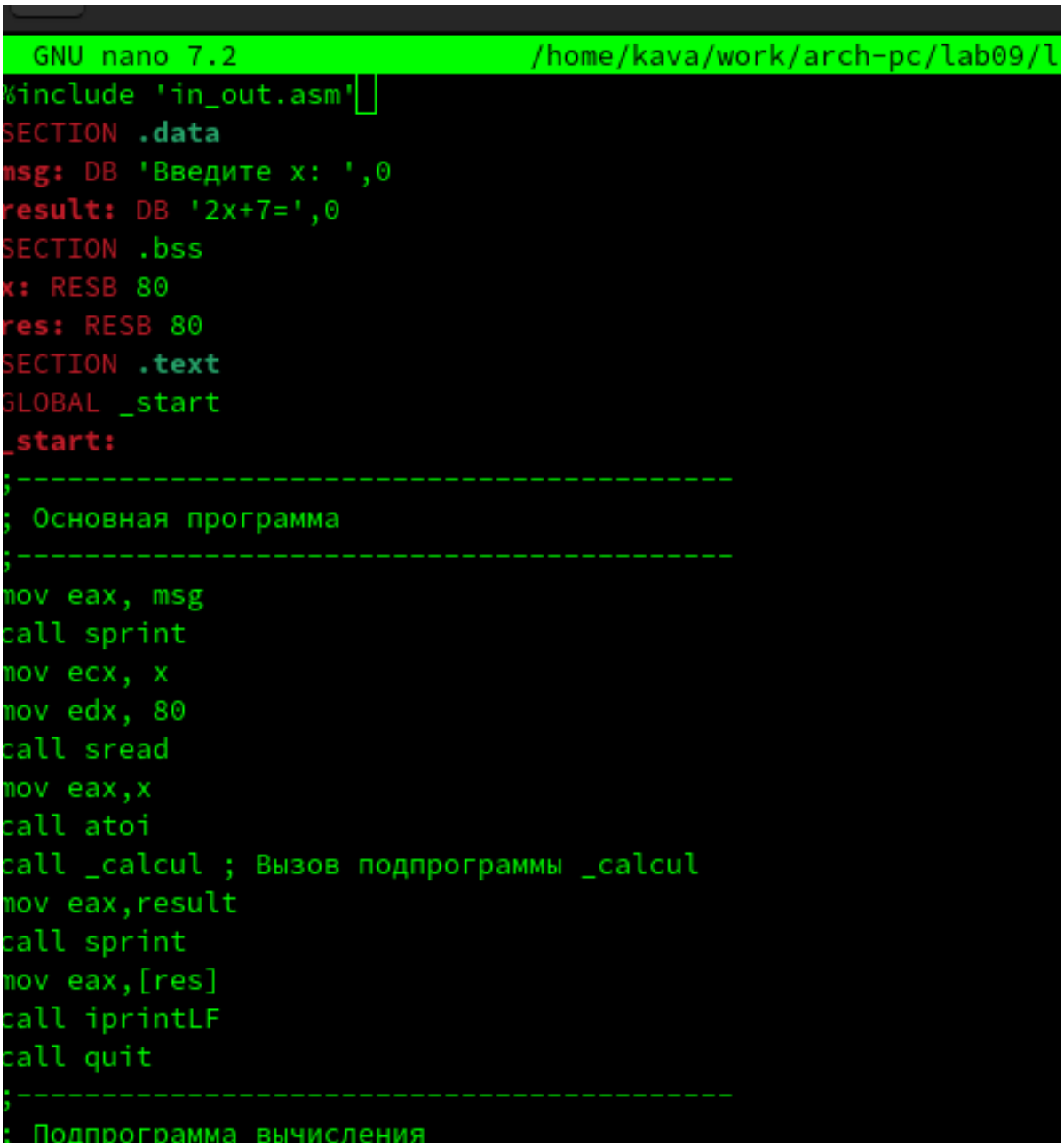

Рис. 2.3: 03

Собираем программу и проверяем её на корректность работы

```
kava@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
kava@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
kava@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
kava@fedora:~/work/arch-pc/lab09$
```

Рис. 2.4: 04

Теперь изменим файл так, чтобы внутри подпрограммы была ещё одна подпрограмма, вычисляющая значение g(x) и чтобы она передавала значение в первую подпрограмму, которая бы уже вычислила значение f(g(x))

```
;-----------------------------------------
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret

ret ; выход из подпрограммы
```

Рис. 2.5: 05

Собираем и проверяем

```
kava@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
kava@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
kava@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 1
f(g(x))=11
kava@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
f(g(x))=23
kava@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 2
f(g(x))=17
```

Рис. 2.6: 06

Создаем второй файл

```
kava@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
kava@fedora:~/work/arch-pc/lab09$
```

Рис. 2.7: 07

И вставляем код из листинга 9.2

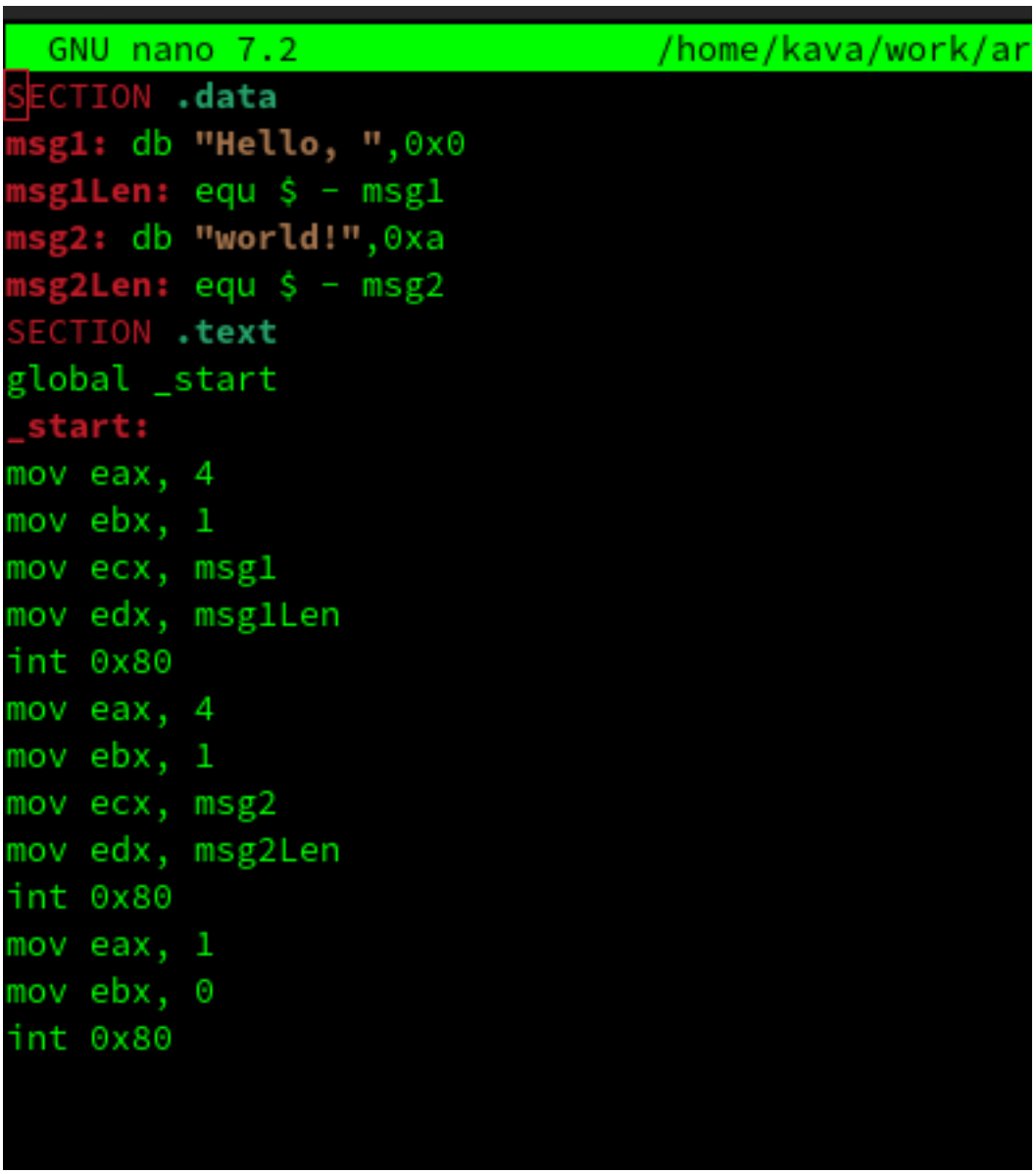

Рис. 2.8: 08

Соберем программу следующим образом

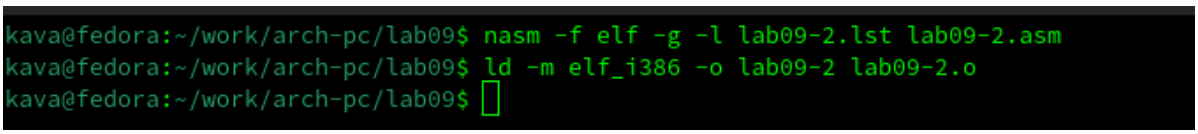

Рис. 2.9: 09

Теперь загружаем ее в gdb

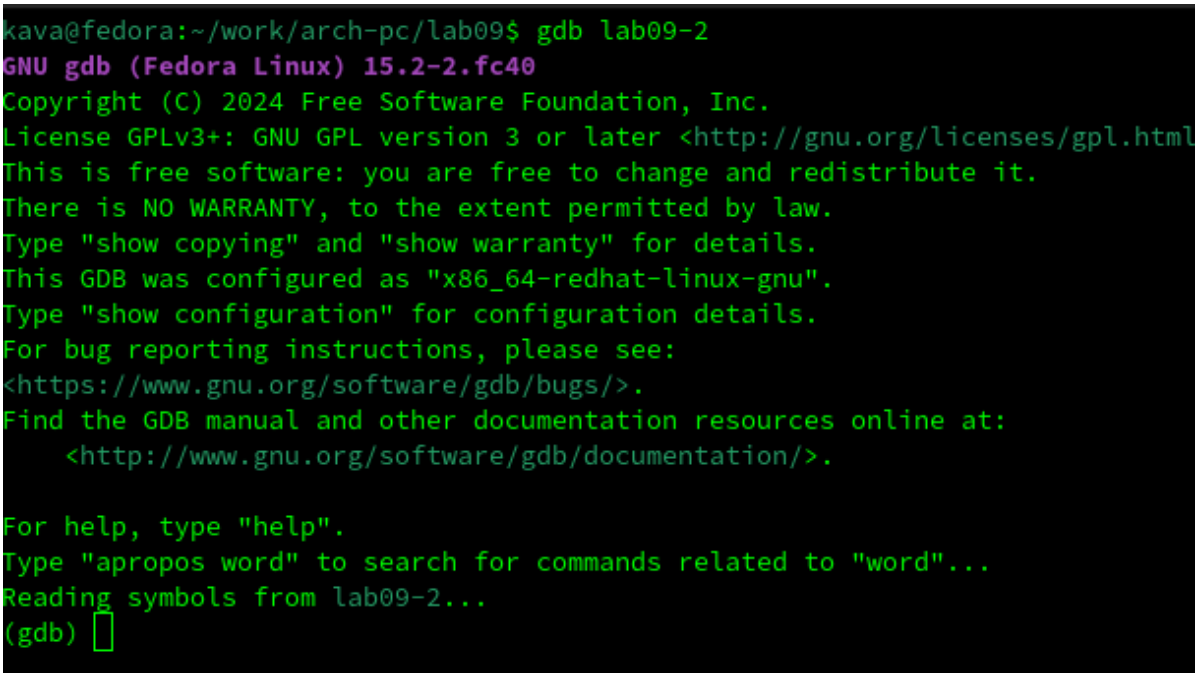

Рис. 2.10: 10

Запускаем ее с помощью команды run

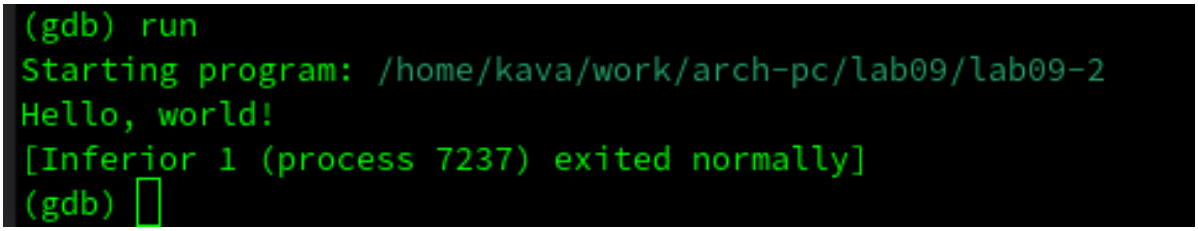

Рис. 2.11: 11

Создадим брейкпоинт на метке _start с помощью команды break

Рис. 2.12: 12

С помощью команды disassemble дизассемблируем её

Рис. 2.13: 13

Переключаем вывода синтекса на intel

Рис. 2.14: 14

Повторяем команду disassemble

Рис. 2.15: 15

Включаем графическое отображение кода

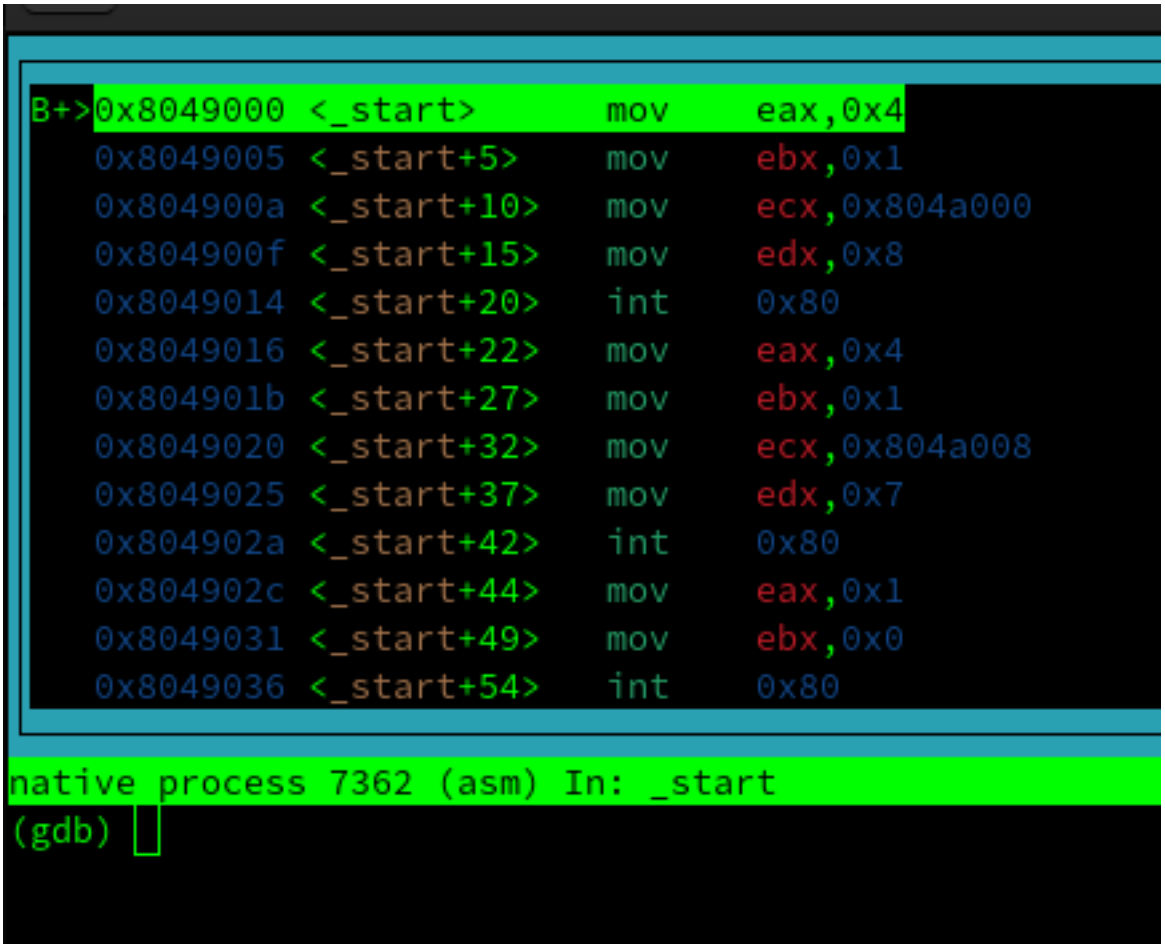

Рис. 2.16: 16

Теперь включаем графическое отображение значений регистров

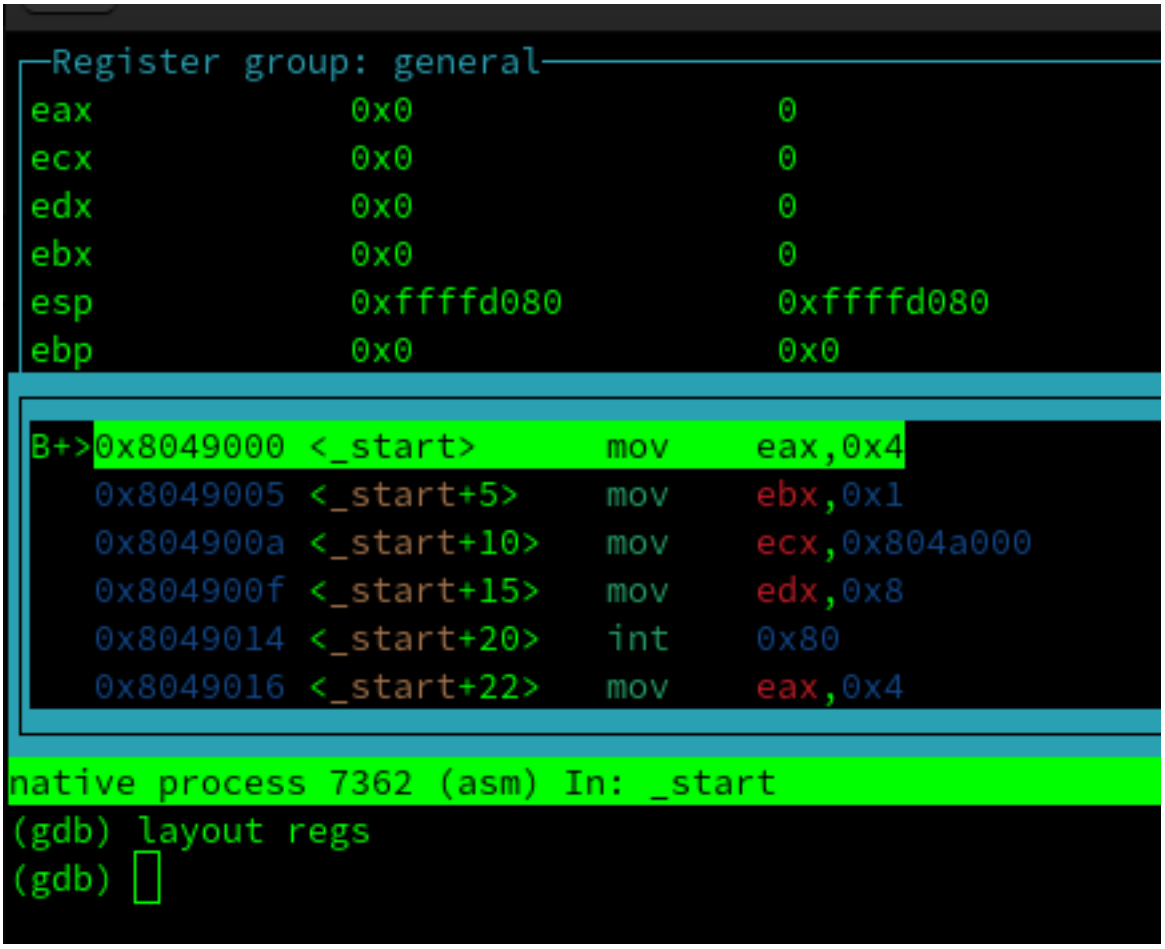

Рис. 2.17: 17

Выводим всю информацию о всех брейкпоинтах

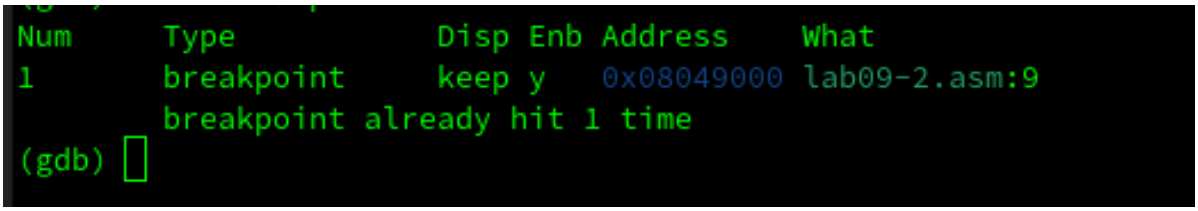

Рис. 2.18: 18

Создаем брейкпоинт по адресу

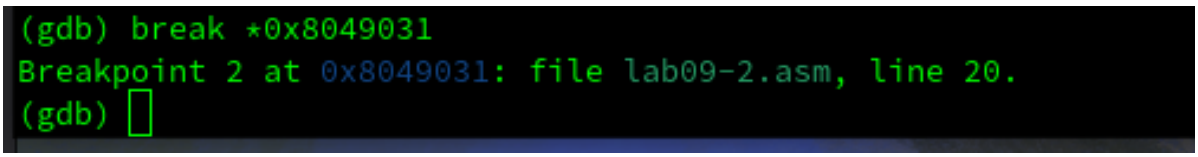

Рис. 2.19: 19

Выводим информацию

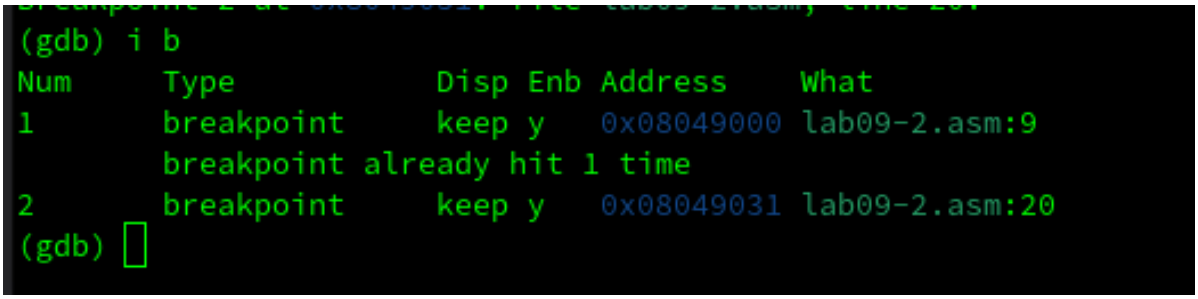

Рис. 2.20: 20

Теперь 5 раз выполняем команду si для построчного выполнения кода

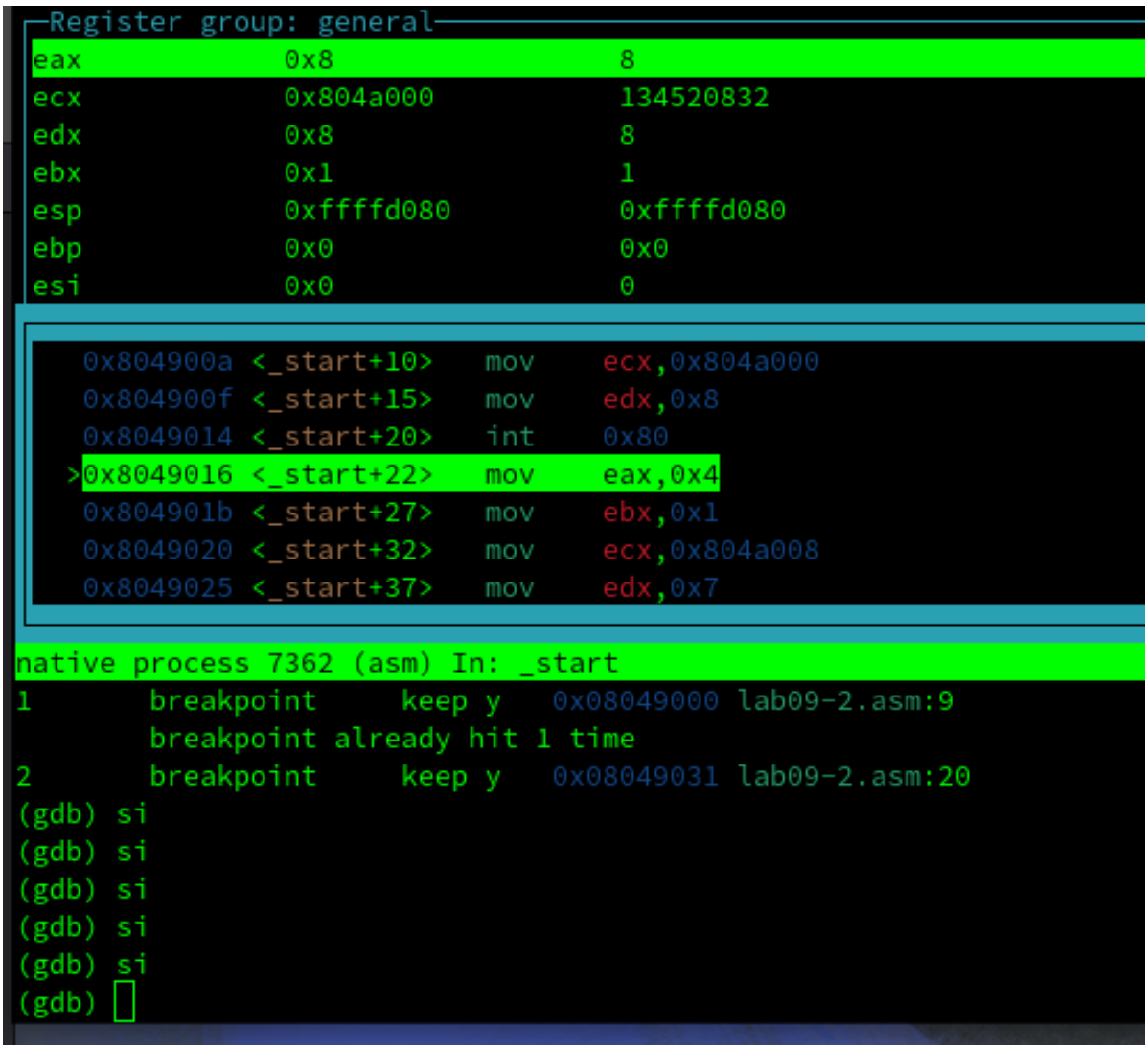

Рис. 2.21: 21

Как видим, поменялись значения регистров eax, ecx, edx и ebx. Теперь выведем информацию о значениях регистров

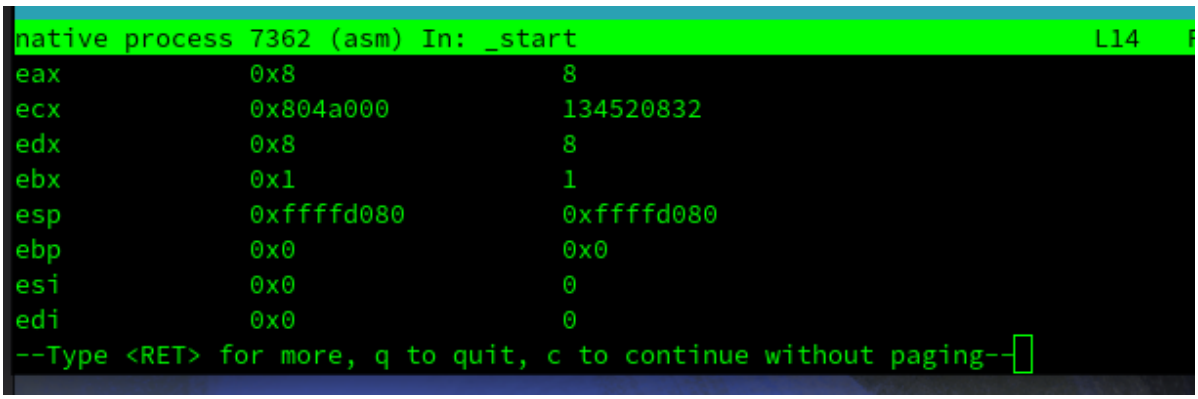

Рис. 2.22: 22

Выводим значения переменной по имени.

Рис. 2.23: 23

Теперь по адресу

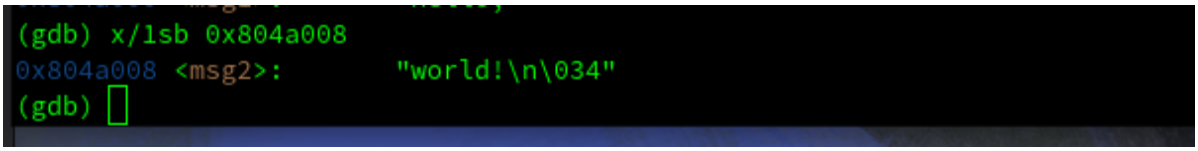

Рис. 2.24: 24

Теперь меняем первый символ

Рис. 2.25: 25

Меняем второй символ

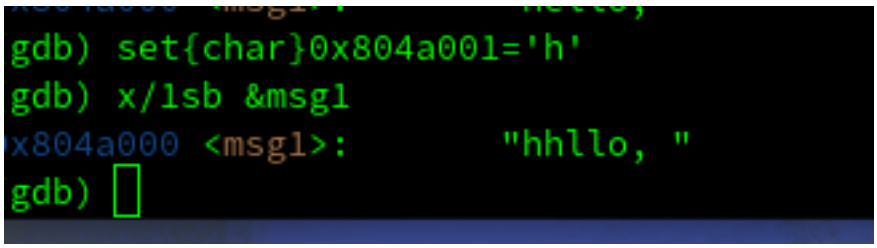

Рис. 2.26: 26

Меняем несколько символов второй переменной

Рис. 2.27: 27

Выводим значения регистра в строковом,двоичном и шестнадцатиричном виде

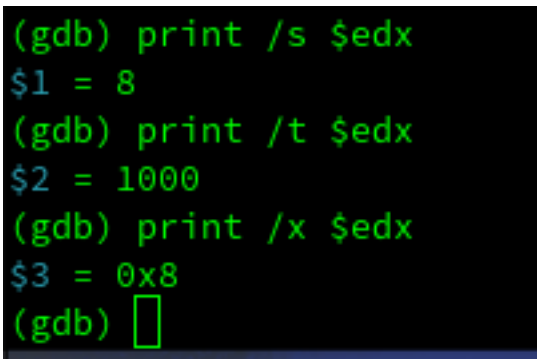

Рис. 2.28: 28

Пробуем изменить значения регистра

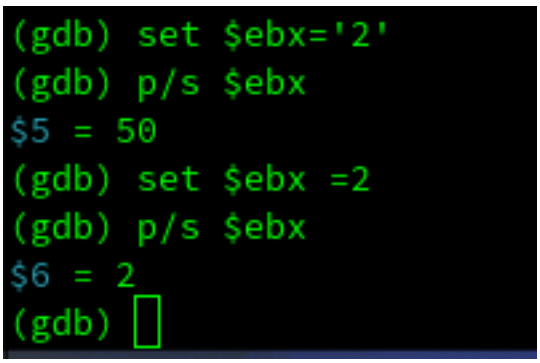

Рис. 2.29: 29

Мы увидим что в регистр записались разные значения, это связано с тем, что в одном случае мы записываем в него число, а в другом случае строчку. Завершаем программу с помощью команды continue и выйдем.

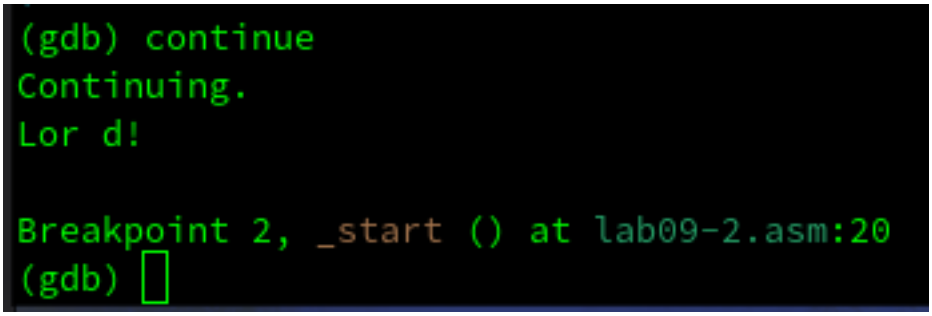

Рис. 2.30: 30

Рис. 2.31: 31

Копируем файл

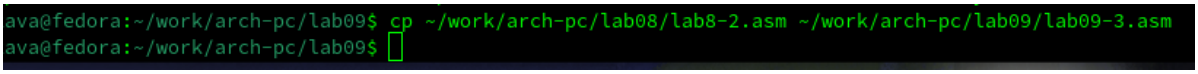

Рис. 2.32: 32

Соберём его и вгрузим в gdb

```
kava@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
kava@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
kava@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 15.2-2.fc40
Copyright (C) 2024 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/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb)
```

Рис. 2.33: 33

Создадим брейкпоинт и запустим программу

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/kava/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 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, _start () at lab09-3.asm:5
5       pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)
```

Рис. 2.34: 34

Выведем значение регистра esp, где хранятся данные о стеке

```
(gdb) x/x $esp
0xffffd030:     0x00000005
(gdb)
```

Рис. 2.35: 35

ВЫведем значения всех элементов стека

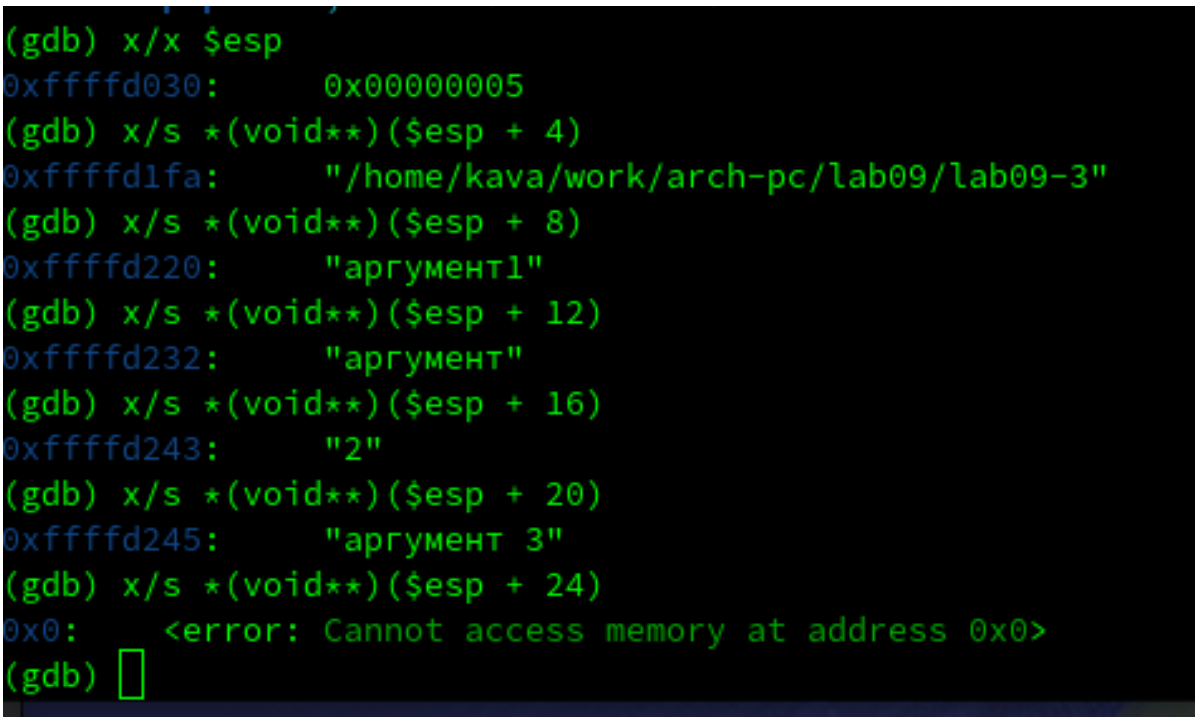

Рис. 2.36: 36

Как видим, для вывода каждого элемента стека нам нужно менять значение адреса с шагом 4. Это связано с тем, что именно с шагом 4 располагаются данные в стеке.

3 Выполнение заданий для самостоятельной работы

Копируем файл первого задания прошлой самостоятельной работы

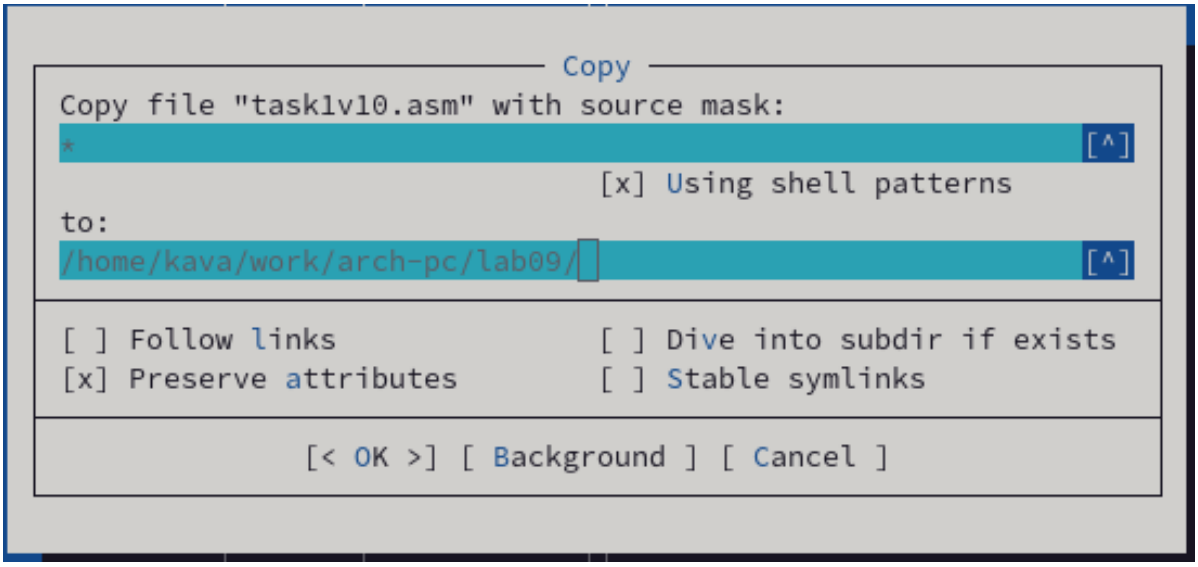

Рис. 3.1: 37

Переписываем так, чтобы он использовался для авчисления выражения подпрограммы

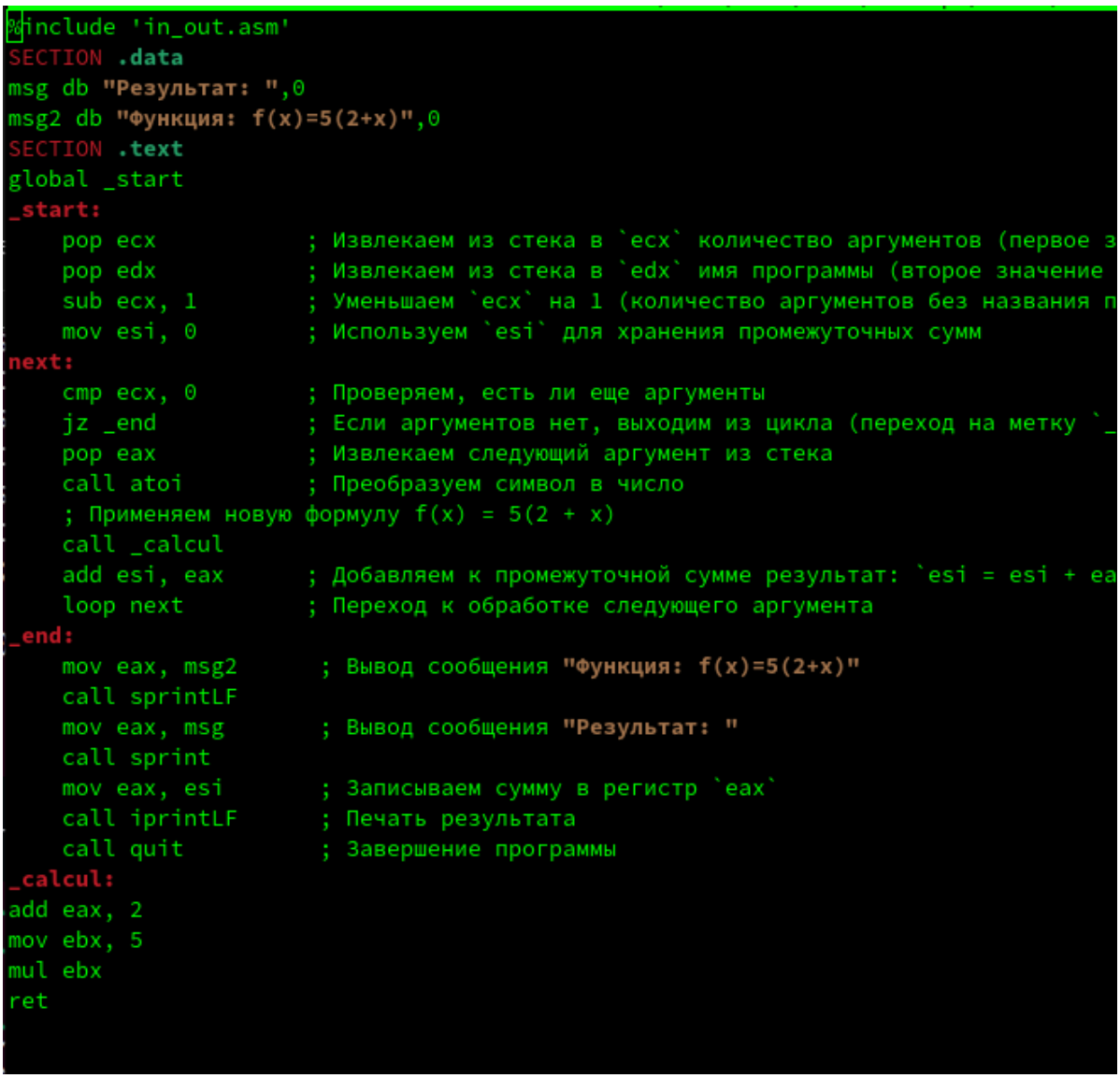

Рис. 3.2: 38

Собираем и проверяем на корректность выполнения.

```
kava@fedora:~/work/arch-pc/lab09$ nasm -f elf task1v10.asm
kava@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o task1v10 task1v10.o
kava@fedora:~/work/arch-pc/lab09$ ./task1v10 1 2 3 4
Функция: f(x)=5(2+x)
Результат: 90
kava@fedora:~/work/arch-pc/lab09$ ./task1v10 1
Функция: f(x)=5(2+x)
Результат: 15
kava@fedora:~/work/arch-pc/lab09$ ./task1v10 1 5
Функция: f(x)=5(2+x)
Результат: 50
```

Рис. 3.3: 39

Создадим файл второго задания самостоятельной работы

```
kava@fedora:~/work/arch-pc/lab09$ touch task2.asm
kava@fedora:~/work/arch-pc/lab09$
```

Рис. 3.4: 40

Далее вставляем код из листинга 9.3

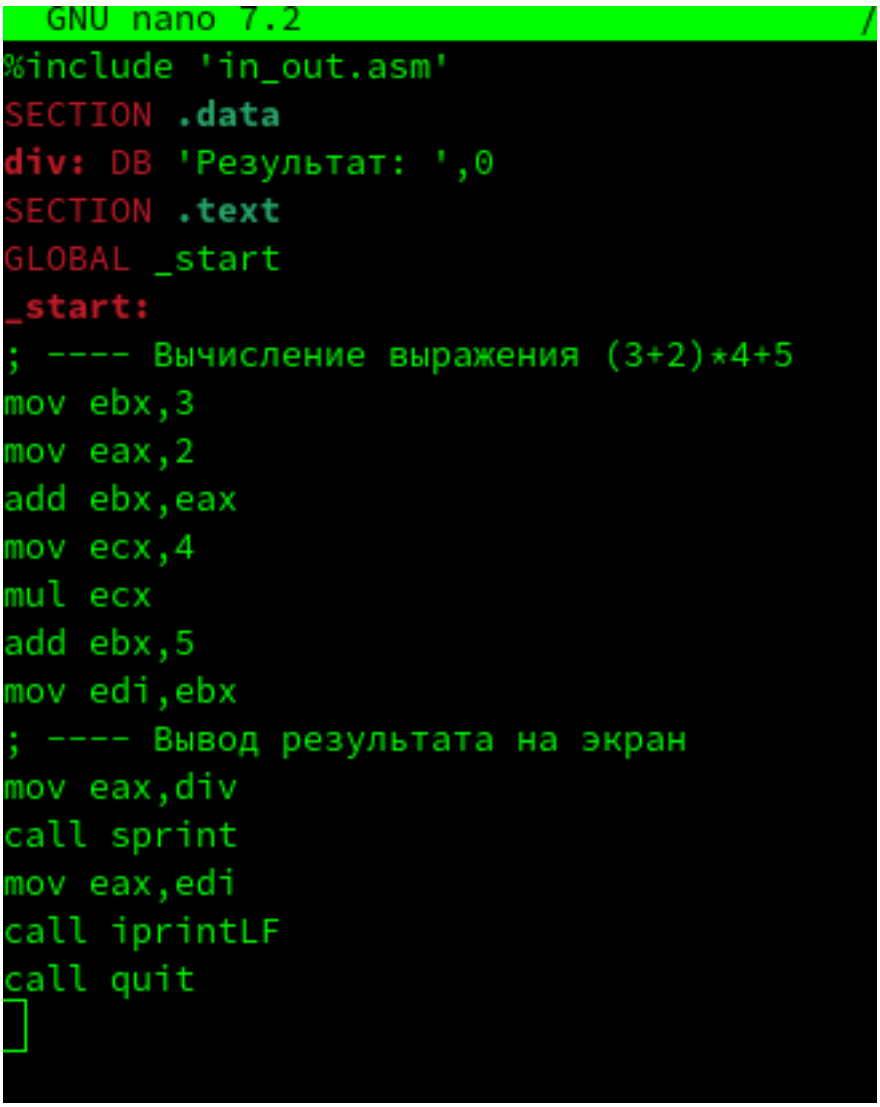

Рис. 3.5: 41

Собираем и запускуаем

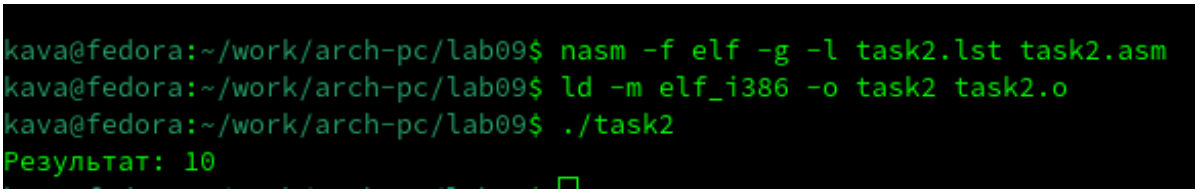

Рис. 3.6: 42

Как видим, код считает значение выражения неправильно. Загрузим его в gdb.

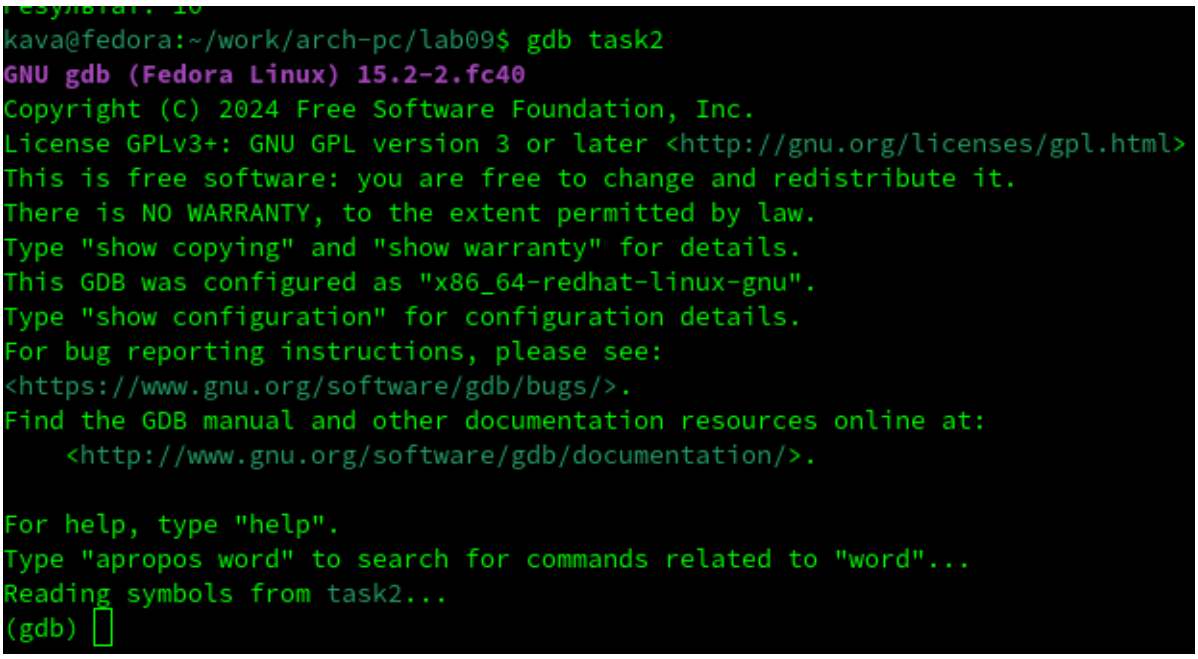

Рис. 3.7: 43

Включеним графическое отображение значений регистров и отображение графического отображения кода.

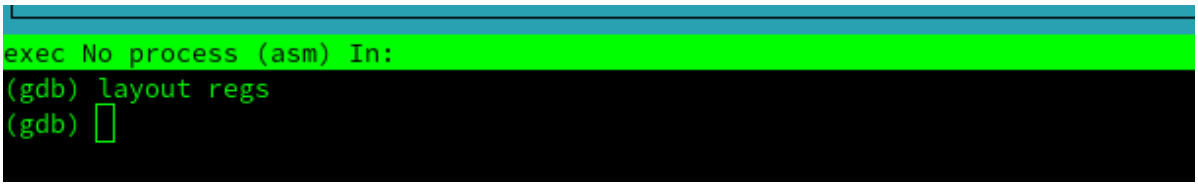

Рис. 3.8: 44

Устанавливаем брейкпоинт на _start

Рис. 3.9: 45

Запускаем и начинаем построчно выполнять код

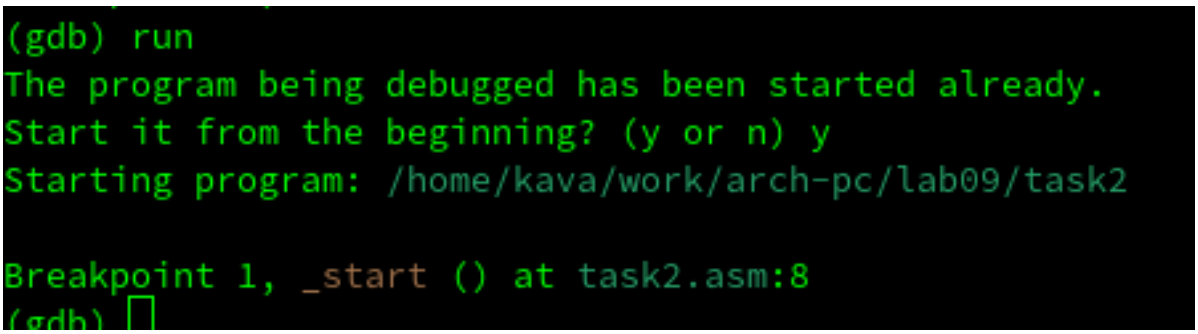

Рис. 3.10: 46

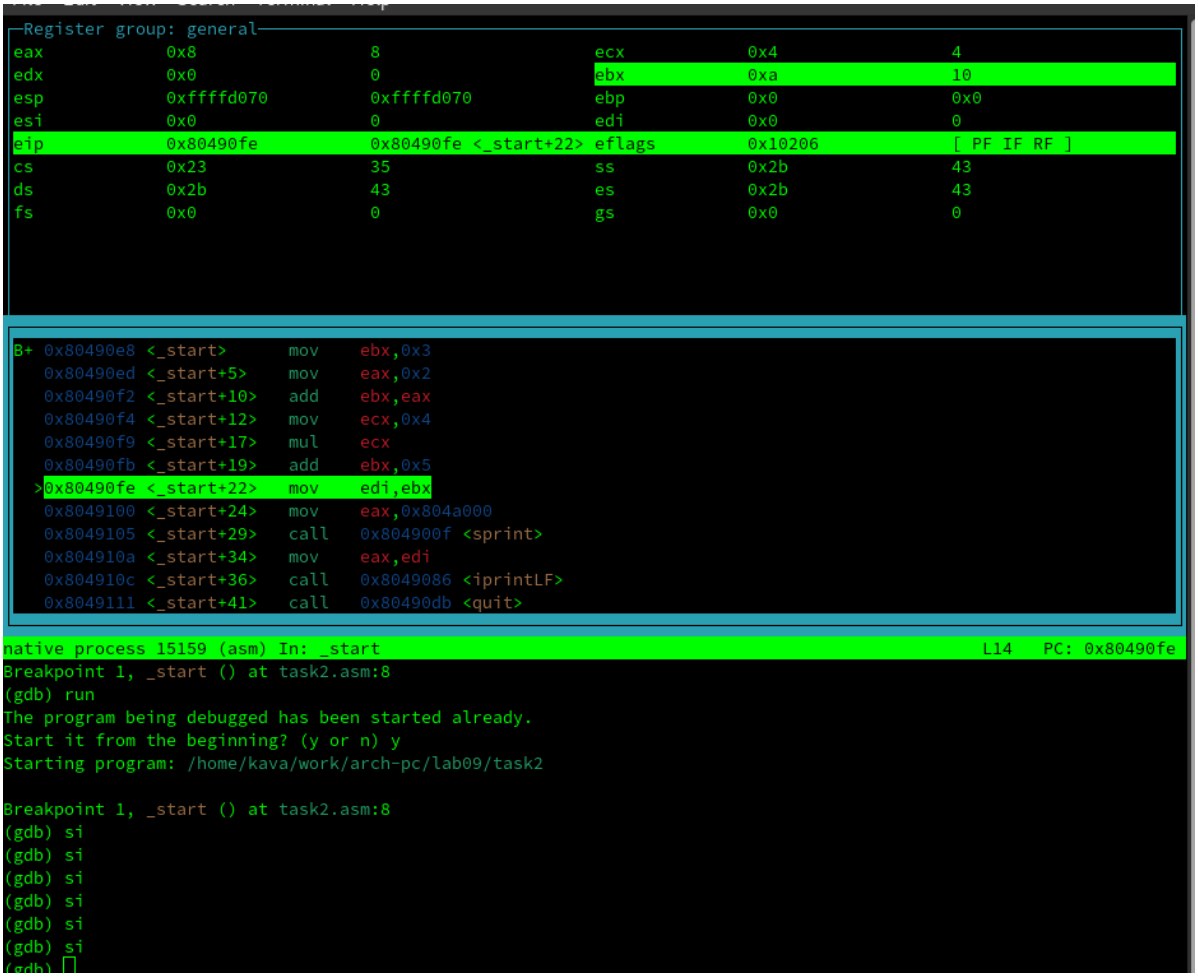

Рис. 3.11: 47

Как видим, мы должны были умножить значение регистра ebx, но умножили регистр eax. Нам необходимо все результаты хранить в регистре eax. Изменим код

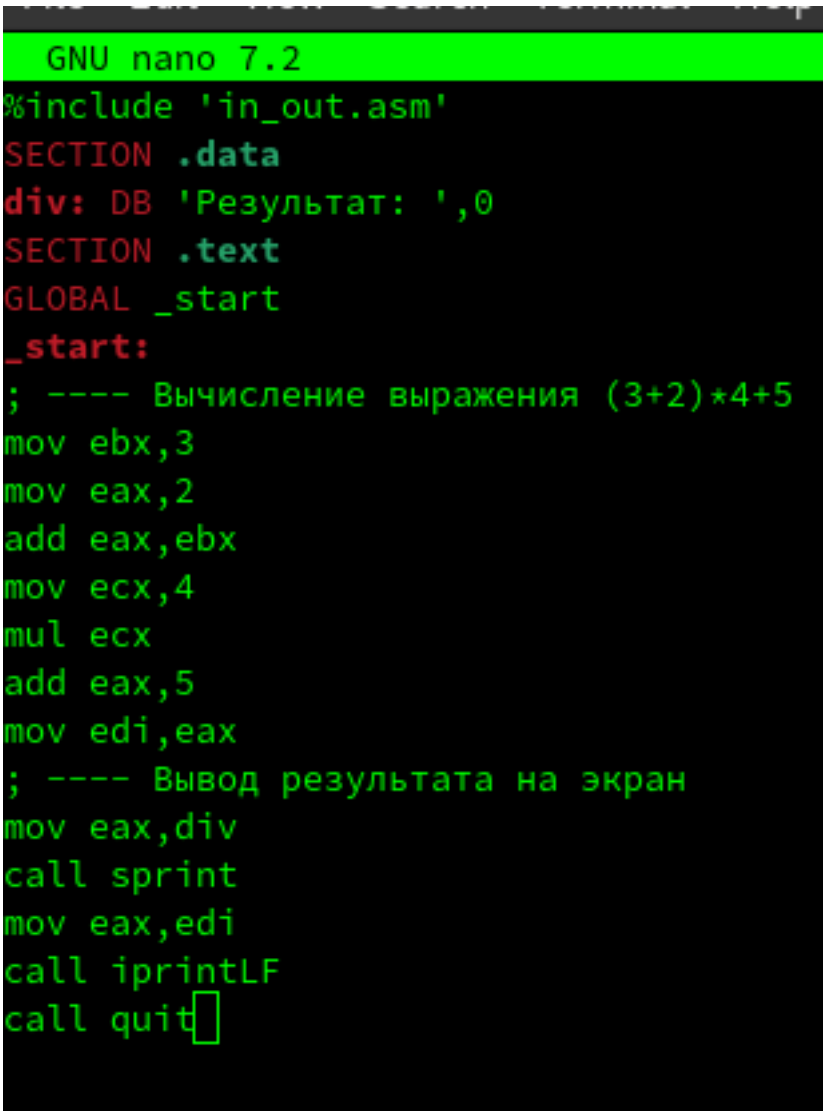

Рис. 3.12: 48

И проверяем на корректность выполнения.

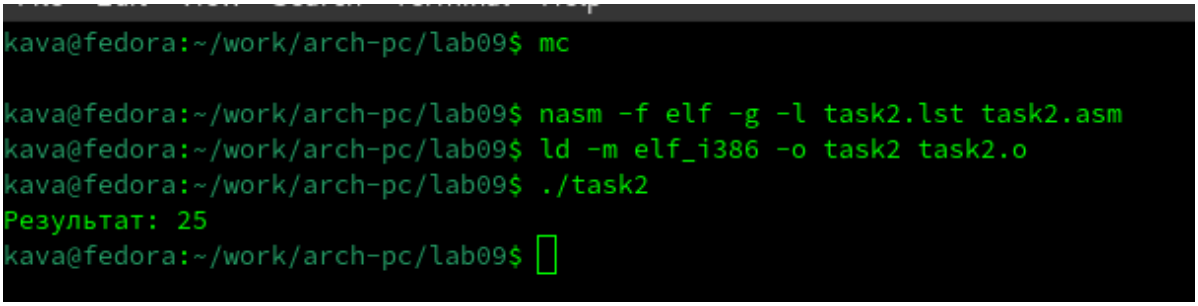

Рис. 3.13: 49

Как видим, теперь код работает корректно

4 Выводы

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