Отчет по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Краснопер Данила Олегович

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

1) Я создал каталог lab9 и создал файл lab9-1.asm

```
dokrasnoper@dk3n62 ~ $ mkdir ~/work/arch-pc/lab09
dokrasnoper@dk3n62 ~ $ cd ~/work/arch-pc/lab09
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ touch lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ls
lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Создание каталога и файла

2)Я ввел текст листинга в файл и запустил программу.

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+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 ; Вызов подпрограммы _calcul
mov eax,result
call sprint
mov eax,[res]
call iprintLF
call quit
;----------------------------------------
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
```

Рис. 2.2: Текст программы

```
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 5
2x+7=17
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $
```

Рис. 2.3: Работа программы

3) Я изменил текст программы, чтобы она решала выражение f(g(x)).

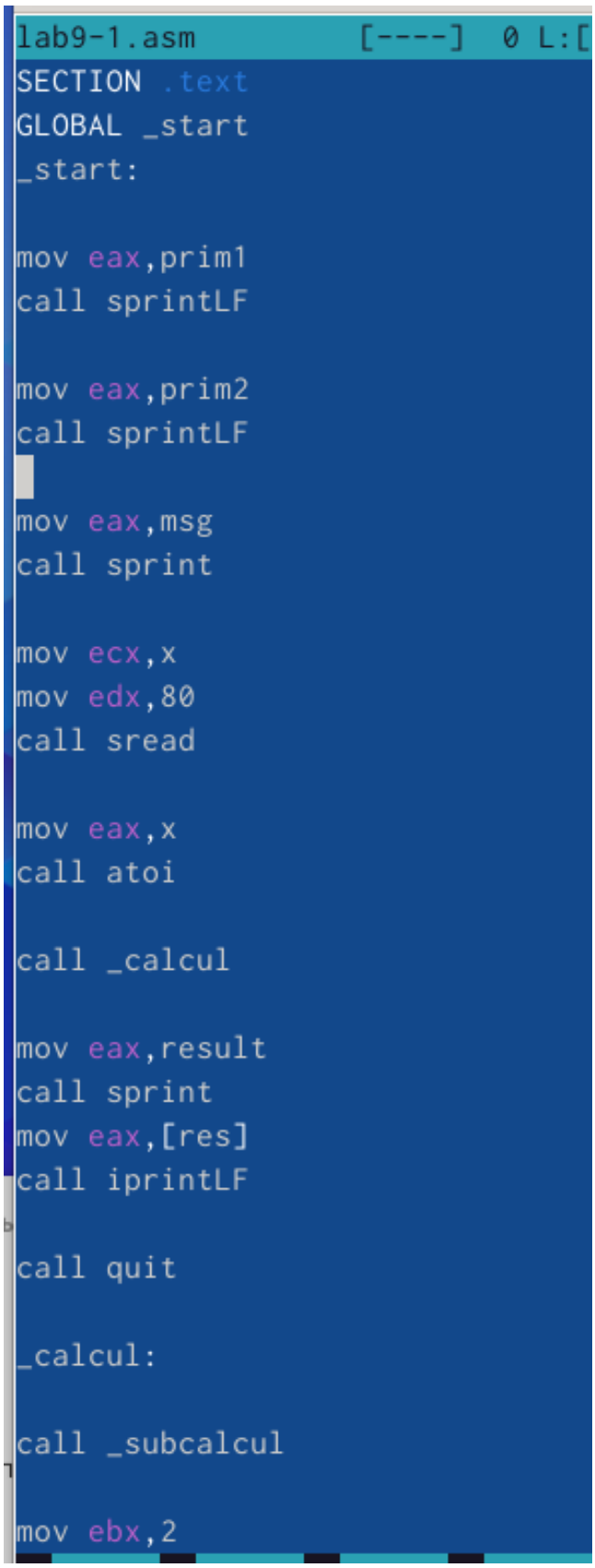


Рис. 2.4: Измененный текст программы

```
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ./lab9-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 1
f(g(x))= 11
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $
```

Рис. 2.5: Проверка работы программы

4)Я создал файл lab9-2.asm и вписал туда программу.

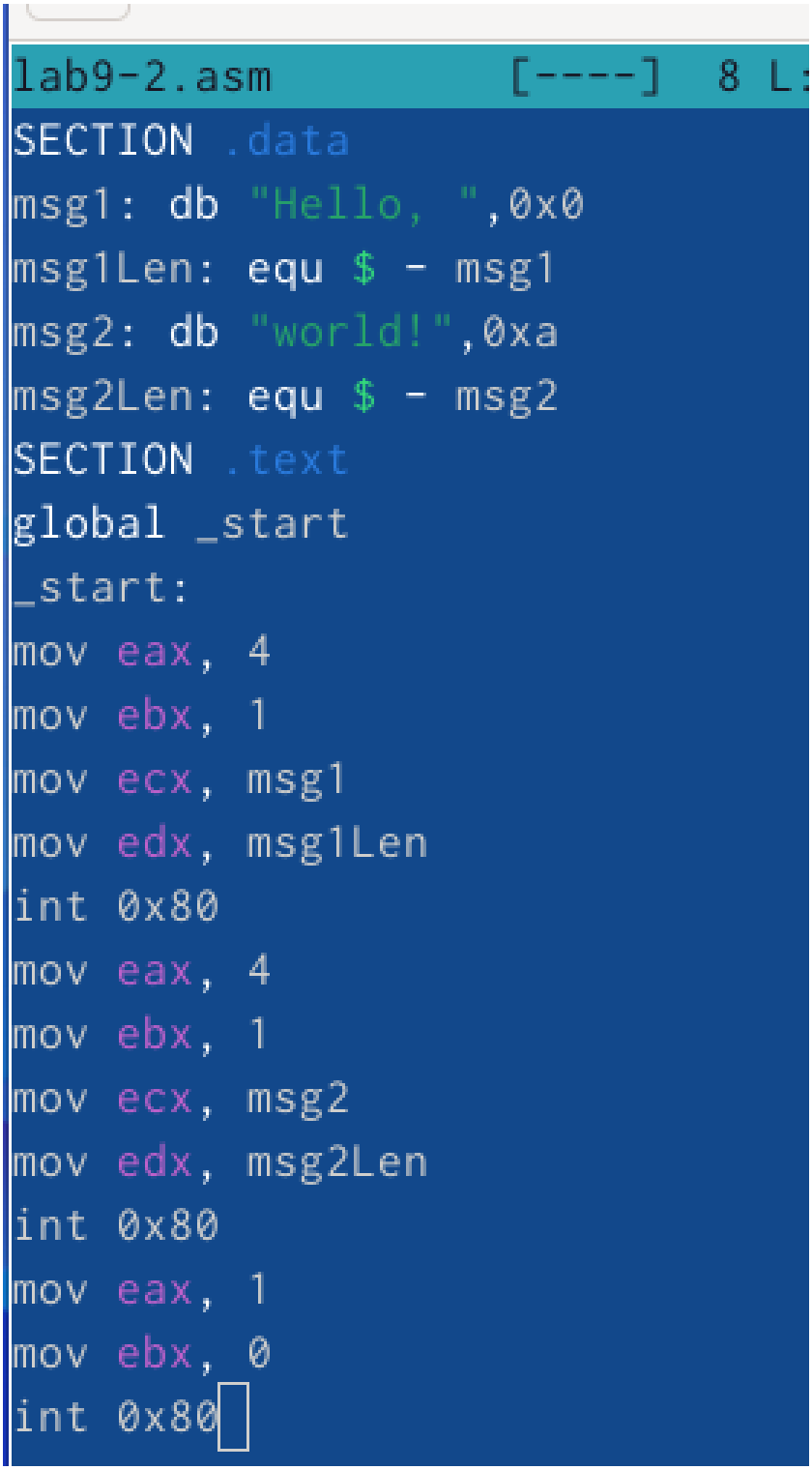


Рис. 2.6: Текст второй программы

5)Я загрузил и запустил файл второй программы в отладчик gdb.

```
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ touch lab9-2.asm
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ mc

dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.a
m
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ d -m elf_i386 -o lab9-2 lab9-2.o
bash: d: команда не найдена
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
dokrasnoper@dk3n62 ~/work/arch-pc/lab09 $ gdb lab9-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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/d/o/dokrasnoper/work/arch-pc/lab
9/lab9-2
Hello, world!
[Inferior 1 (process 4978) exited normally]
(gdb)
```

Рис. 2.7: Отладка второго файла

6) Я поставил брекпоинт на метку _start и запустил программу.

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab0
9/lab9-2

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

Рис. 2.8: Брекпоинт на метку _start

7)Я просмотрел дисассимплированный код программы начиная с метки.

Рис. 2.9: Дисассимплированный код

8) С помощью команды я переключился на intel'овское отображение синтаксиса. Отличие заключается в командах, в диссамилированном отображении в командах используют % и $, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

```
End of assembler dump.
(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
```

Рис. 2.10: Intel'овское отображение

9) Для удобства я включил режим псевдографики.

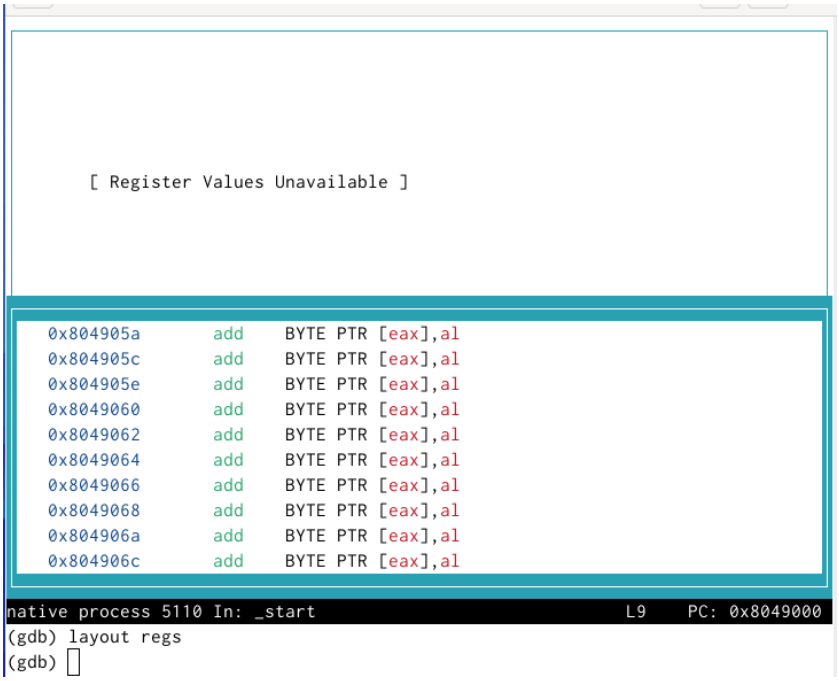


Рис. 2.11: Псевдографика

10) Я посмотрел наличие меток и добавил еще одну метку на предпоследнюю инструкцию.

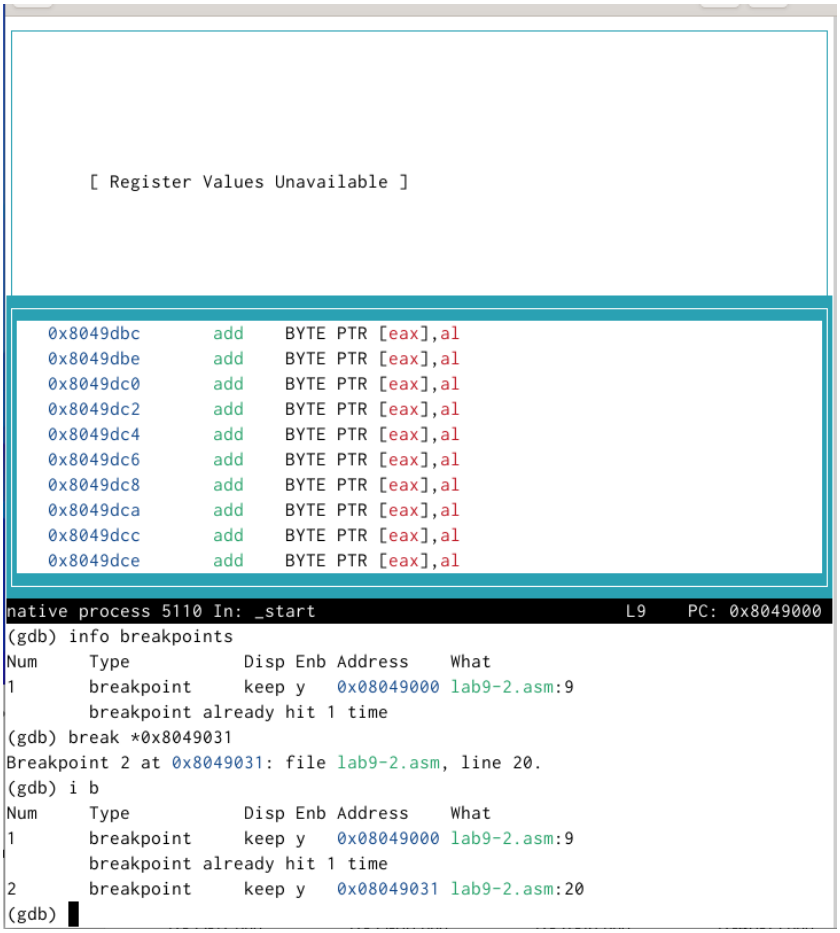


Рис. 2.12: Наличие меток

11) С помощью команды si я посмотрел регистры и изменил их.

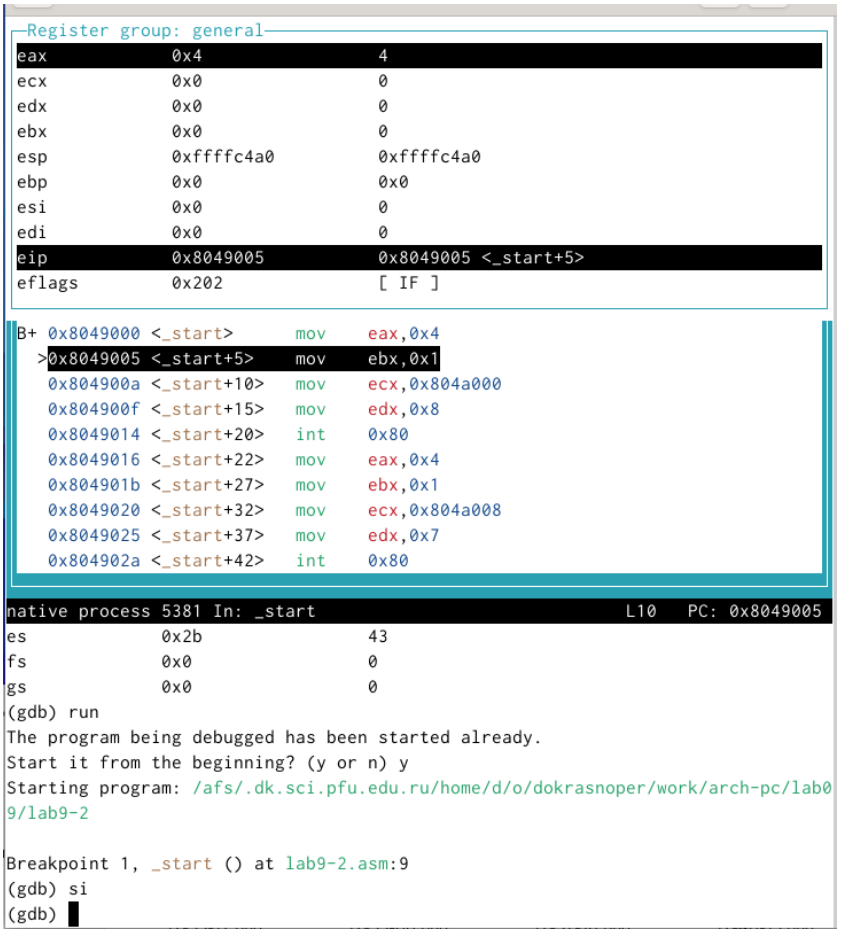


Рис. 2.13: Просмотр регистров

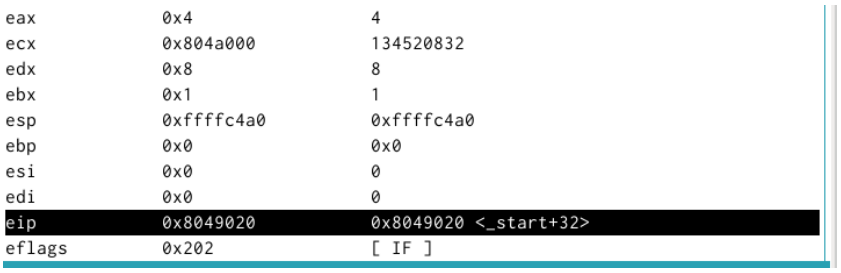


Рис. 2.14: Измененные регистры

12) С помощью команды я посмотрел значение переменной msg1.

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

Рис. 2.15: Просмотри значения переменной

13) Следом я посмотрел значение второй переменной msg2.

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

Рис. 2.16: Значение переменной msg2

14) С помощью команды set я изменил значение переменной msg1.

```
(gdb) set {char}0x804a000='0'
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/lsb &msg1
0x804a000 <msg1>:       "hhllo, "
(gdb)
```

Рис. 2.17: Изменение значения переменной

15)Я изменил переменную msg2.

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/lsb &msg2
0x804a008 <msg2>:       "Lor d!\n\034"
(gdb)
```

Рис. 2.18: Изменение msg2

16)Я вывел значение регистров ecx и eax.

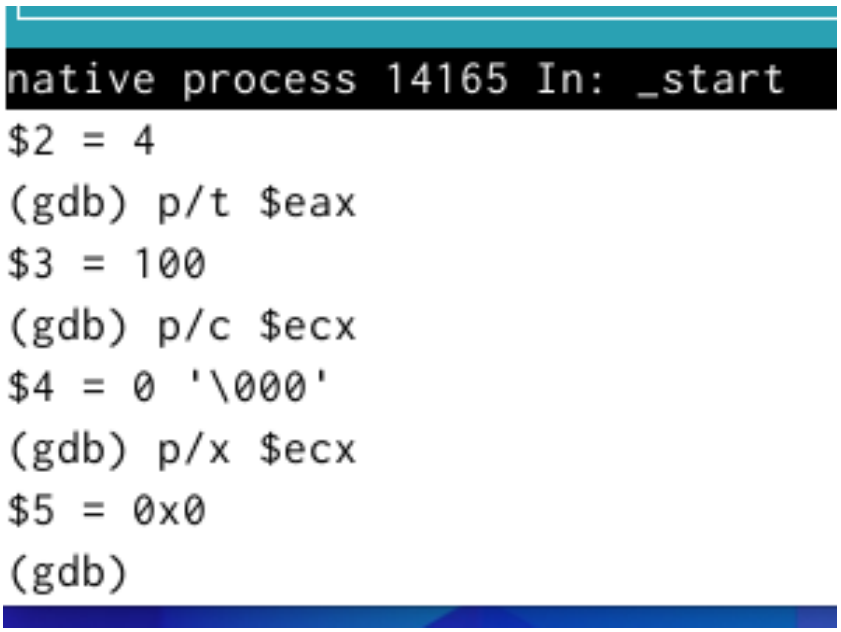


Рис. 2.19: Значение регистров ecx и eax

17) Я изменил значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.

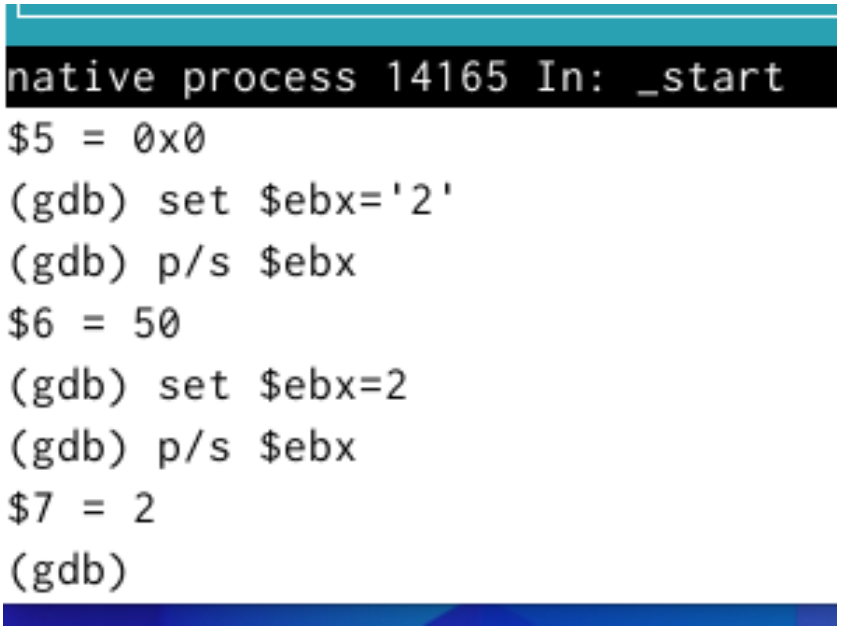


Рис. 2.20: Значение регистров ebx

18) Я завершил работу с файлов вышел.

```
   0x08049031 <+49>:    mov    ebx,0x0
   0x08049036 <+54>:    int    0x80
End of assembler dump.
(gdb) layout regs
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $
```

Рис. 2.21: Завершение работы с файлов

19) Я скопировал файл lab8-2.asm и переименовал его. Запустил файл в отладчике и указал аргументы.

```
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ gdb --args lab9-3 аргумент1 аргумент 2
 'аргумент 3'
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-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)
```

Рис. 2.22: Запуск файла в отладчике

20) Поставил метку на _start и запустил файл.

```
(gdb) b _start
Breakpoint 1 at 0x8049000: file lab9-3.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/o/dokrasnoper/work/arch-pc/lab0
9/lab9-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:9
9       mov eax, 4
(gdb)
```

Рис. 2.23: Запуск файла lab10-3 через метку

21)Я проверил адрес вершины стека и убедился что там хранится 5 элементов.

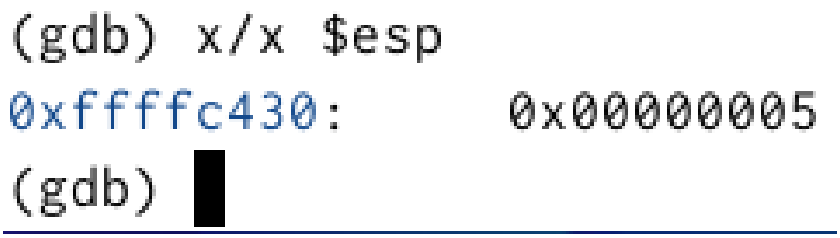


Рис. 2.24: Адрес вершины стека

22) Я посмотрел все позиции стека. По первому адрему хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

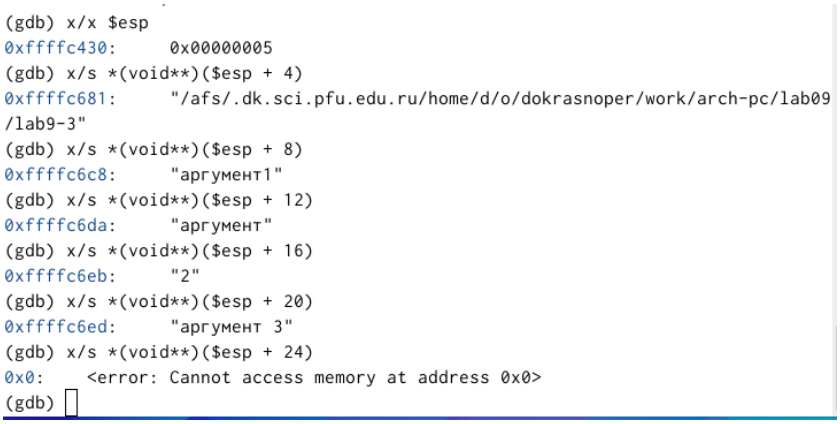
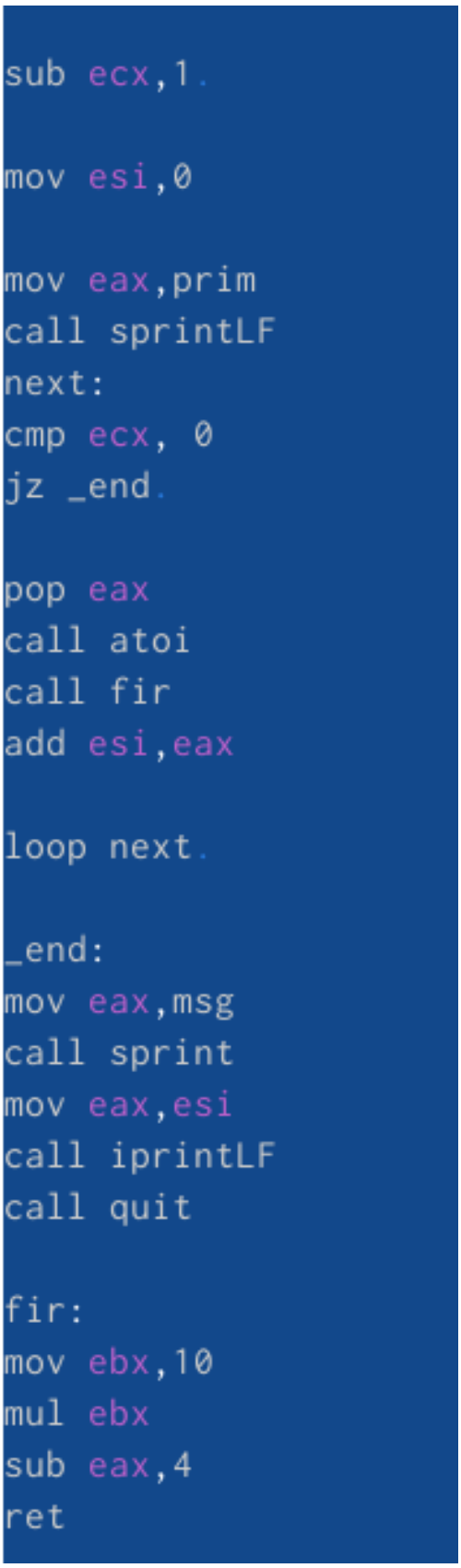


Рис. 2.25: Все позиции стека

3 Самостоятельная работа

1) Я преобразовал программу из лабораторной работы №8 и реализовал вычисления как подпрограмму.



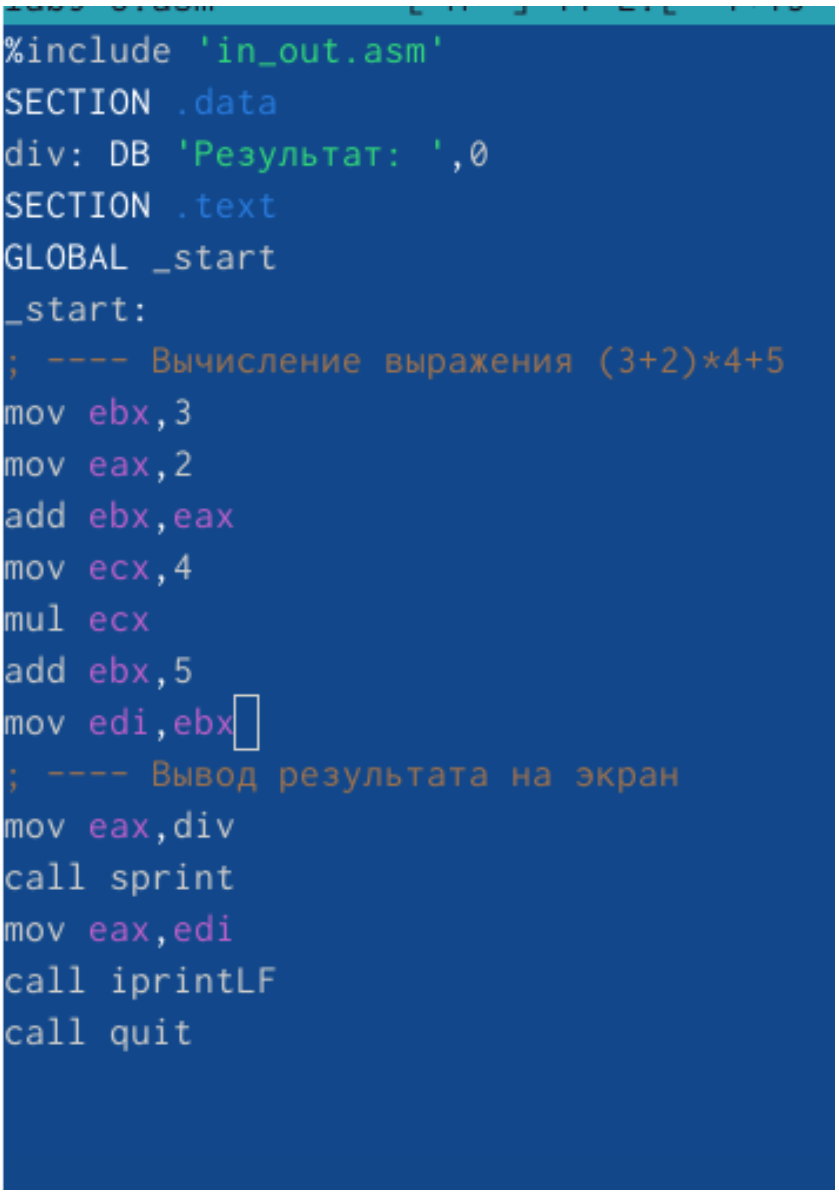
```
sub ecx,1.

mov esi,0

mov eax,prim
call sprintLF
next:
cmp ecx, 0
jz _end.

pop eax
call atoi
call fir
add esi,eax

loop next.

_end:
mov eax,msg
call sprint
mov eax,esi
call iprintLF
call quit

fir:
mov ebx,10
mul ebx
sub eax,4
ret
```

Рис. 3.1: Текст программы

```
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-4 2 3 5
f(x)=15x-9
Результат: 88
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-4 1 1 1
f(x)=15x-9
Результат: 18
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $
```

Рис. 3.2: Запуск программы

2) Я переписал программу и попробовал запустить ее чтобы увидеть ошибку. Ошибка была арифметическая, так как вместо 25,программа выводит 10.



```
%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 ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 3.3: Текст программы

```
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10
dokrasnoper@dk3n55 ~/work/arch-pc/lab09 $
```

Рис. 3.4: Запуск программы

После появления ошибки, я запустил программу в отладчике.

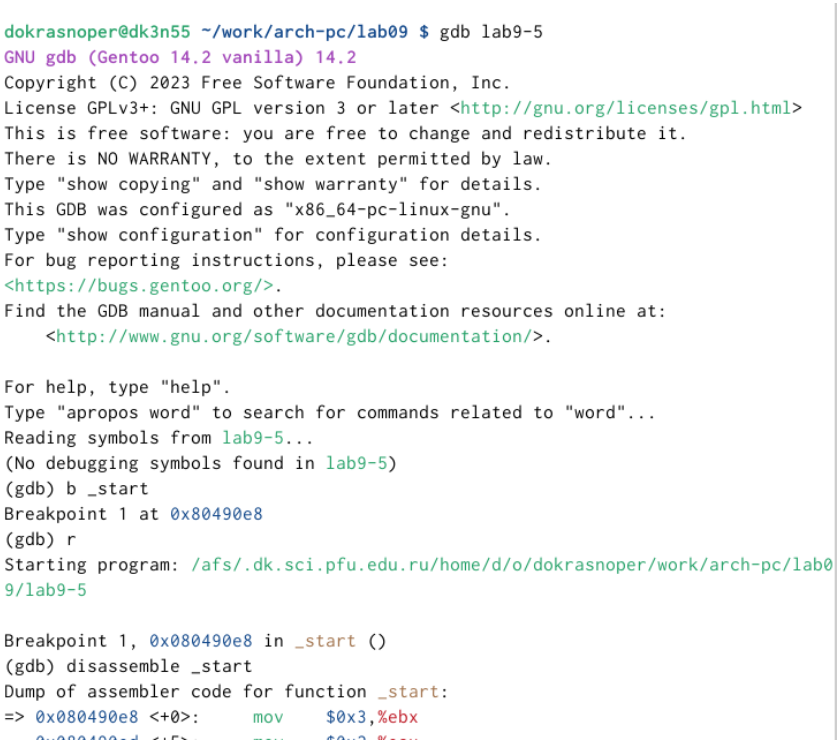


Рис. 3.5: Запуск программы в отладчике

Я открыл регистры и проанализировал их, понял что некоторые регистры стоят не на своих местах и исправил это.

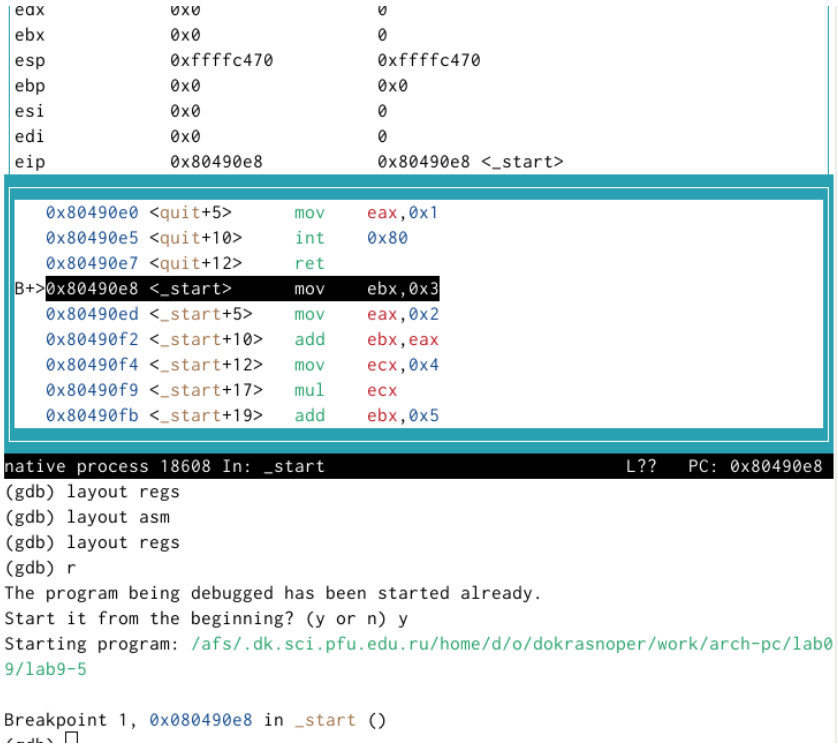


Рис. 3.6: Анализ регистров

Я изменил регистры и запустил программу, программа вывела ответ 25, то есть все работает правильно.

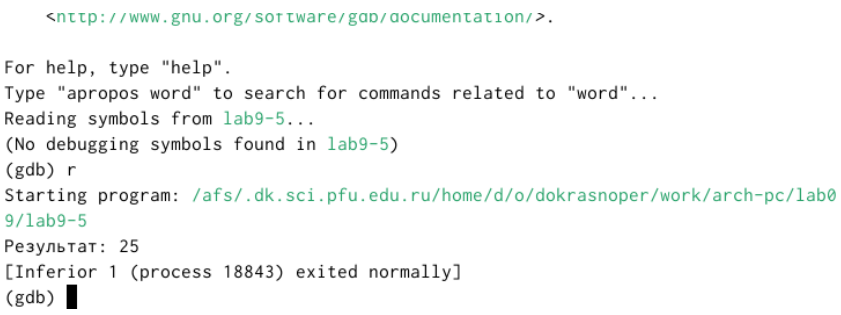


Рис. 3.7: Повторный запуск программы

Вывод

Я приобрел навыки написания программ использованием подпрограмм. Познакомился с методами отладки при помозь GDB и его основными возможностями.