Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB

Сырцева Анастасия Романовна

Содержание

Список иллюстраций

1 Цель работы

Целью данной работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Задание

- Ознакомление с подпрограммами
- Изучение отладки при помощи GDB

3 Выполнение лабораторной работы

Создаю папку для данной лабораторной работы. Перехожу в неё и создаю lab9-1.asm(рис. 3.1).

```
arsihrceva@dk2n21 ~/work/arch-pc $ mkdir lab09
arsihrceva@dk2n21 ~/work/arch-pc $ cd lab09
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ touch lab9-1.asm
```

Рис. 3.1: Создание рабочего каталога и файла

Внимательно изучаю листинг программы с использованием вызова подпрограммы. Для корректной работы необходим внешний файл in_out.asm. Копирую его в рабочую папку(рис. 3.2).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ cp ~/work/in_out.asm ~/work/arch-pc/lab09/in_out.asm
```

Рис. 3.2: Копирование внешнего файла

Ввожу в файл lab9-1.asm текст данной программы(рис. 3.3).

```
lab9-1.asm          [-M--]  7
%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

mov eax,result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:
    mov ebx,2
    mul ebx
    add eax,7
    mov [res],eax
....
    ret
```

Рис. 3.3: Программа с использованием вызова подпрограммы

Создаю и запускаю исполняемый файл. Проверяю работу программы(рис. 3.4).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 7
2x+7=21
```

Рис. 3.4: Результат работы программы

Изменяю программу так, чтобы исходная функция была зависима от другой функции. Для этого добавляю ещё одну подпрограмму в исходную подпрограмму(рис. 3.5).

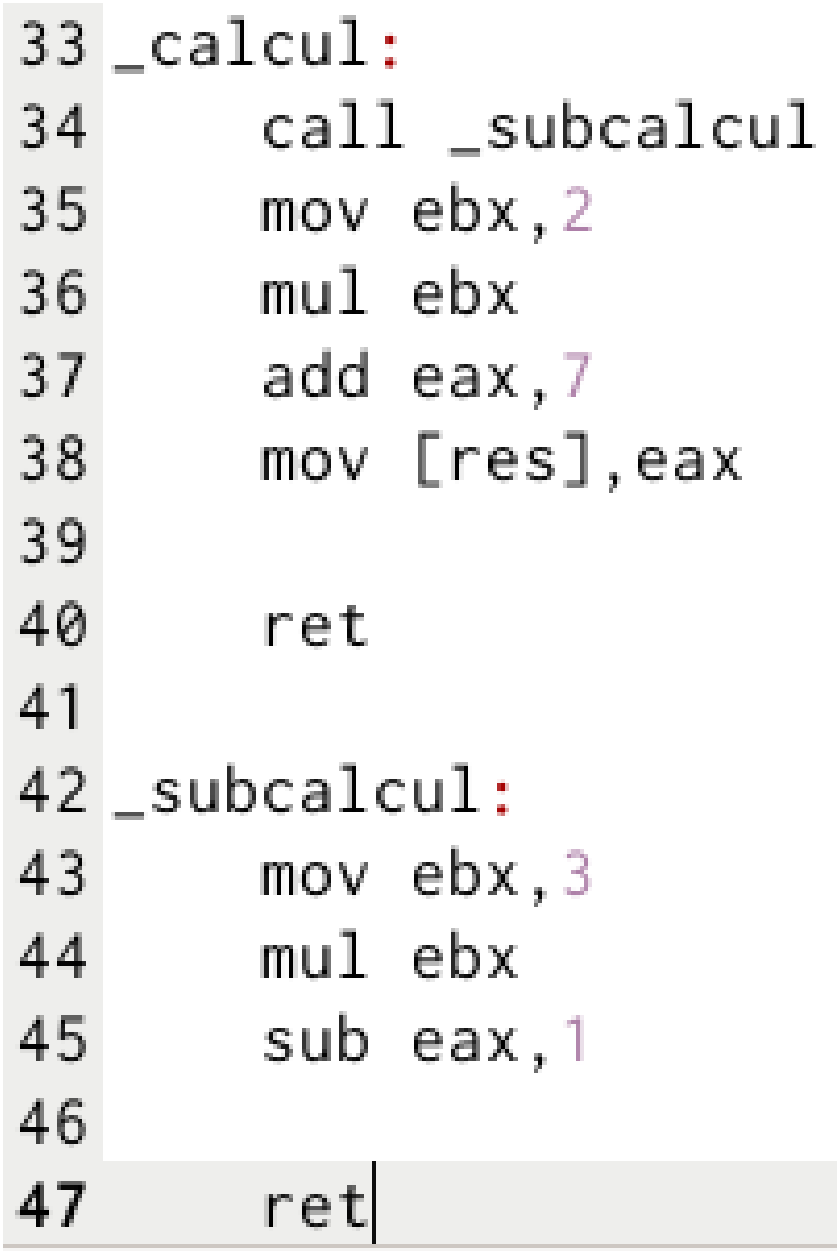

Рис. 3.5: Изменённая часть текста программы

Создаю и запускаю исполняемый файл изменённой программы(рис. 3.6).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 2
2(3x-1)+7=17
```

Рис. 3.6: Запуск и результат изменённой программы

Создаю новый файл lab9-2.asm((рис. 3.7).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ touch lab9-2.asm
```

Рис. 3.7: Создание нового рабочего файла

Ввожу в файл текст программы вывода сообщения Hello world!(рис. 3.8).

```
 1 SECTION .data
 2         msg1: db "Hello, ",0x0
 3         msg1Len: equ $ - msg1
 4
 5         msg2: db "world!",0xa
 6         msg2Len: equ $ - msg2
 7
 8 SECTION .text
 9         global _start
10
11 _start:
12         mov eax,4
13         mov ebx,1
14         mov ecx,msg1
15         mov edx,msg1Len
16         int 0x80
17
18         mov eax,4
19         mov ebx,1
20         mov ecx,msg2
21         mov edx,msg2Len
22         int 0x80
23
24         mov eax,1
25         mov ebx,0
26         int 0x80
```

Рис. 3.8: Текст программы вывода сообщения

Создаю исполняемый файл для работы с GDB(рис. 3.9).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
```

Рис. 3.9: Создание исполняемого файла

Загружаю исполняемый файл в отладчик GDB(рис. 3.10).

```
arsihrceva@dk2n21 ~/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)
```

Рис. 3.10: Загрузка файла в отладчик

Проверяю работу программы, запустив её в оболочке GDB с помощью команды r(рис. 3.11).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 6074) exited normally]
(gdb)
```

Рис. 3.11: Запуск программы

Для более подробного анализа программы устанавливаю брейкпоинт на метку _start, с которой начинается выполнение программы. Запускаю её(рис. 3.12).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 12.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:12
12              mov eax,4
(gdb)
```

Рис. 3.12: Установка брейкпоинта и запуск программы

С помощью команды disassemble _start изучаю дисассимилированный код программы(рис. 3.13).

Рис. 3.13: Просмотр дисассимилированного кода

Переключаюсь на отображение команд с intel'овским синтаксисом(рис. 3.14).

```
(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. 3.14: Отображение команд с intel'овским синтаксисом

В дисассимилированном отображении используютя символы % и $ в командах, intel'овское же отображении они отсутствуют. Для удобства включаю режим псевдографики для более удобного анализа программы(рис. 3.15).

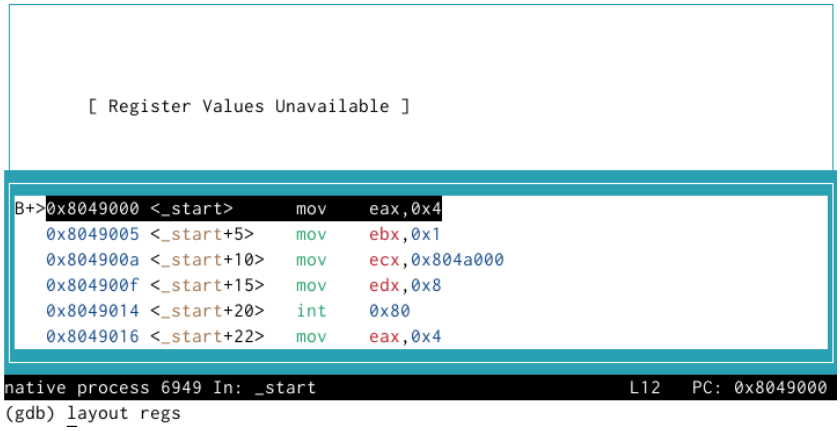

Рис. 3.15: Включение режима псевдографики

Проверяю наличие метки _start(рис. 3.16).

```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08049000 lab9-2.asm:12
        breakpoint already hit 1 time
```

Рис. 3.16: Информация об установленных метках

Вручную ищу последнюю команду кода и смотрю её адрес(рис. 3.17).

```
0x8049031 <_start+49>    mov    ebx,0x0
```

Рис. 3.17: Адрес кода mov ebx,0x0

Устанавливаю очку останова данной строчке(рис. 3.18).навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 25.
```

Рис. 3.18: Точка останова

Ещё раз просматриваю информацию о брейкпоинтах(рис. 3.19).

```
(gdb) i b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08049000 lab9-2.asm:12
        breakpoint already hit 1 time
2       breakpoint     keep y   0x08049031 lab9-2.asm:25
```

Рис. 3.19: Брейкпоинты

Выполняю команду stepi, изучаю значение регистров(рис. 3.20).

```
Register group: general
eax            0x4                4
ecx            0x0                0
edx            0x0                0
ebx            0x0                0
esp            0xffffc3e0         0xffffc3e0
ebp            0x0                0x0
esi            0x0                0
edi            0x0                0
eip            0x8049005          0x8049005 <_start+5>
eflags         0x202              [ IF ]
cs             0x23               35
ss             0x2b               43
```

Рис. 3.20: Значение регистров в начальный момент

Пять раз выполняю инструкции с помощью данной команды(рис. 3.21).

```
(gdb) stepi
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 3.21: Выполнение инструкций

Изучаю изменения значений регистров. Они поменялись у регистров eax, ecx, edx, ebx, eip(рис. 3.22).

Register group: general

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffc3e0	0xffffc3e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43

Рис. 3.22: Изменившиеся значения регистров

Просматриваю значение переменной msg1 по имени(рис. 3.23).

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

Рис. 3.23: Вывод значения msg1

Вручную ищу адрес msg2(рис. 3.24).

```
0x8049020 <_start+32>    mov    ecx,0x804a008
```

Рис. 3.24: Адрес msg2

Просматриваю значение переменной msg2 по адрессу(рис. 3.25).

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:        "world!\n\034"
```

Рис. 3.25: Значение msg2

Изменяю первый символ переменной msg1. Проверяю(рис. 3.26).

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

Рис. 3.26: Изменение значения msg1

Также изменяю первый символ переменной msg2. Проверяю(рис. 3.27).

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

Рис. 3.27: Изменение значения msg2

Вывожу значение регистра edx в шестнадцатиричном формате(рис. 3.28), в двоичном формате (рис. 3.29), в символьном виде(рис. 3.30).

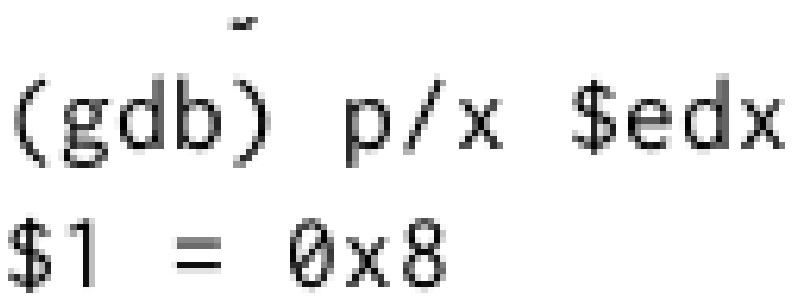

Рис. 3.28: Значение регистра edx в шестнадцатеричном формате

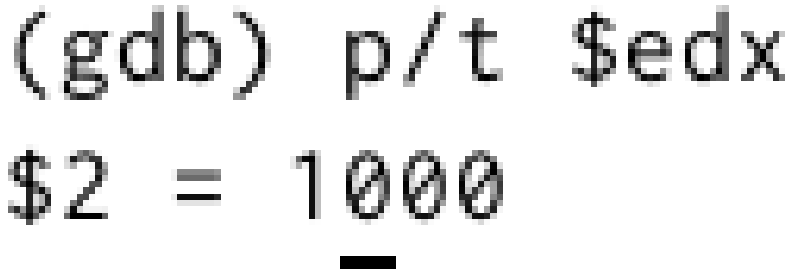

Рис. 3.29: Значение регистра edx в двоичном формате

Рис. 3.30: Значение регистра edx в символьном виде

Изменяю значение регистра ebx с помощью команды set(рис. 3.31).

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

Рис. 3.31: Изменение значения регистра

Выводятся разные значения, так как в первый раз мы вносим значение 2, а во второй - регистр приравниваем к 2. Завершаю выполнение программы и выхожу из GDB. Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, изменяю его название на lab9-3.asm(рис. 3.32).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
```

Рис. 3.32: Копирование нужного файла

Создаю исполняемый файл. При запуске указываю ключ –args и аргументы(рис. 3.33).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент 3'
```

Рис. 3.33: Запуск исполняемого файла

Устонавливаю точку останова перед первой иструкцией(рис. 3.34).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 7.
```

Рис. 3.34: Установка точки останова

Заускаю программу(рис. 3.35).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 а
ргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:7
7               pop ecx
```

Рис. 3.35: Запуск программы

проверяю адрес вершины стека(рис. 3.36).

```
(gdb) x/x $esp
0xffffc3a0:     0x00000005
```

Рис. 3.36: Адрес вершины стека

Убеждаюсь, что в нем хранится 5 аргументов Просматриваю остальные позиции стека(рис. 3.37), (рис. 3.38), (рис. 3.39), (рис. 3.40), (рис. 3.41), (рис. 3.42).

```
(gdb) x/s *(void**)($esp + 4)
0xffffc5f1:     "/afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-3"
```

Рис. 3.37: Позиция стека с адресом имени программы

```
(gdb) x/s *(void**)($esp + 8)
0xffffc637:     "аргумент1"
```

Рис. 3.38: Позиция стека с адресом первого аргумента

```
(gdb) x/s *(void**)($esp + 12)
0xffffc649:     "аргумент"
```

Рис. 3.39: Позиция стека с адресом второго аргумента

```
(gdb) x/s *(void**)($esp + 16)
0xffffc65a:     "2"
```

Рис. 3.40: Позиция стека с адресом третьего аргумента

```
(gdb) x/s *(void**)($esp + 20)
0xffffc65c:     "аргумент 3"
```

Рис. 3.41: Позиция стека с адресом четвёртого аргумента

```
(gdb) x/s *(void**)($esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
```

Рис. 3.42: Позиция стека с адресом пятого аргумента

Шаг изменения адреса равен 4, так как стех хранит до 4 байт. Также компьютер использует новый стек для нормального сохранения данных.

#Самостоятельная работа

3.1 Задание 1

Копирую файл с самостоятельной работой из лабораторной работы №8 и переименовываю его. Изменяю текст программы так, чтобы значение функции вычислялось как подпрограмма(рис. 3.43).

```
%include 'in_out.asm'

SECTION .data
msg DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
pop ecx

pop edx

sub ecx,1

mov esi,0

next:
cmp ecx,0
jz _end

pop eax
call atoi
call _calcul

add esi,eax

loop next

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

_calcul:

mov ebx,2
mul ebx

add eax,15

 ret
```

Рис. 3.43: Изменённый текст программы для вычисления суммы функций

Создаю и запускаю исполняемый файл. Проверяю его работу(рис. 3.44).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ./lab9-4 1 2 3
Результат: 57
```

Рис. 3.44: Результат работы программы

3.2 Задание 2

Создаю новый файл для самостоятельной работы(рис. 3.45).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ touch lab9-5.asm
```

Рис. 3.45: Создание файла

Ввожу текст программы для вычисления выражения(рис. 3.46).

```
%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 ecx,4
        mul ecx
        add ebx,5
        mov edi,ebx

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

call quit
```

Рис. 3.46: Текст программы

Создаю и запускаю исполняемый файл, чтобы проверить его работу(рис. 3.47).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10
```

Рис. 3.47: Результат работы программы

Программа выводит неверный ответ. Вместо 25 - 10 Для поиска ошибки воспользуюсь отладчиком GDB. Загружаю исполняемый файл в данный отладчик, устанавливаю точку останова и запускаю программу(рис. 3.48).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(No debugging symbols found in lab9-5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-5

Breakpoint 1, 0x080490e8 in _start ()
```

Рис. 3.48: Запуск программы в отладчике GDB

Включаю режим псевдографики, просматриваю значение регистров(рис. 3.49).

```
Register group: general
eax            0x0                0
ecx            0x0                0
edx            0x0                0
ebx            0x3                3
esp            0xffffc3e0         0xffffc3e0
ebp            0x0                0x0
esi            0x0                0
edi            0x0                0
eip            0x80490ed          0x80490ed <_start+5>
eflags         0x202              [ IF ]
cs             0x23               35
ss             0x2b               43
```

Рис. 3.49: Значение регистров

Пошагово выполняю инструкции с помощью команды steri. Нахожу ошибку в неправильном порядке операнд. Исправляю это. Создаю и запускаю исполняемый файл. Проверяю его работу(рис. 3.50).

```
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-5.lst lab9-5.asm
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
arsihrceva@dk2n21 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/arsihrceva/work/arch-pc/lab09/lab9-5
Результат: 25
[Inferior 1 (process 17745) exited normally]
```

Рис. 3.50: Проверка работы исправленного кода

4 Выводы

Приобретены навыки написания программ с использованием подпрограмм. Я ознакомилась с методами отладки при помощи GDB и его основными возможностями.