

# **Лабораторная работа №9**

**Архитектура компьютера**

Башиянц Александра Кареновна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Задание</b>	<b>4</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
3.1	Реализация подпрограмм в NASM . . . . .	5
3.2	Отладка программ с помощью GDB . . . . .	6
3.2.1	Добавление точек останова . . . . .	9
3.2.2	Работа с данными программы в GDB . . . . .	9
3.2.3	Обработка аргументов командной строки в GDB . . . . .	11
3.3	Задание для самостоятельной работы . . . . .	12
3.3.1	Задание 1 . . . . .	12
3.3.2	Задание 2 . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>17</b>

# 1 Цель работы

Цель работы — приобрести навыки написания программ с использованием подпрограмм, познакомиться с методами отладки при помощи GDB и его основными возможностями.

## 2 Задание

В этой лабораторной работе необходимо изучить работу циклов и обратку аргументов командной строки.

Необходимо научиться:

- Изучить использование подпрограмм;
- Изучить создание точек останова;
- Узнать работу с данными программы в GDB;
- Изучить как обрабатывать аргументы командной строки в GDB

Выполняя это задание, мы получим практический опыт с работой подпрограмм и методами отладки при помощи GDB

## 3 Выполнение лабораторной работы

### 3.1 Реализация подпрограмм в NASM

Создадим директорию для 9 лабораторной работы и создадим файл lab09-1.asm (рис. 3.1).

```
akbashiyanc@fedora1:~$ mkdir ~/work/arch-pc/lab09
akbashiyanc@fedora1:~$ cd ~/work/arch-pc/lab09
akbashiyanc@fedora1:~/work/arch-pc/lab09$ touch lab09-1.asm
akbashiyanc@fedora1:~/work/arch-pc/lab09$ ls
lab09-1.asm
```

Рис. 3.1: Создание директории

Скопируем файл in\_out.asm из lab06 с помощью mc (рис. 3.2).

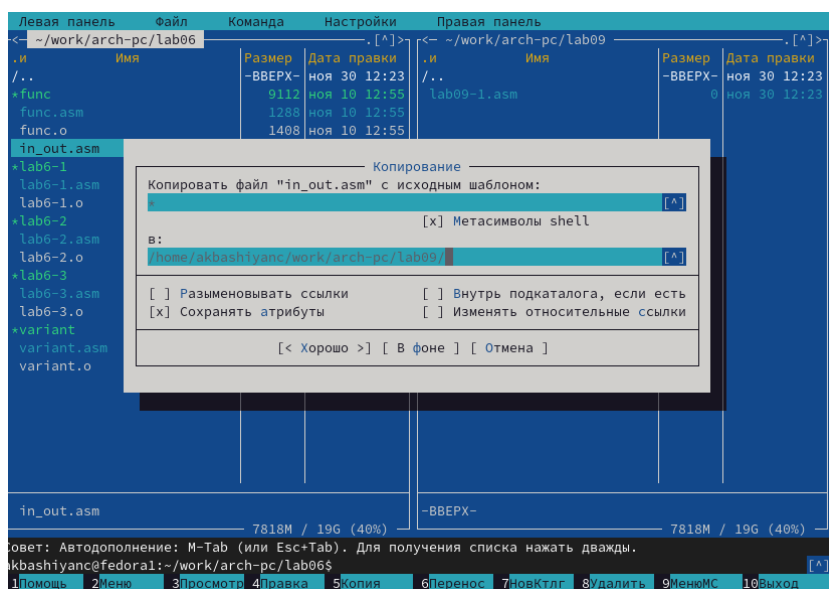


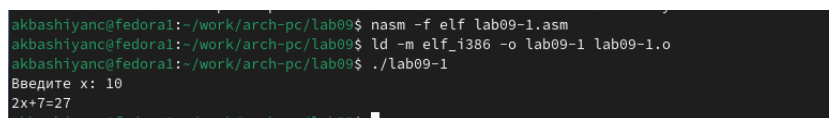
Рис. 3.2: Копирование in\_out.asm

Введем код в lab09-1.asm и создадим исполняемый файл и запустим его (рис. 3.3 и 3.4).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Введите x: ',0
5 result: DB '2x+7=',0
6 SECTION .bss
7 x: RESB 80
```

Рис. 3.3: Ввод кода

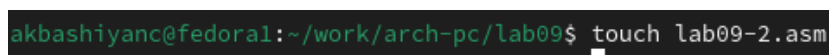


```
akbashiyc@fedora1:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
akbashiyc@fedora1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1.o lab09-1.o
akbashiyc@fedora1:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 10
2x+7=27
```

Рис. 3.4: Запуск файла

## 3.2 Отладка программ с помощью GDB

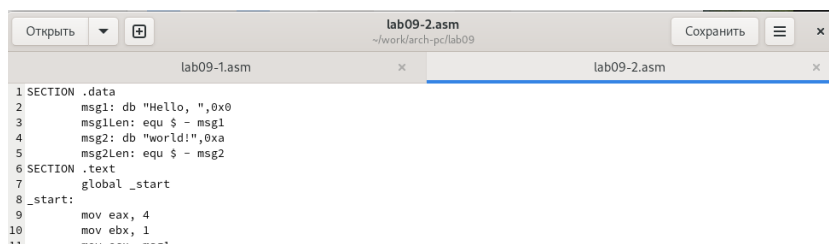
Создадим файл lab09-2.asm (рис. 3.5).



```
akbashiyc@fedora1:~/work/arch-pc/lab09$ touch lab09-2.asm
```

Рис. 3.5: Создание файла

Введем код в lab09-2.asm (рис. 3.6).



```
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
```

Рис. 3.6: Ввод кода

Создадим исполняемый файл (рис. 3.7). Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.

```

akbashiyanc@fedora1:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
akbashiyanc@fedora1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o

```

Рис. 3.7: Создание исполняемого файла

Загрузим исполняемый файл в отладчик gdb (рис. 3.8).

```

akbashiyanc@fedora1:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 15.1-1.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 lab09-2...

```

Рис. 3.8: Загрузка файла в gdb

Запустим файл (рис. 3.9).

```

(gdb) run
Starting program: /home/akbashiyanc/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4390) exited normally]
(gdb)

```

Рис. 3.9: Запуск файла в gdb

Для более подробного анализа программы установим брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустим её (рис. 3.10).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb)

```

Рис. 3.10: Установка брейкпоинта

Посмотрим дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 3.11).

```
(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)
```

Рис. 3.11: disassemble

Переключимся на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel (рис. 3.12).

```
(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)
```

Рис. 3.12: Переключение на синтаксис Intel

Включим режим псевдографики для более удобного анализа программы (рис. 3.13).

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd040 0xffffd040
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

0x080492c4 add     BYTE PTR [eax],al
0x080492c6 add     BYTE PTR [eax],al
0x080492c8 add     BYTE PTR [eax],al
0x080492ca add     BYTE PTR [eax],al
0x080492cc add     BYTE PTR [eax],al
0x080492ce add     BYTE PTR [eax],al
0x080492d0 add     BYTE PTR [eax],al
0x080492d2 add     BYTE PTR [eax],al
0x080492d4 add     BYTE PTR [eax],al

native process 4460 (asm) In: _start      L9    PC: 0x08049000
(gdb) layout regs
(gdb)
```

Рис. 3.13: Включение режим псевдографики



### 3.2.1 Добавление точек останова

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверим это с помощью команды `info breakpoints` (рис. 3.14).

```
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint      keep y   0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
```

Рис. 3.14: Включение режим псевдографики

Установим еще одну точку останова по адресу инструкции (рис. 3.15).

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num    Type             Disp Enb Address      What
1      breakpoint      keep y   0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
2      breakpoint      keep y   0x8049031 lab09-2.asm:20
(gdb)
```

Рис. 3.15: Установка точки останова

### 3.2.2 Работа с данными программы в GDB

Посмотрим содержимое регистров также можно с помощью команды `info registers` (рис. 3.16).

```
native process 4460 (asm) In: _start                               L9    PC: 0x804
eax      0x0              0
ecx      0x0              0
edx      0x0              0
ebx      0x0              0
esp      0xffffd040       0xffffd040
ebp      0x0              0x0
esi      0x0              0
edi      0x0              0
eip      0x8049000       0x8049000 <_start>
eflags   0x202           [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 3.16: Содержимое регистров

Посмотрим значение переменной `msg1` по имени (рис. 3.17).

```
0x0804a000 <msg1>: 0x080c0348
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)
```

Рис. 3.17: Содержимое регистров

Посмотрим значение переменной `msg2` по имени (рис. 3.18).

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

Рис. 3.18: Содержимое регистров

Изменим значение для регистра или ячейки памяти можно с помощью команды `set`, задав ей в качестве аргумента имя регистра или адрес (рис. 3.19).

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

Рис. 3.19: Изменение значения регистра msg1

Изменим 1 символ в msg2 (рис. 3.20).

```
(gdb) set {char}&msg2='M'
(gdb) x/1sb &msg2
No symbol "msg2" in current context.
(gdb) x/1sb &msg2
0x804a008 <msg2>: "World!\n\034"
(gdb)
```

Рис. 3.20: Изменение значения регистра msg2

С помощью команды `set` изменим значение регистра `ebx` (рис. 3.21).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 2
(gdb)
```

Рис. 3.21: Изменение значения регистра ebx

Завершим выполнение программы с помощью команды `continue` (сокращенно `c`) и выйдем из GDB с помощью команды `quit` (сокращенно `q`) (рис. 3.22).

```
(gdb) c
Continuing.
hello, World!
Breakpoint 2, _start () at lab09-2.asm:20
(gdb) q
```

Рис. 3.22: Выход из GDB

### 3.2.3 Обработка аргументов командной строки в GDB

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, назовем его lab9-3.asm (рис. 3.23).

```
akbashiyanc@fedoral: ~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
akbashiyanc@fedoral: ~/work/arch-pc/lab09$
```

Рис. 3.23: Копирование lab8-2.asm

Создадим исполняемый файл (рис. 3.24).

```
akbashiyanc@fedoral:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
akbashiyanc@fedoral:~/work/arch-pc/lab09$
```

Рис. 3.24: Создание исполняемого файла

Загрузим исполняемый файл в отладчик gdb (рис. 3.25).

```
akbashiyanc@fedoral:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 15.1-1.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. 3.25: Загрузка файла в gdb

Для начала установим точку останова перед первой инструкцией в программе и запустим ее (рис. 3.26).

```
(gdb) b _start
Breakpoint 1 at 0x80490e0: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/akbashiyanc/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. 3.26: Загрузка файла в gdb

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы) (рис. 3.27).

```
(gdb) x/x $esp
0xffffd000: 0x00000005
(gdb)
```

Рис. 3.27: Вершина стека

Посмотрим остальные позиции стека (рис. 3.28).

```
(gdb) x/s *(void**)(esp + 4)
0xffffd100: "/home/akbashiyan/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd100: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd1ff: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd210: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd210: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 3.28: Остальные позиции стека

## 3.3 Задание для самостоятельной работы

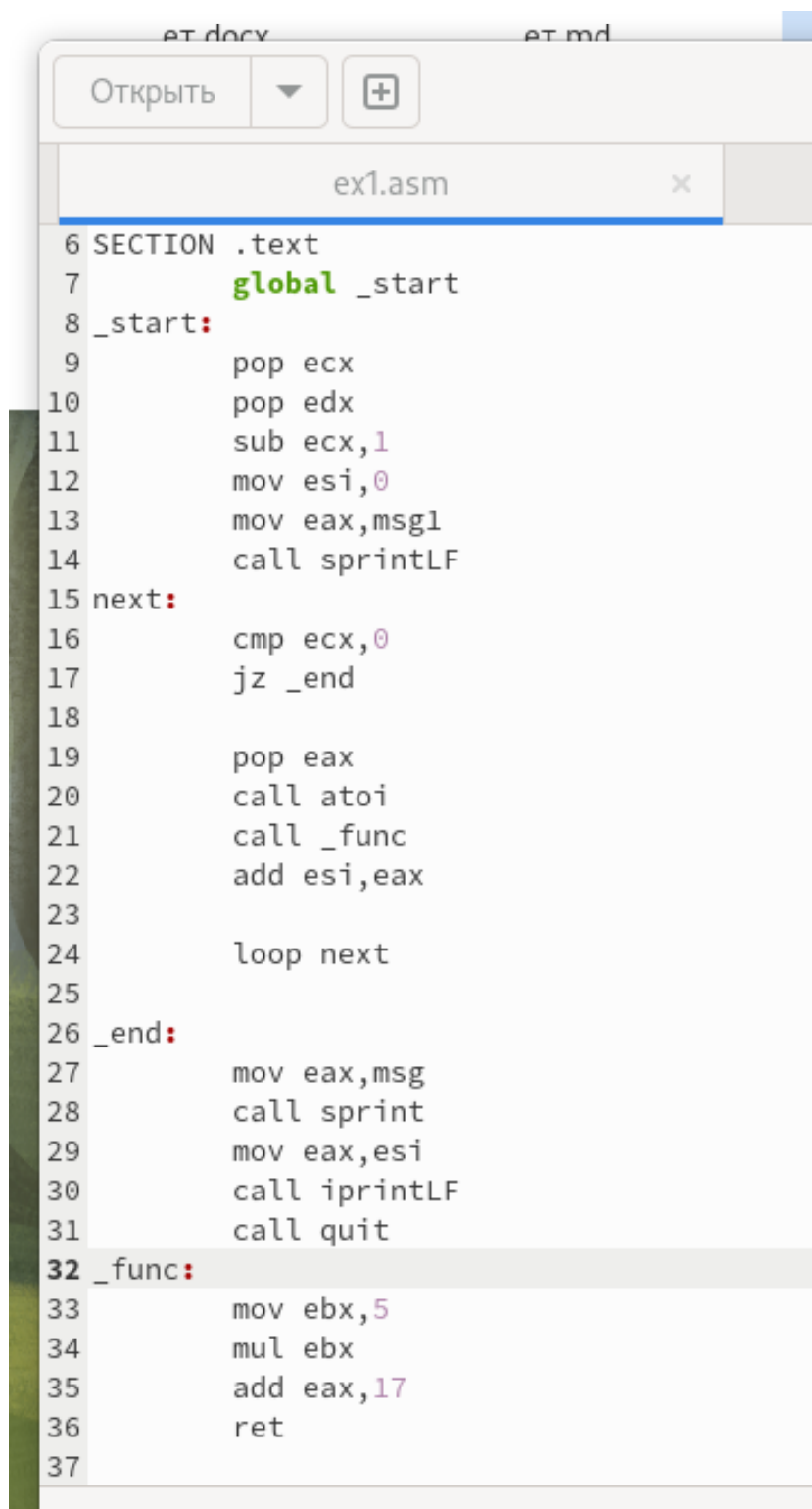
### 3.3.1 Задание 1

Скопируем файл ex1.asm из Лабораторной работы №8 (рис. 3.29).

```
akbashiyan@fedora1:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/ex1.asm ~/work/arch-pc/lab09/ex1.asm
```

Рис. 3.29: Создание файла

Изменим код для нахождения суммы функции  $f(x)=5*x+17$  с помощью подпрограммы (рис. 3.30).



```
6 SECTION .text
7     global _start
8 _start:
9     pop ecx
10    pop edx
11    sub ecx,1
12    mov esi,0
13    mov eax,msg1
14    call sprintfLF
15 next:
16    cmp ecx,0
17    jz _end
18
19    pop eax
20    call atoi
21    call _func
22    add esi,eax
23
24    loop next
25
26 _end:
27    mov eax,msg
28    call sprintf
29    mov eax,esi
30    call iprintLF
31    call quit
32 _func:
33    mov ebx,5
34    mul ebx
35    add eax,17
36    ret
37
```

Рис. 3.30: Изменение кода

Создадим исполняемый файл и запустим его (рис. [-fig. 3.31).

```

akbashiyanc@fedoral:~/work/arch-pc/lab09$ nasm -f elf ex1.asm
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ld -m elf_i386 -o ex1 ex1.o
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ./ex1 1 2 3
f(x)=5x+17
Результат: 81

```

Рис. 3.31: Запуск файла

### 3.3.2 Задание 2

Создадим файл ex2.asm (рис. 3.32).

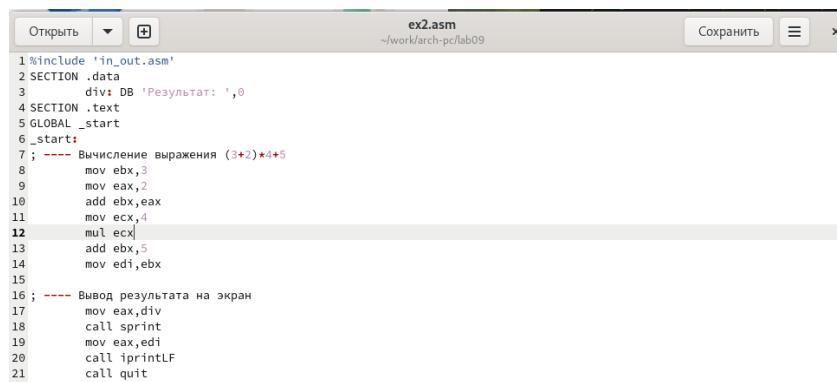
```

akbashiyanc@fedoral:~/work/arch-pc/lab09$ touch ex2.asm

```

Рис. 3.32: Создание файла

Введем код в ex2.asm (рис. 3.33).



```

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
16 ; ---- Вывод результата на экран
17    mov eax,div
18    call sprint
19    mov eax,edi
20    call iprintLF
21    call quit

```

Рис. 3.33: Ввод кода

Создадим исполняемый файл и запустим файл, как обычно (рис. 3.34).

```

akbashiyanc@fedoral:~/work/arch-pc/lab09$ nasm -f elf -g -l ex2.lst ex2.asm
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ld -m elf_i386 -o ex2 ex2.o
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ./ex2
Результат: 10

```

Рис. 3.34: Создание исполняемого файла

Заметим, что ответ неверный. Выводится 10, хотя должно быть 25.

Загрузим исполняемый файл в отладчик gdb (рис. 3.35).

```

akbashyanc@fedoral:~/work/arch-pc/lab09$ gdb ex2
GNU gdb (Fedora Linux) 15.1-1.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 ex2...
(gdb)

```

Рис. 3.35: Загрузка файла в gdb

Для начала установим точку останова перед первой инструкцией в программе и запустим ее (рис. 3.36).

```

(gdb) r
Starting program: /home/akbashyanc/work/arch-pc/lab09/ex2

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.
Результат: 10
[Inferior 1 (process 7613) exited normally]
(gdb)

```

Рис. 3.36: Запуск файла в gdb

Откроем регистры (рис. 3.37).

```

(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>:    mov     ebx,0x3
   0x080490ed <+5>:    mov     eax,0x2
   0x080490f2 <+10>:   add     ebx,eax
   0x080490f4 <+12>:   mov     ecx,0x4
   0x080490f9 <+17>:   mul     ecx
   0x080490fb <+19>:   add     ebx,0x5
   0x080490fe <+22>:   mov     edi,ebx
   0x08049100 <+24>:   mov     eax,0x804a000
   0x08049105 <+29>:   call   0x0804900f <sprint>
   0x0804910a <+34>:   mov     eax,edi
   0x0804910c <+36>:   call   0x08049086 <iprintLF>
   0x08049111 <+41>:   call   0x080490db <quit>
End of assembler dump.
(gdb)

```

Рис. 3.37: Регистры

Заметим, что некоторые регистры стоят не на своих местах. Исправим это (рис. 3.38).

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x80490db <quit>      mov     ebx,0x0
0x80490e0 <quit+5>    mov     eax,0x1
0x80490e5 <quit+10>   int     0x80
0x80490e7 <quit+12>   ret
B> 0x80490e8 <_start>  mov     ebx,0x3
0x80490ed <_start+5>  mov     eax,0x2
0x80490f2 <_start+10> add     ebx,eax
0x80490f4 <_start+12> mov     ecx,0x4
0x80490f9 <_start+17> mul     ecx
0x80490fb <_start+19> add     ebx,0x5
0x80490fe <_start+22> mov     edi,ebx
0x8049100 <_start+24> mov     eax,0x804a000
0x8049105 <_start+29> call    0x804900f <sprint>
0x804910a <_start+34> mov     eax,edi

```

Рис. 3.38: Исправление регистров

Выйдем из gdb, сформируем заново исполняемый файл из запустим файл заново (рис. 3.38).

```

akbashiyanc@fedoral:~/work/arch-pc/lab09$ nasm -f elf -g -l ex2.lst ex2.asm
akbashiyanc@fedoral:~/work/arch-pc/lab09$ ld -m elf_i386 -o ex2 ex2.o
akbashiyanc@fedoral:~/work/arch-pc/lab09$ gdb ex2
GNU gdb (Fedora Linux) 15.1-1.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 ex2...
(gdb) r
Starting program: /home/akbashiyanc/work/arch-pc/lab09/ex2

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.
Результат: 25
[Inferior 1 (process 8022) exited normally]
(gdb) █

```

Рис. 3.39: Запуск исправленного файла

Теперь все считается корректно.



## **4 Выводы**

В ходе выполнения работы были получены навыки работы с подпрограммами и методами отладки при помощи GDB.