

Отчёт по лабораторной работе №9

Иванова Анастасия Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	19
4	Вывод	26

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Ввод текста программы	7
2.3	Создание исполняемого файла и запуск	8
2.4	Изменение текста программы	9
2.5	Проверка работы	10
2.6	Создание файла	10
2.7	Ввод программы	11
2.8	Получение исполняемого файла и загрузка файла в отладчик gdb . .	12
2.9	Запуск программы	12
2.10	Установка метки	12
2.11	Просмотр кода	13
2.12	Переключение на отображение команд	13
2.13	Включение режима псевдографики	14
2.14	Проверка установленной точки	14
2.15	Установка еще одной точки	15
2.16	Просмотр информации о всех точках	15
2.17	Просмотр значения	15
2.18	Просмотр значения	16
2.19	Изменение значения	16
2.20	Изменение значения	16
2.21	Копирование файла	16
2.22	Установка точки	17
2.23	Запуск программы	17
2.24	Позиции стека	18
3.1	Ввод программы	20
3.2	Проверка работы	20
3.3	Создание файла	21
3.4	Ввод программы	22
3.5	Запуск программы	22
3.6	Просмотр процесс исполнения программы	23
3.7	Изменение кода программы	24
3.8	Проверка работы	24

Список таблиц

1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

1. Создадим каталог для выполнения лабораторной работы № 9, перейдем в него и создадим файл lab09-1.asm рис. 2.1:



```
asivanova@anastasia-750XGK:~$ mkdir ~/work/arch-pc/lab09
asivanova@anastasia-750XGK:~$ cd ~/work/arch-pc/lab09
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ touch lab09-1.asm
```

Рисунок 2.1: Создание каталога и файла

2. В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы _calcul.

Введем в файл lab09-1.asm текст программы рис. 2.2:

```

GNU nano 6.2
#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

mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

mov ebx, 2
mul ebx
add eax, 7

```

Рисунок 2.2: Ввод текста программы

Создадим исполняемый файл и проверим его работу рис. 2.3:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ mc
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
2x+7=13
```

Рисунок 2.3: Создание исполняемого файла и запуск

Изменим текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. Т.е. x передается в подпрограмму `_calcul` из нее в подпрограмму `_subcalcul`, где вычисляется выражение $g(x)$, результат возвращается в `_calcul` и вычисляется выражение $f(g(x))$ рис. 2.4:


```

GNU nano 6.2
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB 'f(g(x)) = 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 eax, [esp+4]
mov ebx, 3
mul ebx
sub eax, 1

ret

```

Рисунок 2.4: Изменение текста программы

Запустим измененный файл рис. 2.5:

```
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 4
f(g(x)) = 2*(3x-1)+7 = 29
```

Рисунок 2.5: Проверка работы

3. Создадим файл lab09-2.asm с текстом программы печати сообщения Hello world! рис. 2.6:

```
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ touch lab09-2.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ ls
in out.asm lab09-1 lab09-1.asm lab09-1.o lab09-2.asm
```

Рисунок 2.6: Создание файла

рис. 2.7:

```
GNU nano 6.2 /home/anastasia,  
SECTION .data  
msg1: db "Hello, ",0x0  
msg1Len: equ $ - msg1  
  
msg2: db "world!",0xa  
msg2Len: equ $ - msg2  
SECTION .text  
global _start  
  
_start:  
    mov eax, 4  
    mov ebx, 1  
    mov ecx, msg1  
    mov edx, msg1Len  
    int 0x80  
  
    mov eax, 4  
    mov ebx, 1  
    mov ecx, msg2  
    mov edx, msg2Len  
    int 0x80  
  
    mov eax, 1  
    mov ebx, 0  
    int 0x80
```

Рисунок 2.7: Ввод программы

Получим исполняемый файл и трансляцию программ, а затем загрузим исполняемый файл в отладчик gdb рис. 2.8:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1-22.04.2) 12.1
Copyright (C) 2022 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:
<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...

```

Рисунок 2.8: Получение исполняемого файла и загрузка файла в отладчик gdb

Проверим работу программы, запустим ее в оболочке GDB с помощью команды `run` рис. 2.9:

```

(gdb) run
Starting program: /home/anastasia/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 22249) exited normally]
(gdb)

```

Рисунок 2.9: Запуск программы

Для более подробного анализа программы установим брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустим её рис. 2.10:

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /home/anastasia/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4

```

Рисунок 2.10: Установка метки

Посмотрим дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` рис. 2.11:

```
(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.
```

Рисунок 2.11: Просмотр кода

Переключимся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` рис. 2.12:

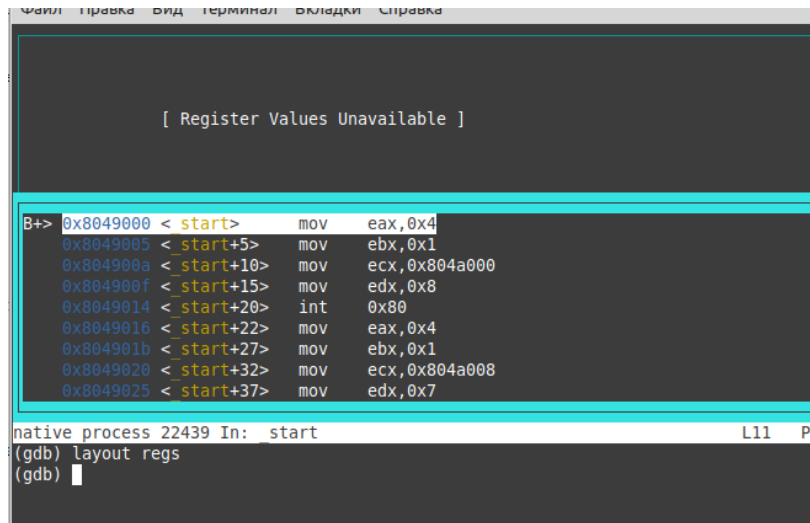
```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
      0x08049005 <+5>:      mov     ebx,0x1
      0x0804900a <+10>:     mov     ecx,0x804a000
      0x0804900f <+15>:     mov     edx,0x8
      0x08049014 <+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.
```

Рисунок 2.12: Переключение на отображение команд

Различия отображения синтаксиса: AT&T синтаксис более последователен (всегда источник→цель, явные суффиксы размера), а Intel синтаксис более читаем для человека (естественный порядок «куда←откуда»).

Включим режим псевдографики для более удобного анализа программы

рис. 2.13:



The screenshot shows the GDB interface with the assembly window displaying code from address 0x8049000 to 0x8049025. The code includes instructions like `mov eax,0x4`, `mov ebx,0x1`, `mov ecx,0x804a000`, `mov edx,0x8`, `int 0x80`, and `mov` instructions for `eax`, `ebx`, `ecx`, and `edx`. The console window shows the command `(gdb) layout regs` and the status `native process 22439 In: start`.

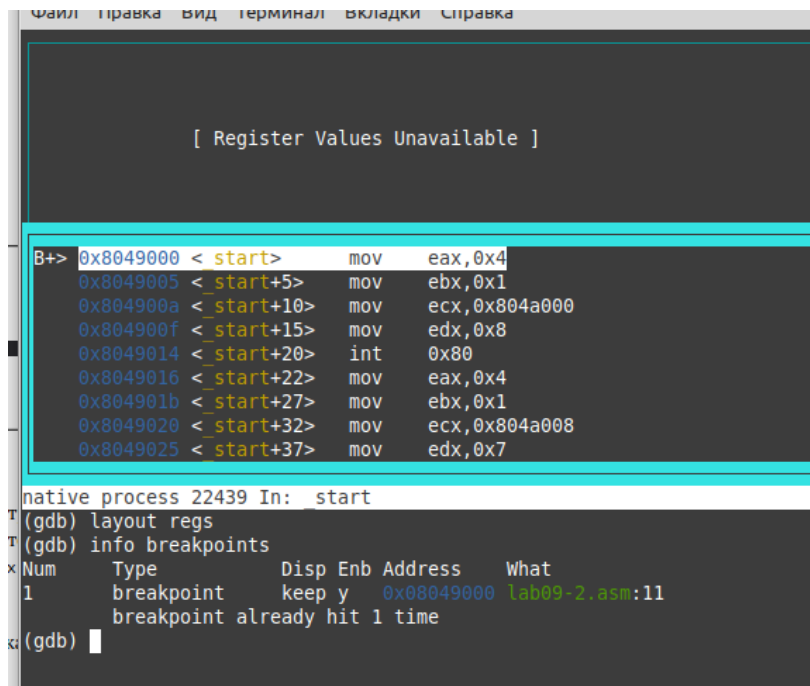
```
[ Register Values Unavailable ]

B+> 0x8049000 < _start>    mov     eax,0x4
0x8049005 < _start+5>    mov     ebx,0x1
0x804900a < _start+10>   mov     ecx,0x804a000
0x804900f < _start+15>   mov     edx,0x8
0x8049014 < _start+20>   int     0x80
0x8049016 < _start+22>   mov     eax,0x4
0x804901b < _start+27>   mov     ebx,0x1
0x8049020 < _start+32>   mov     ecx,0x804a008
0x8049025 < _start+37>   mov     edx,0x7

native process 22439 In: start L11 PC
(gdb) layout regs
(gdb)
```

Рисунок 2.13: Включение режима псевдографики

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверим это с помощью команды `info breakpoints` рис. 2.14:



The screenshot shows the GDB interface with the assembly window displaying the same code as in Figure 2.13. The console window shows the command `(gdb) info breakpoints` and the output showing a single breakpoint at address 0x08049000.

```
[ Register Values Unavailable ]

B+> 0x8049000 < _start>    mov     eax,0x4
0x8049005 < _start+5>    mov     ebx,0x1
0x804900a < _start+10>   mov     ecx,0x804a000
0x804900f < _start+15>   mov     edx,0x8
0x8049014 < _start+20>   int     0x80
0x8049016 < _start+22>   mov     eax,0x4
0x804901b < _start+27>   mov     ebx,0x1
0x8049020 < _start+32>   mov     ecx,0x804a008
0x8049025 < _start+37>   mov     edx,0x7

native process 22439 In: start
(gdb) layout regs
(gdb) info breakpoints
Num      Type             Disp Enb Address            What
1        breakpoint       keep y 0x08049000 lab09-2.asm:11
breakpoint already hit 1 time
(gdb)
```

Рисунок 2.14: Проверка установленной точки

Установим еще одну точку остановки по адресу инструкции рис. 2.15:

```
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al

native process 23813 In: _start L11 P
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 24.
(gdb) i b
Num      Type             Disp Enb Address      What
1        breakpoint       keep y 0x08049000 lab09-2.asm:11
          breakpoint already hit 1 time
2        breakpoint       keep y 0x08049031 lab09-2.asm:24
(gdb)
```

Рисунок 2.15: Установка еще одной точки

Посмотрим информацию о всех установленных точках останова рис. 2.16:

```
0x8049046          add    BYTE PTR [eax],al
0x8049048          add    BYTE PTR [eax],al
0x804904a          add    BYTE PTR [eax],al
0x804904c          add    BYTE PTR [eax],al
0x804904e          add    BYTE PTR [eax],al

native process 23813 In: _start
eax          0x0
ecx          0x0
edx          0x0
ebx          0x0
esp          0xffffd220 0xffffd220
ebp          0x0
esi          0x0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рисунок 2.16: Просмотр информации о всех точках

Посмотрим значение переменной msg1 по имени рис. 2.17:

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

Рисунок 2.17: Просмотр значения

Посмотрим значение переменной msg2 по адресу рис. 2.18:

```
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
```

Рисунок 2.18: Просмотр значения

С помощью команды set изменим значение регистра ebx рис. 2.19:

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

Рисунок 2.19: Изменение значения

рис. 2.20:

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

Рисунок 2.20: Изменение значения

Разницу вывода команд p/s \$ebx: p/s — показывает число со знаком (как положительное или отрицательное), p/d или p — показывает как абсолютное значение (только положительное)

4. Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm, создадим исполняемый файл и загрузим исполняемый файл в отладчик, указав аргументы рис. 2.21:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2
.asm ~/work/arch-pc/lab09/lab09-3.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ls
in out.asm lab09-1.asm lab09-2 lab09-2.lst lab09-3.asm
lab09-1 lab09-1.o lab09-2.asm lab09-2.o
```

Рисунок 2.21: Копирование файла

Для начала установим точку останова перед первой инструкцией в программе и запустим ее рис. 2.22:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'ар
мент 3'
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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:
<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...
```

Рисунок 2.22: Установка точки

рис. 2.23:

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

Рисунок 2.23: Запуск программы

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки. Как видно, число аргументов равно 5 – это имя программы lab09-3 и непосредственно аргументы: аргумент1, аргумент, 2 и „аргумент 3“.

Посмотрим остальные позиции стека рис. 2.24:

```

(gdb) x/x $esp
0xffffd1f0: 0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xffffd3be: "/home/anastasia/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xffffd3e9: "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xffffd3fb: "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xffffd40c: "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffd40e: "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рисунок 2.24: Позиции стека

Объясним, почему шаг изменения адреса равен 4: шаг равен 4 байтам потому что в 32-битной архитектуре каждый указатель на строку аргумента занимает 4 байта в памяти.

3 Задание для самостоятельной работы

1. Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $\varphi(x)$ как подпрограмму.

Введем программу рис. 3.1:

```

GNU nano 6.2
#include 'in_out.asm'
SECTION .data
func_msg db "Функция: f(x)=30x-11",0
result_msg db "Результат: ",0
SECTION .text
global _start
f:
mov ebx,30
mul ebx
sub eax,11
ret
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
next:
cmp ecx,0
jz _end
pop eax
call atoi
call f
add esi,eax
dec ecx
jmp next
_end:
mov eax,func_msg
call printf
mov eax,result_msg
call printf
mov eax,esi
call printf
call quit

```

Рисунок 3.1: Ввод программы

Создадим исполняемый файл и проверим его работу рис. 3.2:

```

asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf zadanie.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o zadanie.o zadanie.o
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ./zadanie
Функция: f(x)=30x-11
Результат: 0

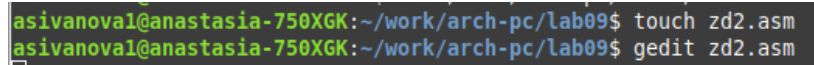
```

Рисунок 3.2: Проверка работы

2. Нам приведена в пример программа вычисления выражения $(3 + 2) * 4 + 5$.

При запуске данная программа дает неверный результат. Проверим это, а затем помощью отладчика GDB, анализируя изменения значений регистров, определим ошибку и исправим ее.

Создадим файл и запишем туда текст программы рис. 3.3:

A terminal window with a dark background and light green text. The prompt is 'asivanova1@anastasia-750XGK:~/work/arch-pc/lab09\$'. The first command is 'touch zd2.asm' and the second is 'gedit zd2.asm'.

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ touch zd2.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ gedit zd2.asm
```

Рисунок 3.3: Создание файла

рис. 3.4:

```

GNU nano 6.2 /home/ana
%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

```

Рисунок 3.4: Ввод программы

Создадим исполняемый файл и запустим программу рис. 3.5:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ nasm -f elf zd2.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o zd2 zd2.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ ./zd2
Результат: 10

```

Рисунок 3.5: Запуск программы

С помощью GDB посмотрим что происходит в программе, а затем исправим код в файле рис. 3.6:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab09$ gdb ./zd2
GNU gdb (Ubuntu 12.1-0ubuntu1-22.04.2) 12.1
Copyright (C) 2022 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:
<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 ./zd2...
(gdb) (gdb) break _start
Undefined command: "". Try "help".
(gdb) break _start
Breakpoint 1 at 0x80490e8: file zd2.asm, line 8.
(gdb) run
Starting program: /home/anastasia/work/arch-pc/lab09/zd2

Breakpoint 1, _start () at zd2.asm:8
8      mov ebx,3
(gdb) stepi
9      mov eax,2
(gdb) info registers
eax             0x0
ecx             0x0
edx             0x0
ebx             0x3
esp             0xffffd230    0xffffd230
ebp             0x0
esi             0x0
edi             0x0
eip             0x80490ed    0x80490ed <_start+5>
eflags         0x202        [ IF ]
cs              0x23        35
ss              0x2b        43
ds              0x2b        43
es              0x2b        43
fs              0x0
gs              0x0
(gdb) print $eax
$1 = 0
(gdb) print $ebx
$2 = 3
(gdb) print $ecx
$3 = 0
(gdb) continue
Continuing.
Результат: 10
[Inferior 1 (process 15454) exited normally]
(gdb)

```

Рисунок 3.6: Просмотр процесс исполнения программы

рис. 3.7:

```

GNU nano 6.2 /hor
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

mov ebx,3
mov eax,2
add 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

```

Рисунок 3.7: Изменение кода программы

Создадим исполняемый файл и запустим программу рис. 3.8:

```

asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ nasm -f elf zd2.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ ld -m elf_i386 -o zd2 zd2.o
asivanova@anastasia-750XGK:~/work/arch-pc/lab09$ ./zd2
Результат: 25

```

Рисунок 3.8: Проверка работы

Теперь программы выведит верное значение.

4 Вывод

Мы приобрели навыки написания программ с использованием подпрограмм, а также ознакомились с методами отладки при помощи GDB и его основными возможностями.