

# **Отчет по лабораторной работе №9**

**Дисциплина: Архитектура компьютера**

Морошан Матвей Корнелиович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация подпрограмм в NASM . . . . .	6
3.2	Отладка программ с помощью GDB . . . . .	8
3.3	Выполнение заданий для самостоятельной работы . . . . .	20
<b>4</b>	<b>Выводы</b>	<b>25</b>

# Список иллюстраций

3.1	Создание каталога, перехожу в него, создаю файл . . . . .	6
3.2	Копирование in_out.asm . . . . .	6
3.3	Текст программы lab09-1.asm . . . . .	7
3.4	Создание исполняемого файла и запуск . . . . .	7
3.5	Измененный текст программы lab09-1.asm . . . . .	8
3.6	Результат работы . . . . .	8
3.7	Создание lab09-2.asm . . . . .	9
3.8	Текст программы lab09-2.asm . . . . .	9
3.9	Трансляция, загрузка в gdb и проверка работы программы . . . . .	9
3.10	Брейкпоинт на метку _start . . . . .	10
3.11	Дисассимилированный код программы . . . . .	10
3.12	Переключение на отображение команд . . . . .	11
3.13	Режим псевдографики . . . . .	12
3.14	Команда info break . . . . .	13
3.15	Установка точки останова и просмотр информации . . . . .	13
3.16	Команда si . . . . .	14
3.17	info registers . . . . .	15
3.18	Значение переменной msg1 . . . . .	16
3.19	Значение msg2 и изменение первого символа msg1 . . . . .	17
3.20	Команда print . . . . .	18
3.21	Продолжение и выход . . . . .	19
3.22	Копирование в файл lab09-3.asm . . . . .	19
3.23	Исполняемый файл, загрузка в отладчик, установка точки останова	20
3.24	Позиции стека . . . . .	20
3.25	Создание файла lab09-4 . . . . .	20
3.26	Изменение программы . . . . .	21
3.27	Создание исполняемого файла и запуск . . . . .	21
3.28	Создание файла lab09-5 . . . . .	21
3.29	Текст программы . . . . .	22
3.30	Проверка программы . . . . .	22
3.31	Запуск программы в отладчике . . . . .	23
3.32	Регистры . . . . .	24
3.33	Вывод правильного ответа . . . . .	24

# 1 Цель работы

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

## **2 Задание**

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Выполнение заданий для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9, перехожу в него и создаю файл (рис. 3.1).

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

Рис. 3.1: Создание каталога, перехожу в него, создаю файл

Копирую файл in\_out.asm из загрузок в каталог для выполнения лабораторной работы №9 (рис. 3.2)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ cp ~/Загрузки/in_out.asm in_out.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1.asm
```

Рис. 3.2: Копирование in\_out.asm

Ввожу текст программы из листинга 9.1 в файл lab09-1.asm (рис. 3.3)

```

> mc [mkmoroshan@dk2n21.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab09
lab09-1.asm [-M--] 27 L:[ 1+34 35/ 35] *(707 / 707b) <EOF> [*][X]
#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:
;-----
; Основная программа
;-----
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. 3.3: Текст программы lab09-1.asm

Создаю исполняемый файл и запускаю (рис. 3.4)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 6
2x+7=19

```

Рис. 3.4: Создание исполняемого файла и запуск

Изменяю текст программы для функции  $f(g(x))$  (рис. 3.5)

```

> mc [mkmoroshan@dk8n60.dk.sci.pfu.edu.ru:~/work/arch-pc/lab09
lab09-1.asm      [-M--]  8 L: [ 1+40  41/ 44] *(446 / 464b) 0010 0x00A [*][X]
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: 2',0
prim1: DB 'f(x)=2x+7',0
prim2: DB 'g(x)=3x-1',0
result: DB 'f(g(x))= 17',0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, prim1
call sprintLF

mov eax, prim2
call sprintLF

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:

```

Рис. 3.5: Измененный текст программы lab09-1.asm

Создаю исполняемый файл и запускаю его (рис. 3.6)

```

mkmoroshan@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
mkmoroshan@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
mkmoroshan@dk8n60 ~/work/arch-pc/lab09 $ ./lab09-1
f(x)=2x+7
g(x)=3x-1
Введите x: 2
f(g(x))= 17

```

Рис. 3.6: Результат работы

## 3.2 Отладка программ с помощью GDB

Создаю файл lab09-2.asm с текстом программы из листинга 9.2 (рис. 3.7)



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

Рис. 3.7: Создание lab09-2.asm

Текст программы lab09-2.asm (рис. 3.8)

```
> mc [mkmoroshan@dk2n21.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab09
lab09-2.asm [-M--] 8 L:[ 1+20 21/ 21] *(293 / 293b) <EOF> [*](X)
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
```

Рис. 3.8: Текст программы lab09-2.asm

Получаю исполняемый файл и провожу трансляцию с ключом -g, загружаю исполняемый файл в отладчик gdb. Также проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 3.9)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ gdb lab09-2
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 lab09-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/k/mkmoroshan/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 8391) exited normally]
(gdb)
```

Рис. 3.9: Трансляция, загрузка в gdb и проверка работы программы

Устанавливаю брейкпоинт на метку `_start` и запускаю (рис. 3.10)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/k/mkmooroshan/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) █
```

Рис. 3.10: Брейкпоинт на метку `_start`

Смотрю дисассимилированный код программы с помощью команды `disassemble` 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.
```

Рис. 3.11: Дисассимилированный код программы

Переключаюсь 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.
```

Рис. 3.12: Переключение на отображение команд

Включаю режим псевдографики для более удобного анализа команды (рис. 3.13)

```
> gdb lab09-2

[ 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
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80

native process 8552 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) 
```

Рис. 3.13: Режим псевдографики

С помощью `info breakpoints` проверяю установку точек останова по имени метки на предыдущих шагах (рис. 3.14)

```

> gdb lab09-2
[ 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
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80

native process 8552 In: _start L9 PC: 0x8049000
(gdb) layout regs
(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: Команда info break

Устанавливаю точку останова и смотрю информацию о всех установленных точках останова (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   0x08049031 lab09-2.asm:20
(gdb)

```

Рис. 3.15: Установка точки останова и просмотр информации

Выполняю команду si (рис. 3.16)

```
> gdb lab09-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
0x804902a <_start+42>      int     0x80
0x804902c <_start+44>      mov     eax,0x1
b+ 0x8049031 <_start+49>      mov     ebx,0x0
0x8049036 <_start+54>      int     0x80

native process 8552 In: _start L10 PC: 0x8049005
(gdb) layout regs
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y 0x08049000 lab09-2.asm:9
      breakpoint already hit 1 time
(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 0x08049031 lab09-2.asm:20
(gdb) si
(gdb) |
```

Рис. 3.16: Команда si

Результат команды info registers (рис. 3.17)

```
> gdb lab09-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
b+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80

native process 8552 In: _start L10 PC: 0x8049005
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
```

Рис. 3.17: info registers

Смотрю значение переменной msg1 (рис. 3.18)

```
> gdb lab09-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x804904e add BYTE PTR [eax],al
0x8049050 add BYTE PTR [eax],al
0x8049052 add BYTE PTR [eax],al
0x8049054 add BYTE PTR [eax],al
0x8049056 add BYTE PTR [eax],al
0x8049058 add BYTE PTR [eax],al
0x804905a add BYTE PTR [eax],al
0x804905c add BYTE PTR [eax],al
0x804905e add BYTE PTR [eax],al
0x8049060 add BYTE PTR [eax],al
0x8049062 add BYTE PTR [eax],al
0x8049064 add BYTE PTR [eax],al
0x8049066 add BYTE PTR [eax],al

native process 8552 In: _start L10 PC: 0x8049005
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
```

Рис. 3.18: Значение переменной msg1

Смотрю значение переменной msg2 и изменяю первый символ переменной msg1 (рис. 3.19)



```

> gdb lab09-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x8049072 add BYTE PTR [eax],al
0x8049074 add BYTE PTR [eax],al
0x8049076 add BYTE PTR [eax],al
0x8049078 add BYTE PTR [eax],al
0x804907a add BYTE PTR [eax],al
0x804907c add BYTE PTR [eax],al
0x804907e add BYTE PTR [eax],al
0x8049080 add BYTE PTR [eax],al
0x8049082 add BYTE PTR [eax],al
0x8049084 add BYTE PTR [eax],al
0x8049086 add BYTE PTR [eax],al
0x8049088 add BYTE PTR [eax],al
0x804908a add BYTE PTR [eax],al

native process 8552 In: _start L10 PC: 0x8049005
ss      0x2b     43
ds      0x2b     43
es      0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "

```

Рис. 3.19: Значение msg2 и изменение первого символа msg1

Использую команду print (рис. 3.20)

```

> gdb lab09-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x8049072 add BYTE PTR [eax],al
0x8049074 add BYTE PTR [eax],al
0x8049076 add BYTE PTR [eax],al
0x8049078 add BYTE PTR [eax],al
0x804907a add BYTE PTR [eax],al
0x804907c add BYTE PTR [eax],al
0x804907e add BYTE PTR [eax],al
0x8049080 add BYTE PTR [eax],al
0x8049082 add BYTE PTR [eax],al
0x8049084 add BYTE PTR [eax],al
0x8049086 add BYTE PTR [eax],al
0x8049088 add BYTE PTR [eax],al
0x804908a add BYTE PTR [eax],al

native process 8552 In: _start L10 PC: 0x8049005
0x804a008 <msg2>: "world!\n\034"
(gdb) set (char)&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)

```

Рис. 3.20: Команда print

Продолжаю и выхожу (рис. 3.21)

```

> gdb lab09-2
eax      0x4      4
ecx      0x1      1
ecx      0x804a008 134520840
edx      0x7      7
ebx      0x1      1
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eip      0x8049031 0x8049031 <_start+49>
cs       0x23     35
ss       0x2b     43

0x8049072      add     BYTE PTR [eax],al
0x8049016 <_start+22> mov     eax,0x4
0x804901b <_start+27> mov     ebx,0x1
0x8049020 <_start+32> mov     ecx,0x804a008
0x8049025 <_start+37> mov     edx,0x7
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
0x804903e      add     BYTE PTR [eax],al
0x8049040      add     BYTE PTR [eax],al

native process 8552 In: _start
(gdb) x/1sb &msg1
2
031
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb) c
Continuing.
hello, world!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) q
A debugging session is active.

    Inferior 1 [process 8552] will be killed.

Quit anyway? (y or n) y

```

Рис. 3.21: Продолжение и выход

Копирую файл lab8-2.asm в файл с именем lab09-3.asm (рис. 3.22)

```

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

```

Рис. 3.22: Копирование в файл lab09-3.asm

Создаю исполняемый файл, загружаю исполняемый файл в отладчик, указав аргументы. Устанавливаю точку останова перед первой инструкцией в программе и запускаю (рис. 3.23)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab09 $ gdb --args lab09-3 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 lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/k/mkmoroshan/work/arch-pc/lab09/lab09-3 аргумент1 а
ргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffc290: 0x00000005
(gdb)

```

Рис. 3.23: Исполняемый файл, загрузка в отладчик, установка точки останова

Элементы расположены с интервалом в 4 единицы т.к стек может хранить до 4 байт (рис. 3.24)

```

(gdb) x/s *(void**)(esp+4)
0xfffffc512: "/afs/.dk.sci.pfu.edu.ru/home/m/k/mkmoroshan/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp+8)
0xfffffc559: "аргумент1"
(gdb) x/s *(void**)(esp+12)
0xfffffc56b: "аргумент"
(gdb) x/s *(void**)(esp+16)
0xfffffc57c: "2"
(gdb) x/s *(void**)(esp+20)
0xfffffc57e: "аргумент 3"
(gdb) x/s *(void**)(esp+24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 3.24: Позиции стека

### 3.3 Выполнение заданий для самостоятельной работы

Создаю файл lab09-4.asm для задания 1 (рис. 3.25)

```

mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ touch lab09-4.asm

```

Рис. 3.25: Создание файла lab09-4

Изменяю программу, реализовав вычисление значения функции  $f(x)$  как подпрограмму (рис. 3.26)

```
> mc [mkmoroshan@dk5n51.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab09
lab09-4.asm [-----] 3 L: [ 1+31 32/ 33] *(367 / 368b) 0010 0x00A [*][X]
%include "in_out.asm"
SECTION .data
prim DB 'f(x)=2(x-1)',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end
pop eax
call atoi
call fir
add esi,eax
loop next
_end:
mov eax,otv
call sprintf
mov eax,esi
call iprintf
call quit
call quit
fir:
mov ebx,2
mul ebx
add eax,-2
ret
```

Рис. 3.26: Изменение программы

Создаю исполняемый файл и проверяю работу программы (рис. 3.27)

```
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4
f(x)=2(x-1)
Результат: 12
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 3 4 5 6
f(x)=2(x-1)
Результат: 30
```

Рис. 3.27: Создание исполняемого файла и запуск

Создаю файл lab09-5.asm для задания 2 (рис. 3.28)

```
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ touch lab09-5.asm
```

Рис. 3.28: Создание файла lab09-5

Ввожу текст программы из листинга 9.3 (рис. 3.29)

```

> mc [mkmoroshan@dk5n51.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab09
lab09-5.asm [-M--] 9 L:[ 1+19 20/ 20] *(348 / 348b) <EOF> [*][X]
#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 iprintfLF
call quit

```

Рис. 3.29: Текст программы

Создаю исполняемый файл и проверяю, что программа выдает ошибку (рис. 3.30)

```

mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10

```

Рис. 3.30: Проверка программы

Запускаю программу в отладчике (рис. 3.31)

```

mkmoroshan@dk5n51 ~/work/arch-pc/lab09 $ gdb lab09-5
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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/k/mkmoroshan/work/arch-pc/lab09/lab09-5

Breakpoint 1, 0x080490e8 in _start ()
(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   0x804900f <sprint>
      0x0804910a <+34>:   mov     eax,edi
      0x0804910c <+36>:   call   0x8049086 <iprintLF>
      0x08049111 <+41>:   call   0x80490db <quit>
End of assembler dump.

```

Рис. 3.31: Запуск программы в отладчике

Некоторые регистры стоят не на своих местах, поэтому исправляю это (рис. 3.32)

```

> gdb lab09-5
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffc2e0 0xffffc2e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490ed 0x80490ed <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x80490d7 <atoi.restore+1> pop    edx
0x80490d8 <atoi.restore+2> pop    ecx
0x80490d9 <atoi.restore+3> pop    ebx
0x80490da <atoi.restore+4> ret
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

```

Рис. 3.32: Регистры

Запускаю еще раз программу и она выводит правильный ответ 25 (рис. 3.33)

```

mkmoroshan@dk8n60 ~/work/arch-pc/lab09 $ gdb lab09-5
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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/k/mkmoroshan/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 7622) exited normally]

```

Рис. 3.33: Вывод правильного ответа



## 4 Выводы

При выполнении данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм и познакомился с методами отладки при помощи GDB и его основными возможностями.