

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

дисциплина: Архитектура компьютера

Студент: Семёнов Александр Дмитриевич

Группа: НКАбд-05-25

МОСКВА

2025 г.

Содержание

1. Цель работы.

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи **GDB** и его основными возможностями.

2. Задание.

Изучить изменение регистров, модификацию памяти, анализ стека и аргументов командной строки, а также найти и исправить логические ошибки в коде.

3. Выполнение лабораторной работы.

3.1 Реализация подпрограмм в NASM.

Я создал каталог для лабораторной работы, перешел в него, скопировал файл **in_out.asm** и создал файл.

```
adsemyonov@fedora:~$ mkdir ~/work/arch-pc/lab09
adsemyonov@fedora:~$ cd ~/work/arch-pc/lab09
adsemyonov@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/in_out.asm ~^C
adsemyonov@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/in_out.asm ~/work/arch-pc/lab09
adsemyonov@fedora:~/work/arch-pc/lab09$ touch lab09-1.asm
```

Рис. 1. Создание, переход, копирование.

Потом я ввел текст из листинга 9.1.

The screenshot shows a terminal window with the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
0 _start:
1 mov eax, msg
2 call sprint
3 mov ecx, x
4 mov edx, 80
5 call sread
6 mov eax,x
7 call atoi
8 call _calcul
9 mov eax,result
0 call sprint
1 mov eax,[res]
2 call iprintLF
3 call quit
4 _calcul:
5 mov ebx,2
6 mul ebx
7 add eax,7
8 mov [res],eax
9 ret
0 mov eax, msg
1 call sprint
2 mov ecx, x
3 mov edx, 80
4 call sread
5 mov eax,x
6 call atoi
7 mov ebx,2
8 mul ebx
9 add eax,7
0 mov [res],eax
1 ret
```

Рис. 2. Текст программы.

Я создал файл и запустил его, чтобы проверить работу.

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

Рис. 3. Создание и запуск файла.

Далее я изменил текст программы, добавив подпрограмму `_subcalcul` в

подпрограмму **_calcul** для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x)=2x+7$, $g(x)=3x-1$.

The screenshot shows a terminal window with the title "lab09-1.asm" and the path "~/work/arch-pc/lab09". The window contains assembly code for a program named lab09-1.asm. The code includes sections for data (.data), bss (.bss), and text (.text). It defines variables msg and result, and declares _start, _calcul, and _subcalcul as global symbols. The .text section contains code to print a message, read input from the user, call the _calcul subroutine, print the result, and exit. The _calcul subroutine performs the calculation $f(g(x)) = 2(3x-1)+7$. The _subcalcul subroutine performs the calculation $g(x) = 3x-1$.

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2(3x-1)+7=',0
5 SECTION .bss
6 x: RESB 80
7 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax,x
17 call atoi
18 call _calcul
19 mov eax,result
20 call sprint
21 mov eax,[res]
22 call iprintLF
23 call quit
24 _calcul:
25 push eax
26 call _subcalcul
27 mov ebx, 2
28 mul ebx
29 add eax, 7
30 mov [res], eax
31 pop eax
32 ret
33 _subcalcul:
34 mov ebx, 3
35 mul ebx
36 sub eax, 1
37 ret|
```

Рис. 4. Изменение программы.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
adsemyonov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
adsemyonov@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 10
2(3x-1)+7=65
```

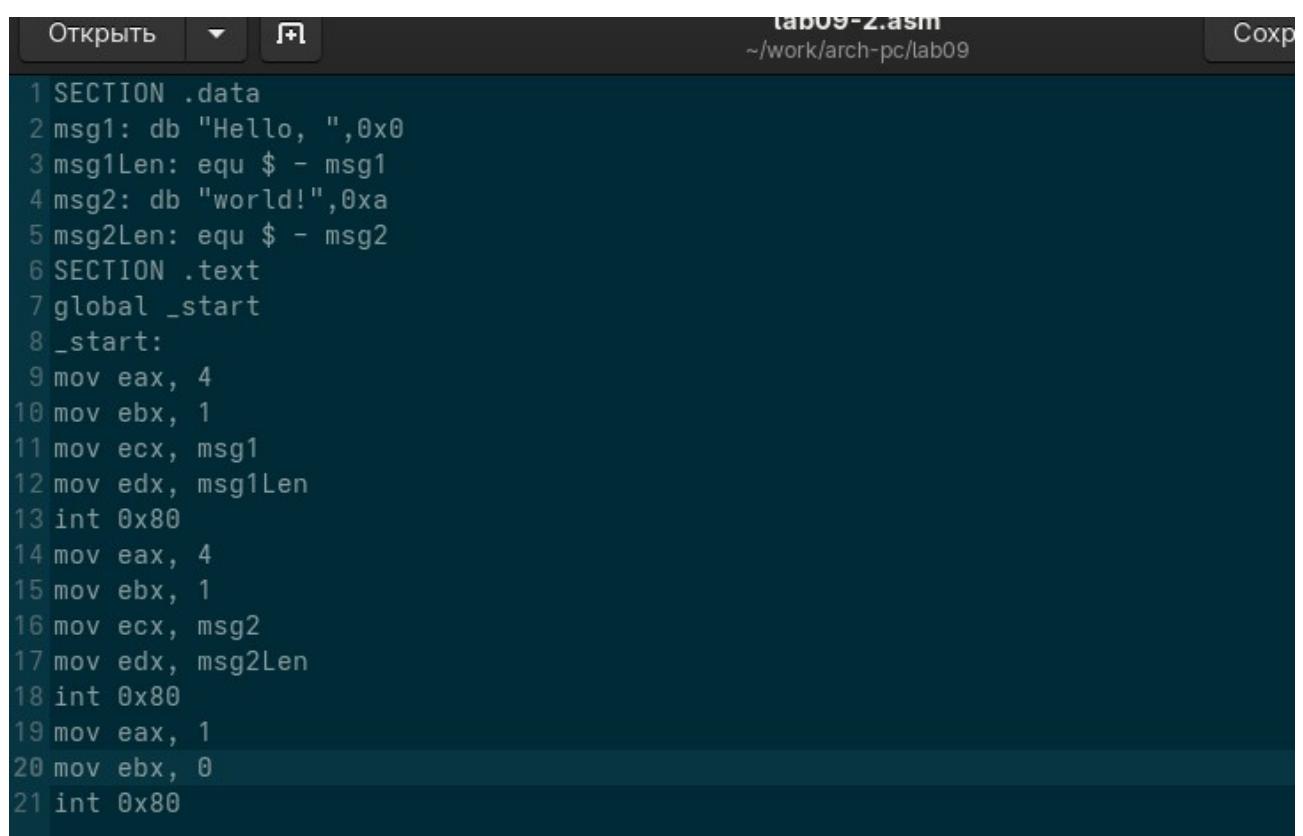
Рис. 5. Создание и запуск файла.

3.2 Отладка программ с помощью GDB.

Я создал файл **lab09-2.asm** и ввел туда текст программы из листинга 9.2.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
```

Рис. 6. Создание файла.



The screenshot shows a text editor window with the file **lab09-2.asm** open. The code is as follows:

```
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
12 mov edx, msg1Len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2Len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 7. Текст программы.

Для работы с **GDB** трансляцию программы я провел с ключом **-g** и загрузил файл в отладчик **GDB**.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
adsemyonov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
adsemyonov@fedora:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 16.3-6.fc43
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...
(gdb) 
```

Рис. 8. Загрузка файла в **gdb**.

```
[Inferior 1 (process 20450) exited normally]
(gdb) run
Starting program: /home/adsemyonov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 20450) exited normally]
```

Рис. 9. Запуск программы с помощью **run**.

Для более подробного анализа программы я установил брейкпоинт на метку **_start**, с которой начинается выполнение любой ассемблерной программы и запустил ее.

```
(gdb) break _start
Breakpoint 1 at 0x8048080: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/adsemyonov/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) 
```

Рис. 10. Брейкпоинт и запуск программы.

Я посмотрел диассимилированный код программы с помощью команды **disassemble**, начиная с метки **_start**.

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    $0x4,%eax
    0x08048085 <+5>:    mov    $0x1,%ebx
    0x0804808a <+10>:   mov    $0x8049000,%ecx
    0x0804808f <+15>:   mov    $0x8,%edx
    0x08048094 <+20>:   int    $0x80
    0x08048096 <+22>:   mov    $0x4,%eax
    0x0804809b <+27>:   mov    $0x1,%ebx
    0x080480a0 <+32>:   mov    $0x8049008,%ecx
    0x080480a5 <+37>:   mov    $0x7,%edx
    0x080480aa <+42>:   int    $0x80
    0x080480ac <+44>:   mov    $0x1,%eax
    0x080480b1 <+49>:   mov    $0x0,%ebx
    0x080480b6 <+54>:   int    $0x80
End of assembler dump.
(gdb) █
```

Рис. 11. Диассимилированный код программы.

Я переключился на отображение команд с **Intel'овским** синтаксисом, введя команду **set disassembly-flavor intel**.

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    eax,0x4
    0x08048085 <+5>:    mov    ebx,0x1
    0x0804808a <+10>:   mov    ecx,0x8049000
    0x0804808f <+15>:   mov    edx,0x8
    0x08048094 <+20>:   int    0x80
    0x08048096 <+22>:   mov    eax,0x4
    0x0804809b <+27>:   mov    ebx,0x1
    0x080480a0 <+32>:   mov    ecx,0x8049008
    0x080480a5 <+37>:   mov    edx,0x7
    0x080480aa <+42>:   int    0x80
    0x080480ac <+44>:   mov    eax,0x1
    0x080480b1 <+49>:   mov    ebx,0x0
    0x080480b6 <+54>:   int    0x80
End of assembler dump.
(gdb) █
```

Рис. 12. Отображение команд с **Intel'овским** синтаксисом.

Различия между синтаксисом ATT и Intel заключаются в порядке operandов (ATT - Operand источника указан первым. Intel - Operand назначения указан первым), их размере (ATT - размер operandов указывается явно с помощью суффиксов, непосредственные operandы предваряются символом \$; Intel — Размер operandов неявно определяется контекстом, как ax, eax, непосредственные operandы пишутся напрямую), именах регистров (ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов)

The screenshot shows the GDB interface on a Fedora system. The title bar reads "adsemyonov@fedora:~/work/arch-pc/lab09 — gdb lab09-2 ~/work/arch-pc/lab09". The main window has two panes. The top pane displays the assembly code starting at address 0x8048080, with instructions like mov eax, 0x4, mov ebx, 0x1, etc. The bottom pane shows the register dump for the "general" group, listing registers eax through es with their current values.

```

Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcf80 0xfffffcf80
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip 0x8048080 0x8048080 <_start>
eflags        0x202      [ IF ]
cs            0x23        35
ss            0x2b        43
ds            0x2b        43
es            0x2b        43

```

```

B+>0x8048080 <_start>    mov    eax,0x4
 0x8048085 <_start+5>    mov    ebx,0x1
 0x804808a <_start+10>   mov    ecx,0x8049000
 0x804808f <_start+15>   mov    edx,0x8
 0x8048094 <_start+20>   int    0x80
 0x8048096 <_start+22>   mov    eax,0x4
 0x804809b <_start+27>   mov    ebx,0x1
 0x80480a0 <_start+32>   mov    ecx,0x8049008
 0x80480a5 <_start+37>   mov    edx,0x7
 0x80480aa <_start+42>   int    0x80
 0x80480ac <_start+44>   mov    eax,0x1
 0x80480b1 <_start+49>   mov    ebx,0x0
 0x80480b6 <_start+54>   int    0x80
 0x80480b8          add    BYTE PTR [eax],al

```

```

Native process 21034 (asm) In: _start
(gdb) layout regs

```

Рис. 13. Включение режима псевдографики

3.3 Добавление точек останова.

Я проверил, что точка останова по имени метки `_start` была установлена, с помощью команды `info breakpoints`.

```
(gdb) info breakpoints
(gdb) Num      Type            Disp Enb Address     What
1       breakpoint      keep y  0x08048080 lab09-2.asm:9
      breakpoint already hit 1 time
```

Рис. 14. Проверка установки точки останова.

Я определил адрес предпоследней инструкции и установил точку останова и посмотрел информацию о всех установленных точках останова.

```
(gdb) b *0x80480b1
Breakpoint 2 at 0x80480b1: file lab09-2.asm, line 20.
(gdb) i b
Num      Type            Disp Enb Address     What
1       breakpoint      keep y  0x08048080 lab09-2.asm:9
      breakpoint already hit 1 time
2       breakpoint      keep y  0x080480b1 lab09-2.asm:20
(gdb)
```

Рис. 15. Определение адреса инструкции и просмотр информации.

Я выполнила 5 инструкций с помощью команды **stepi** и проследила за изменением значений регистров.

Register group: general		
eax	0x4	4
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xfffffcf80	0xfffffcf80
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048085	0x8048085 <_start+5>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

Рис. 16. Изменение №1

Register group: general		
eax	0x4	4
ecx	0x0	0
edx	0x0	0
ebx	0x1	1
esp	0xfffffcf80	0xfffffcf80
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x804808a	0x804808a <_start+10>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

Рис. 17. Изменение №2

Register group: general		
eax	0x4	4
ecx	0x8049000	134516736
edx	0x0	0
ebx	0x1	1
esp	0xfffffcf80	0xfffffcf80
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x804808f	0x804808f <_start+15>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

Рис. 18. Изменение №3

Register group: general		
eax	0x4	4
ecx	0x8049000	134516736
edx	0x8	8
ebx	0x1	1
esp	0xfffffcf80	0xfffffcf80
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048094	0x8048094 <_start+20>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

Рис. 19. Изменение №4

Register group: general		
eax	0x8	8
ecx	0x8049000	134516736
edx	0x8	8
ebx	0x1	1
esp	0xfffffcf80	0xfffffcf80
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048096	0x8048096 <_start+22>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

Рис. 20. Изменение №5

Изменялись регистры: eax, ebx, ecx, edx, eip.

Я посмотрел значение переменной msg1 по имени.

```
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "Hello, "
(gdb)
```

Рис. 21. Значение **msg1** по имени.

Я изменил первый символ переменной **msg1**.

```
(gdb) set {char}0x8049000='h'
(gdb) x/1sb 0x8049000
0x8049000 <msg1>:      "hello, "
(gdb) █
```

Рис. 22. Изменение первого символа.

Еще я заменил первый символ во второй переменной **msg2**.

```
(gdb) set {char}0x8049008='f'
(gdb) x/1sb 0x8049008
0x8049008 <msg2>:      "forld!\n\034"
(gdb) █
```

Рис. 23. Изменение символа во второй переменной.

Я вывел в различных форматах значение регистра **edx**, с помощью команды **set** я изменил значение регистра **ebx**.

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) p/x $ebx
$2 = 0x32
(gdb) p/x $edx
$3 = 0x8
(gdb) p/t $edx
$4 = 1000
(gdb) p/c $edx
$5 = 8 '\b'
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)
```

Рис. 24. Вывод в различных форматах.

Команда **'p/s'** выводит числовое значение регистра в десятичном виде, поэтому при присваивании символу **'2'** (**ASCII-код 50**) выводится 50, а при присваивании числа **'2'** выводится 2.

Я завершил выполнение программы с помощью команды **continue (c)** и вышел.

```
(gdb) continue
Continuing.
forld!
Breakpoint 2, _start () at lab09-2.asm:20
```

Рис. 25. Завершение программы.

3.4 Обработка аргументов командной строки в GDB.

Я создал файл **lab09-3.asm**.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ touch lab09-3.asm
```

Рис. 26. Создание файла.

Я скопировал файл **lab8-2.asm** в файл с именем **lab09-3.asm**.

Создал исполняемый файл, использовал ключ **args** и загрузил исполняемый файл в отладчик, указав аргументы.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab093.lst lab09-3.asm
adsemyonov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
adsemyonov@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 16.3-6.fc43
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. 28. Создание файла и загрузка в отладчик.

Я установил точку останова перед первой инструкцией в программе и запустил ее.

```

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 0x8048148: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/adsemyonov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<ima:enforcing>
<https://debuginfod.fedoraproject.org/>
<ima:ignore>
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.
Downloading separate debug info for system-supplied DSO at 0xf7ffb000

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx
(gdb)

```

Рис. 29. Установка точки останова.

Адрес вершины стека хранится в регистре **esp** и по этому адресу располагается число равное количеству аргументов командной строки. Я посмотрел остальные позиции стека.

```

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
(gdb) x/x $esp
0xfffffcf50: 0x00000005
(gdb) x/s *(void**)( $esp + 4 )
0xfffffd139: "/home/adsemyonov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8 )
0xfffffd165: "аргумент1"
(gdb) x/s *(void**)( $esp + 12 )
0xfffffd177: "аргумент"
(gdb) x/s *(void**)( $esp + 16 )
0xfffffd188: "2"
(gdb) x/s *(void**)( $esp + 20 )
0xfffffd18a: "аргумент 3"
(gdb) x/s *(void**)( $esp + 24 )
0x0: <error: Cannot access memory at address 0x0>
(gdb) 

```

Рис. 30. Адрес вершины и остальные позиции стека.

Шаг изменения адреса равен 4, потому что в 32-битной архитектуре размер указателя составляет 4 байта, и аргументы командной строки передаются в стек как массив таких указателей, каждый из которых занимает 4 байта.

4. Выполнение самостоятельной работы.

Задание №1

Я создал файл для выполнения самостоятельной работы.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ touch lab09-4.asm
adsemyonov@fedora:~/work/arch-pc/lab09$
```

Рис. 31. Создание файла.

Затем я изменил программу из лабораторной работы №8.

```
~/work/arch-pc/lab09
1 %include 'in_out.asm'
2 SECTION .data
3 msg_func db "Функция: f(x) = 7 + 2x", 0
4 msg_result db "Результат: ", 0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 mov eax, msg_func
9 call sprintLF
10 pop ecx
11 pop edx
12 sub ecx, 1
13 mov esi, 0
14 next:
15 cmp ecx, 0h
16 jz _end
17 pop eax
18 call atoi
19 call _calculate_fx
20 add esi, eax
21 loop next
22 _end:
23 mov eax, msg_result
24 call sprint
25 mov eax, esi
26 call iprintLF
27 call quit
28 _calculate_fx:
29 mov ebx, 8
30 mul ebx
31 sub eax, 3
32 ret
```

Рис. 32. Изменение программы.

Я создал и запустил файл.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-4.lst lab09-4.asm
adsemyonov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
adsemyonov@fedora:~/work/arch-pc/lab09$ gdb lab09-4
GNU gdb (Fedora Linux) 16.3-6.fc43
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 lab09-4...
(gdb) b _start
Breakpoint 1 at 0x8048168: file lab09-4.asm, line 8.
(gdb) run
Starting program: /home/adsemyonov/work/arch-pc/lab09/lab09-4

This GDB supports auto-downloading debuginfo from the following URLs:
  <ima:enforcing>
  <https://debuginfod.fedoraproject.org/>
  <ima:ignore>
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 lab09-4.asm:8
8      mov eax, msg_func
(gdb) run 2 2 7
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/adsemyonov/work/arch-pc/lab09/lab09-4 2 2 7

Breakpoint 1, _start () at lab09-4.asm:8
8      mov eax, msg_func
(gdb) █
```

Рис. 33. Создание и запуск файла.

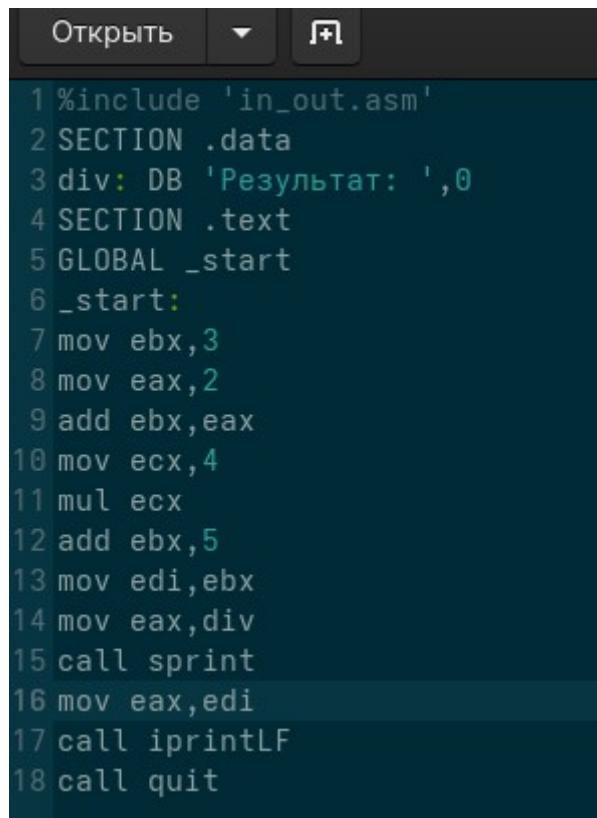
Задание №2

Я создал файл для выполнения задания №2.

```
adsemyonov@fedora:~/work/arch-pc/lab09$ touch lab09-5.asm
adsemyonov@fedora:~/work/arch-pc/lab09$
```

Рис. 34. Создание файла.

Я ввел программу из листинга 9.3.



The screenshot shows a debugger's assembly view. The title bar has a 'File' menu with 'Открыть' (Open) and a '+' icon. The assembly code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov ebx,3
8 mov eax,2
9 add ebx,eax
10 mov ecx,4
11 mul ecx
12 add ebx,5
13 mov edi,ebx
14 mov eax,div
15 call sprint
16 mov eax,edi
17 call iprintLF
18 call quit
```

Рис. 35. Текст программы.

Я запустил программу в режиме отладчика и пошагово через **si** просмотрела изменение значений регистров через **i r**.

The screenshot shows a debugger interface with two main panes. The top pane displays the general register group, listing registers eax, ecx, edx, ebx, esp, ebp, esi, edi, eip, and eflags with their current values. The bottom pane shows the assembly code starting at address 0x8048172, which includes calls to functions like `sprint`, `iprintLF`, and `quit`.

Register group: general		
eax	0x2	2
ecx	0x0	0
edx	0x0	0
ebx	0x5	5
esp	0xfffffcfa0	0xfffffcfa0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048174	0x8048174 <_start+12>
eflags	0x206	[PF IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

```
0x8048172 <_start+10> add    %eax,%ebx
>0x8048174 <_start+12> mov    $0x4,%ecx
0x8048179 <_start+17> mul    %ecx
0x804817b <_start+19> add    $0x5,%ebx
0x804817e <_start+22> mov    %ebx,%edi
0x8048180 <_start+24> mov    $0x8049000,%eax
0x8048185 <_start+29> call   0x804808f <sprint>
0x804818a <_start+34> mov    %edi,%eax
0x804818c <_start+36> call   0x8048106 <iprintLF>
0x8048191 <_start+41> call   0x804815b <quit>
0x8048196 add    %al,(%eax)
0x8048198 add    %al,(%eax)
0x804819a add    %al,(%eax)
0x804819c add    %al,(%eax)
```

Рис. 36. Просмотр.

The screenshot shows a debugger interface with two main sections. The top section displays the general register group:

Register group: general		
eax	0x2	2
ecx	0x4	4
edx	0x0	0
ebx	0x5	5
esp	0xfffffcfa0	0xfffffcfa0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048179	0x8048179 <_start+17>
eflags	0x206	[PF IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43

The bottom section shows the assembly code starting at address 0x8048172:

```

0x8048172 <_start+10>    add    %eax,%ebx
0x8048174 <_start+12>    mov    $0x4,%ecx
>0x8048179 <_start+17>    mul    %ecx
0x804817b <_start+19>    add    $0x5,%ebx
0x804817e <_start+22>    mov    %ebx,%edi
0x8048180 <_start+24>    mov    $0x8049000,%eax
0x8048185 <_start+29>    call   0x804808f <sprint>
0x804818a <_start+34>    mov    %edi,%eax
0x804818c <_start+36>    call   0x8048106 <iprintLF>
0x8048191 <_start+41>    call   0x804815b <quit>
0x8048196
0x8048198
0x804819a
0x804819c

```

Рис. 37. Просмотр.

The screenshot shows a debugger interface with two main sections. The top section displays the general register group:

Register group: general		
eax	0x2	2
ecx	0x4	4
edx	0x0	0
ebx	0x5	5
esp	0xfffffcfa0	0xfffffcfa0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048179	0x8048179 <_start+17>
eflags	0x206	[PF IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43
fs	0x0	0

The bottom section shows the assembly code starting at address 0x8048172:

```
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 38. Просмотр.

При выполнении инструкции **mul ecx** можно заметить, что результат умножения записывается в регистр **eax**, но также меняет и **edx**. Значение регистра **ebx** не обновляется напрямую, поэтому результат программы неверно подсчитывает функцию.

The screenshot shows a terminal window with the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov ebx,3
8 mov eax,2
9 add ebx,eax
10 mov eax,ebx
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 mov eax,div
16 call sprint
17 mov eax,edi
18 call iprintLF
19 call quit
```

Рис. 39. Изменение программы.

Я создал и запустил файл.

```
[adsemyonov@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
adsemyonov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
adsemyonov@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
```

Рис. 40. Создание и запуск файла.

5. Выводы.

Я приобрел навыков написания программ с использованием подпрограмм и познакомился с методами отладки при помощи **GDB** и его основными возможностями.

Список литературы.

1. <https://esystem.rudn.ru/course/view.php?id=112>