

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Логинов Георгий Евгеньевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Задания для самостоятельной работы	20
6	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Создание файла и директории	8
4.2	Редактирование файла	9
4.3	Запуск исполняемого файла	9
4.4	Редактирование файла	10
4.5	Запуск исполняемого файла	11
4.6	Создание и редактирование файла	11
4.7	Запуск программы в отладочной оболочке	12
4.8	Запуск программы в отладочной оболочке	12
4.9	Просмотр дисассимилированного кода программы	13
4.10	Изменение синтаксиса	14
4.11	Включение псевдографики	14
4.12	Проверка	15
4.13	Добавление точки остова и проверка её наличия	15
4.14	Выполнение некоторых инструкций	16
4.15	Просмотр значения переменной	16
4.16	Просмотр значения переменной	16
4.17	Изменение символа в переменной	17
4.18	Изменение символа в переменной	17
4.19	Вывод значения в разных форматах	17
4.20	Изменение значения решистра	18
4.21	Копирование файла и создание исполняемого	18
4.22	Запуск программы в оболочке отладки	18
4.23	Количество аргументов 5	19
4.24	Просмотр позиций стека	19
5.1	Исполнение файла	20
5.2	Создание и редактирование файла	22
5.3	Несостыковка	22
5.4	Несостыковка	23
5.5	Исполнение файла	23

Список таблиц

1 Цель работы

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

2 Задание

Реализация подпрограмм в NASM Отладка программ с помощью GDB Задание для самостоятельной работы

3 Теоретическое введение

Подпрограмма — поименованная или иным образом идентифицированная часть компьютерной программы, содержащая описание определённого набора действий.

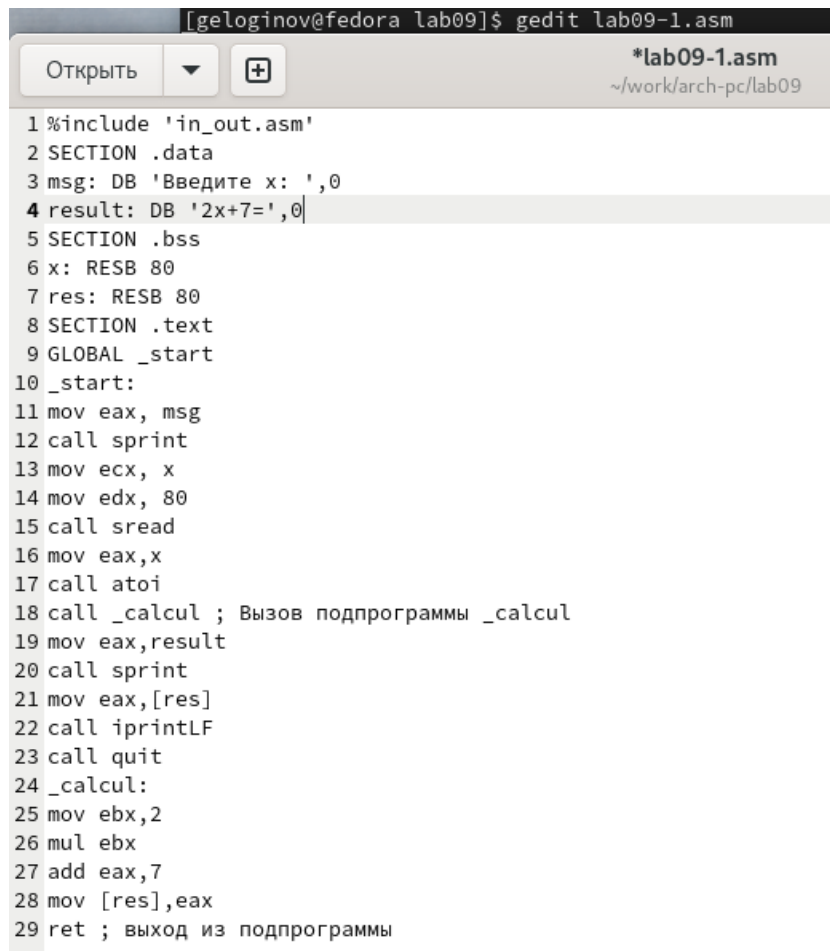
4 Выполнение лабораторной работы

##Реализация подпрограмм в NASM Создаю рабочую директорию и файл в ней. (рис. 4.1).

```
[geloginov@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[geloginov@fedora ~]$ cd ~/work/arch-pc/lab09  
[geloginov@fedora lab09]$ touch lab09-1.asm  
[geloginov@fedora lab09]$
```

Рис. 4.1: Создание файла и директории

Редактирую файл вводя текст листинга. (рис. 4.2).

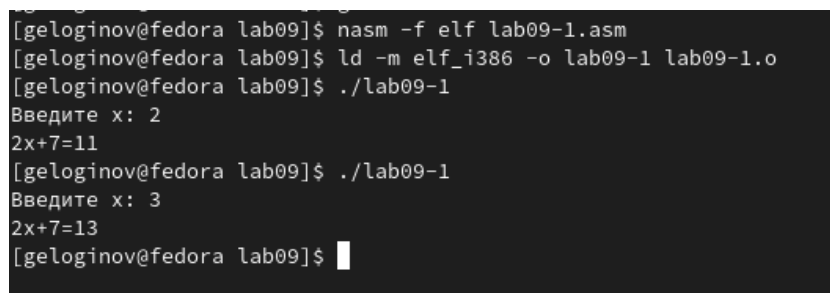


```
[geloginov@fedora lab09]$ gedit lab09-1.asm
Открыть ▼ + *lab09-1.asm
~/work/arch-pc/lab09

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
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 ; Вызов подпрограммы _calcul
19 mov eax, result
20 call sprint
21 mov eax, [res]
22 call iprintLF
23 call quit
24 _calcul:
25 mov ebx, 2
26 mul ebx
27 add eax, 7
28 mov [res], eax
29 ret ; выход из подпрограммы
```

Рис. 4.2: Редактирование файла

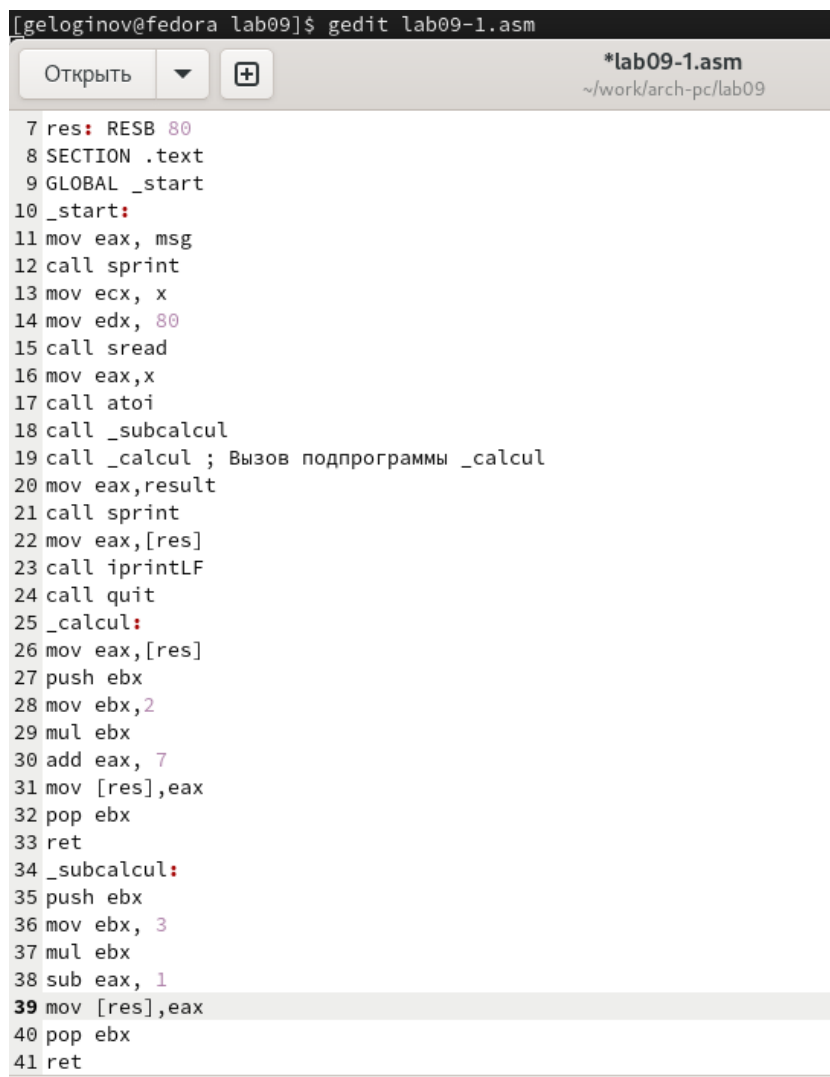
Создаю исполняемый файл. Проверяю корректность работы программы (рис. 4.3). Программа отработала корректно.



```
[geloginov@fedora lab09]$ nasm -f elf lab09-1.asm
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[geloginov@fedora lab09]$ ./lab09-1
Введите x: 2
2x+7=11
[geloginov@fedora lab09]$ ./lab09-1
Введите x: 3
2x+7=13
[geloginov@fedora lab09]$
```

Рис. 4.3: Запуск исполняемого файла

Изменяю текст программы чтобы она считала $f(g(x))$. (рис. 4.4).



```
[geloginov@fedora lab09]$ gedit lab09-1.asm
Открыть  *lab09-1.asm
~/work/arch-pc/lab09

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 _subcalcul
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax, result
21 call sprint
22 mov eax, [res]
23 call iprintLF
24 call quit
25 _calcul:
26 mov eax, [res]
27 push ebx
28 mov ebx, 2
29 mul ebx
30 add eax, 7
31 mov [res], eax
32 pop ebx
33 ret
34 _subcalcul:
35 push ebx
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 mov [res], eax
40 pop ebx
41 ret
```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл. Проверяю корректность работы программы (рис. 4.5). Программа отработала корректно.

```

[geloginov@fedora lab09]$ nasm -f elf lab09-1.asm
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[geloginov@fedora lab09]$ ./lab09-1
Введите x: 2
f(g(x))=17
[geloginov@fedora lab09]$ ./lab09-1
Введите x: 3
f(g(x))=23
[geloginov@fedora lab09]$

```

Рис. 4.5: Запуск исполняемого файла

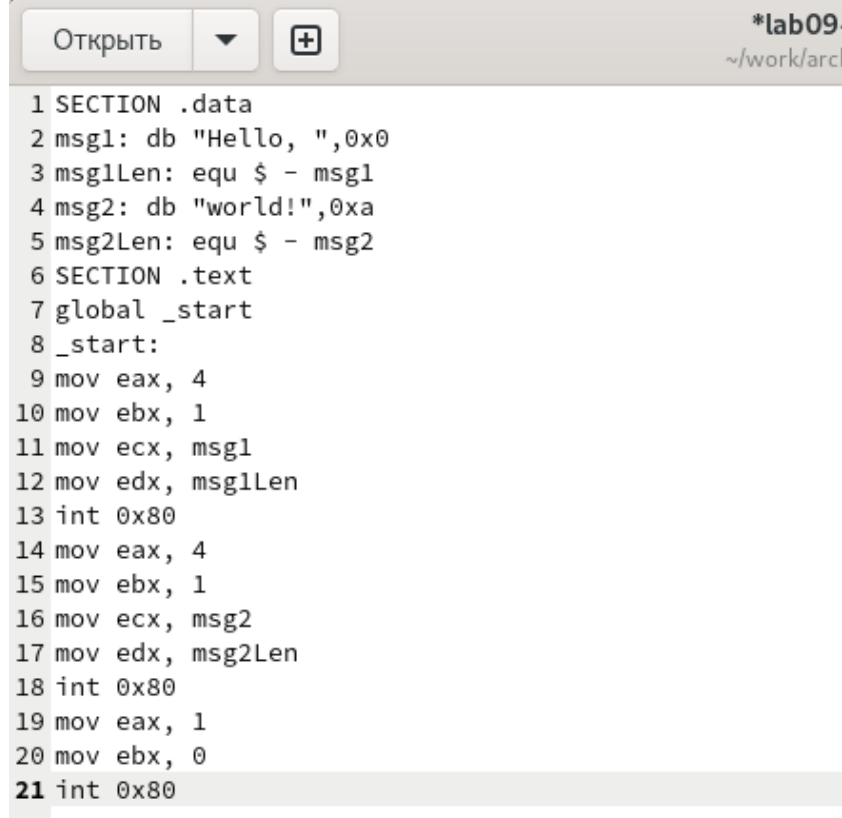
##Отладка программ с помощью GDB

Создаю файл. Ввожу в него текст листинга (рис. 4.6).

```

[geloginov@fedora lab09]$ touch lab09-2.asm
[geloginov@fedora lab09]$ gedit lab09-2.asm

```



```

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. 4.6: Создание и редактирование файла

Запускаю программу в отладочной оболочке GDB (рис. 4.7).

```

Starting program: /home/geloginov/work/arch-pc/lab09/lab09-2

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.
Hello, world!
[Inferior 1 (process 5357) exited normally]

```

Рис. 4.7: Запуск программы в отладочной оболочке

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. 4.8).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/geloginov/work/arch-pc/lab09/lab09-2

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

```

Рис. 4.8: Запуск программы в отладочной оболочке

Смотрю дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 4.9).

```

(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. 4.9: Просмотр дисассимилированного кода программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 4.10). В представлении ATТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении `intel` так записываются адреса вторых аргументов.

```

(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. 4.10: Изменение синтаксиса

Включаю режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. 4.11).

```

[ Register Values Unavailable ]

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 5380 In: _start          L9    PC: 0x8049000
(gdb) layout regs
(gdb)

```

Рис. 4.11: Включение псевдографики

Проверяю наличие точки останова.(рис. 4.12).

```

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

```

Рис. 4.12: Проверка

Добавляю ещё одну точку останова и проверяю сколько точек останова есть(рис. 4.13).

```

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 5601 In: _start
(gdb) layout regs
(gdb) b *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. 4.13: Добавление точки остова и проверка её наличия

Выполняю 5 инструкций с помощью команды si(рис. 4.14). Изменились значения регистров eax ecx edx ebx

```

--Register group: general--
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1e0 0xffffd1e0
ebp      0x0      0x0

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

active process 3326 In: _start L14 PC: 0x8049016
0 0
s 0x0 0
gdb) si
gdb) si
gdb) si
gdb) si
gdb) si

```

Рис. 4.14: Выполнение некоторых инструкций

Просматриваю значение переменной msg1 по имени(рис. 4.15).

```

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

```

Рис. 4.15: Просмотр значения переменной

Просматриваю значение переменной msg2 по адресу(рис. 4.16).

```

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

```

Рис. 4.16: Просмотр значения переменной

Изменяю первый символ переменной msg1 (рис. 4.17).


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

Рис. 4.17: Изменение символа в переменной

Изменяю первый символ переменной msg2 (рис. 4.18).

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

Рис. 4.18: Изменение символа в переменной

Вывожу значение edx в разных форматах: строчном, 16-ричном, двоичном (рис. 4.19).

```
(gdb) p/s $edx
$1 = 8
(gdb) p/x
$2 = 0x8
(gdb) p/t
$3 = 1000
(gdb) □
```

Рис. 4.19: Вывод значения в разных форматах

С помощью команды set изменяю значение регистра ebx(рис. 4.20). Разница вывода из-за того что в первом случае 2 это символ а во втором число.

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

Рис. 4.20: Изменение значения регистра

Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы No8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm:(рис. 4.21). И создал исполняемый файл.

```
[geloginov@fedora lab09]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
[geloginov@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab9-3.o
ld: невозможно найти lab9-3.o: Нет такого файла или каталога
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[geloginov@fedora lab09]$
```

Рис. 4.21: Копирование файла и создание исполняемого

Запускаю программу в оболочке GDB (рис. 4.22).

```
[geloginov@fedora lab09]$ gdb --args lab09-3 arg1 arg 2 'arg 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
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-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) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/geloginov/work/arch-pc/lab09/lab09-3 arg1 arg 2 arg\ 3
```

Рис. 4.22: Запуск программы в оболочке отладки

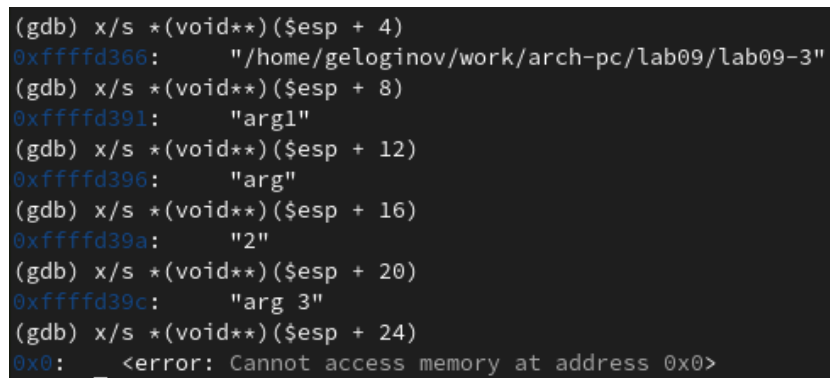
Узнаю количество аргументов (рис. 4.23).



```
(gdb) x/x $esp
0xfffffd1c0:      0x00000005
```

Рис. 4.23: Количество аргументов 5

Смотрю все позиции стека (рис. 4.24). Их адреса располагаются в 4 байтах друг от друга (именно столько занимает элемент стека)



```
(gdb) x/s *(void**)( $esp + 4)
0xfffffd366:      "/home/geloginov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffd391:      "arg1"
(gdb) x/s *(void**)( $esp + 12)
0xfffffd396:      "arg"
(gdb) x/s *(void**)( $esp + 16)
0xfffffd39a:      "2"
(gdb) x/s *(void**)( $esp + 20)
0xfffffd39c:      "arg 3"
(gdb) x/s *(void**)( $esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
```

Рис. 4.24: Просмотр позиций стека

5 Задания для самостоятельной работы

Работа файла из лабораторной работы 8, но с использованием подпрограмм (рис. 5.1).

```
[geloginov@fedora lab09]$ nasm -f elf lab09-4.asm
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o
[geloginov@fedora lab09]$ ./lab09-4
функция: f(x)=3(10 + x)
результат: 0
[geloginov@fedora lab09]$ ./lab09-4 1 2 3 4
функция: f(x)=3(10 + x)
результат: 150
[geloginov@fedora lab09]$
```

Рис. 5.1: Исполнение файла

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
f_x db "функция: f(x)=3(10 + x)",0h
```

```
msg db 10,13,'результат: ',0h
```

```
SECTION .text
```

```
global _start
```

```
_f:
```

```
push ebx
```

```
add eax, 10
```

```
mov ebx, 3
```

```

mul ebx
pop ebx
ret

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi
call _f
add esi, eax

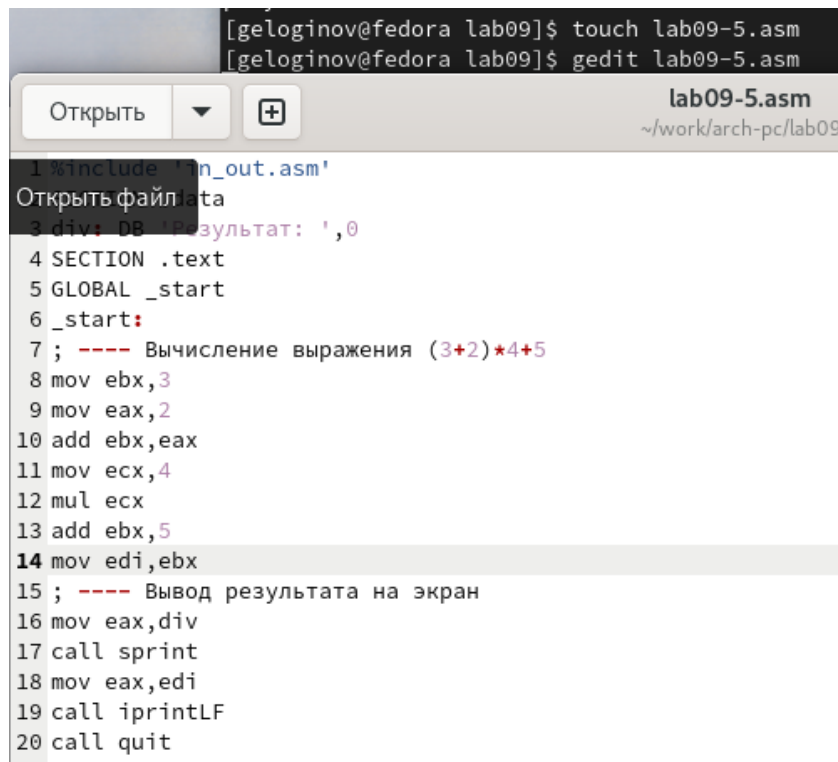
loop next

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

call quit

```

Создаю и редактирую файл (рис. 5.2).

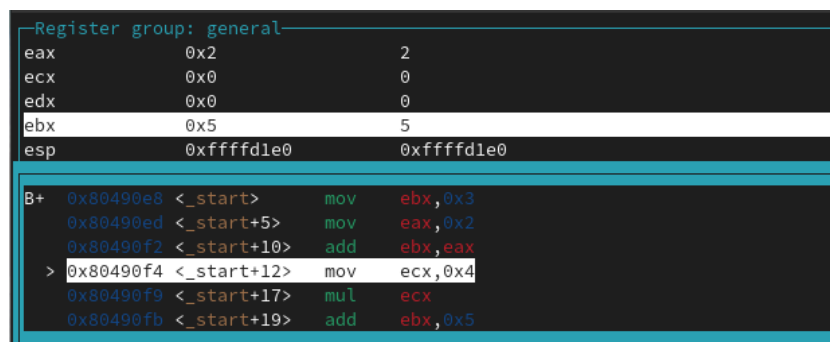


The image shows a terminal window at the top with the following commands: `[geloginov@fedora lab09]$ touch lab09-5.asm` and `[geloginov@fedora lab09]$ gedit lab09-5.asm`. Below the terminal is a window of the gedit text editor. The title bar shows "lab09-5.asm" and the path "~/.work/arch-pc/lab09". The editor contains the following assembly code:

```
1 %include 'in_out.asm'
Открыть файл data
3 div: 00 '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 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

Рис. 5.2: Создание и редактирование файла

Нахожу несостыковки при использовании отладчика(рис. 5.3).



The image shows a debugger window with the title "Register group: general". It displays the following register values:

Register	Value
eax	0x2
ecx	0x0
edx	0x0
ebx	0x5
esp	0xffffd1e0

Below the register window, a list of assembly instructions is shown with their addresses and disassembled code:

```
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
```

Рис. 5.3: Несостыковка

Нахожу несостыковки при использовании отладчика(рис. 5.4).

Register group: general		
eax	0x8	8
ecx	0x4	4
edx	0x0	0
ebx	0xa	10
esp	0xffffd1e0	0xffffd1e0

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>

Рис. 5.4: Несостыковка

Работа исправленного файла(рис. 5.5).

```
[geloginov@fedora lab09]$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
[geloginov@fedora lab09]$ ld -m elf_i386 -o lab09-5 lab09-5.o
[geloginov@fedora lab09]$ ./lab09-5
Результат: 25
```

Рис. 5.5: Исполнение файла

Ошибка была в строчках

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

Исправленный код

```
%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 eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```


6 Выводы

В результате выполнения работы, я научился организовывать код в подпрограммы и познакомился с базовыми функциями отладчика gdb.

Список литературы

Лабораторная работа 9