

# **Отчет по лабораторной работе №9**

**Понятие подпрограммы. Отладчик GDB.**

Полина Алексеевна Ларионова

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>3</b>	<b>Задания для самостоятельной работы</b>	<b>17</b>
<b>4</b>	<b>Выводы</b>	<b>22</b>

# Список иллюстраций

2.1	Создание каталога . . . . .	5
2.2	Листинг 9.1 . . . . .	6
2.3	Работа программы . . . . .	6
2.4	Измененный текст . . . . .	7
2.5	Результат . . . . .	7
2.6	Листинг 9.2 . . . . .	8
2.7	Работа программы в отладчике . . . . .	9
2.8	Брейкпоинт . . . . .	9
2.9	Дисассимилированный код программы . . . . .	10
2.10	Intel . . . . .	10
2.11	Режим псевдографики . . . . .	11
2.12	Точка останова . . . . .	11
2.13	Все точки останова . . . . .	12
2.14	Регистры . . . . .	12
2.15	msg1 . . . . .	13
2.16	msg2 . . . . .	13
2.17	Изменение значения для регистра . . . . .	13
2.18	Замена символов . . . . .	13
2.19	Различные форматы edx . . . . .	14
2.20	Изменение значения ebx . . . . .	14
2.21	Аргументы . . . . .	15
2.22	Breakpoint_start . . . . .	15
2.23	Вершина стека . . . . .	15
2.24	Остальные позиции . . . . .	16
3.1	Преобразованная программа . . . . .	17
3.2	Листинг 9.3 . . . . .	18
3.3	Исполняемый файл . . . . .	18
3.4	Программа в отладчике . . . . .	19
3.5	Ошибка в регистрах . . . . .	20
3.6	Верный ответ . . . . .	21

# 1 Цель работы

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

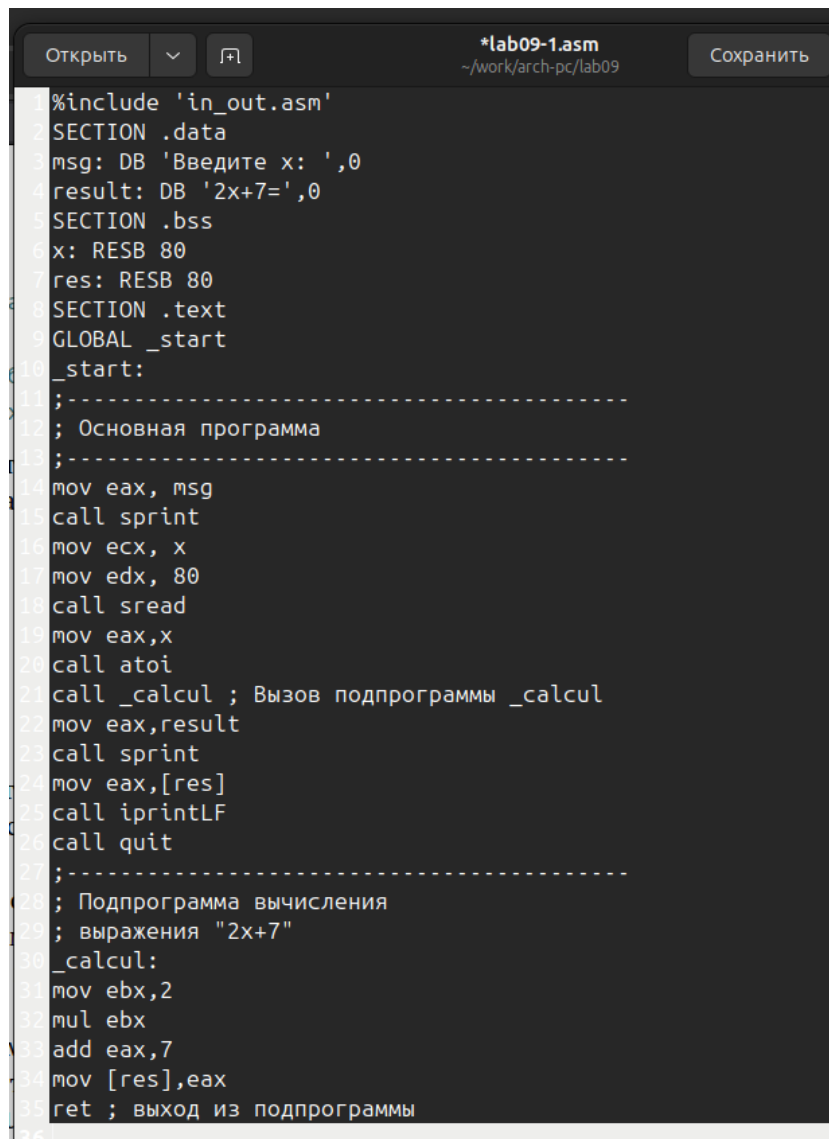
## 2 Выполнение лабораторной работы

Я создала каталог для выполнения лабораторной работы и создала файл lab09-1.asm.

```
palarionova@linux:~$ mkdir ~/work/arch-pc/lab09
palarionova@linux:~$ cd ~/work/arch-pc/lab09
palarionova@linux:~/work/arch-pc/lab09$ touch lab09-1.asm
palarionova@linux:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание каталога

Затем я ввела в файл текст программы из листинга 9.1



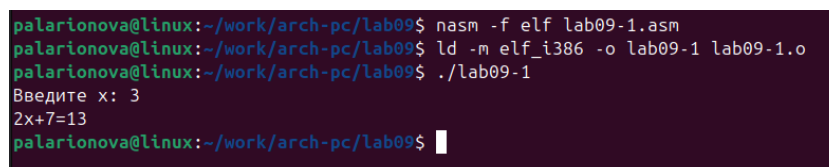
```

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 ;-----
12 ; Основная программа
13 ;-----
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintLF
26 call quit
27 ;-----
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx, 2
32 mul ebx
33 add eax, 7
34 mov [res], eax
35 ret ; выход из подпрограммы
36

```

Рис. 2.2: Листинг 9.1

и создала исполняемый файл.



```

palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
palarionova@linux:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
2x+7=13
palarionova@linux:~/work/arch-pc/lab09$

```

Рис. 2.3: Работа программы

Я изменила текст программы, добавив подпрограмму subcalcul\_

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 prim1: DB '2x+7=',0
5 prim2: DB '3x-1=',0
6 result: DB 'f(g(x))= ',0
7 SECTION .bss
8 x: RESB 80
9 res: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, prim1
14 call sprint
15 mov eax, prim2
16 call sprintLF
17 mov eax, msg
18 call sprint
19 mov ecx, x
20 mov edx, 80
21 call sread
22 mov eax, x
23 call atoi
24 call _calcul
25 mov eax, result
26 call sprint
27 mov eax, [res]
28 call iprintLF
29 call quit
30 _calcul:
31 call _subcalcul
32 mov ebx, 2
33 mul ebx
34 add eax, 7
35 mov [res], eax
36 _subcalcul:
37 mov ebx, 3
38 mul ebx
39 sub eax, 1
40 ret ; выход из подпрограммы

```

Рис. 2.4: Измененный текст

и проверила его работу.

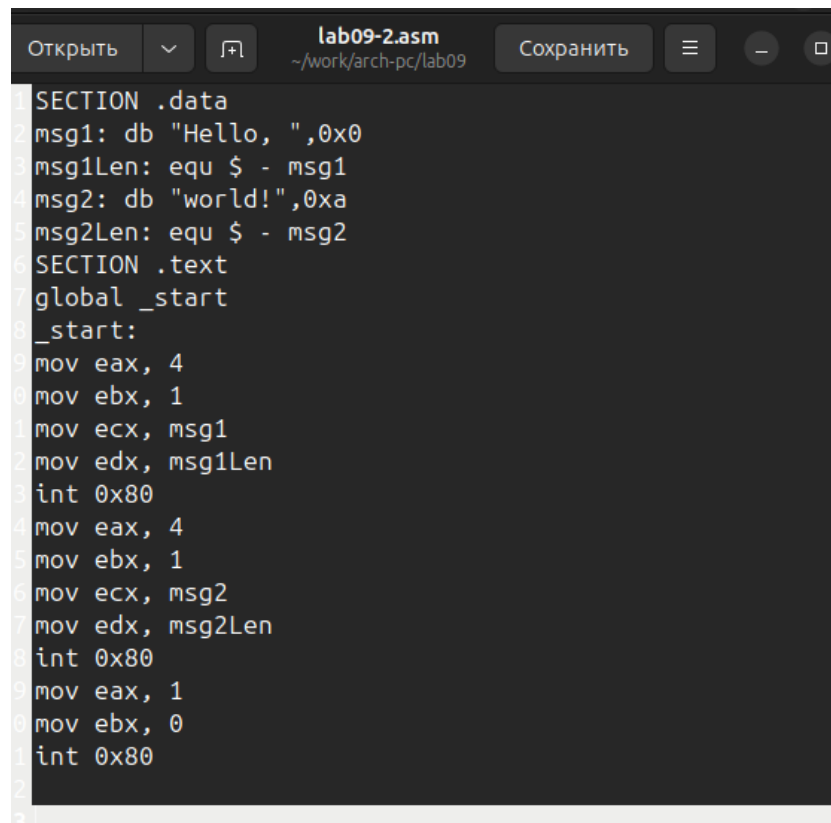
```

palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
palarionova@linux:~/work/arch-pc/lab09$ ./lab09-1
2x+7=3x-1=
Введите x: 3
f(g(x))= 23
palarionova@linux:~/work/arch-pc/lab09$

```

Рис. 2.5: Результат

Я создала файл lab09-2.asm с текстом программы из листинга 9.2.



```
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
22
23
```

Рис. 2.6: Листинг 9.2

Далее я добавила в исполняемый файл отладочную информацию и загрузила исполняемый файл в отладчик gdb, проверив работу программы.



```

palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
palarionova@linux:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
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-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) r
Starting program: /home/palarionova/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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 0xf7ffc000
Hello, world!
[Inferior 1 (process 5376) exited normally]
(gdb) █

```

Рис. 2.7: Работа программы в отладчике

Для более подробного анализа программы я установила брейкпоинт на метку `_start`.

```

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

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

```

Рис. 2.8: Брейкпоинт

Затем я посмотрела дисассимилированный код программы

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
    0x08049005 <+5>:      mov     $0x1,%ebx
    0x0804900a <+10>:     mov     $0x804a000,%ecx
    0x0804900f <+15>:     mov     $0x8,%edx
    0x08049014 <+20>:     int     $0x80
    0x08049016 <+22>:     mov     $0x4,%eax
    0x0804901b <+27>:     mov     $0x1,%ebx
    0x08049020 <+32>:     mov     $0x804a008,%ecx
    0x08049025 <+37>:     mov     $0x7,%edx
    0x0804902a <+42>:     int     $0x80
    0x0804902c <+44>:     mov     $0x1,%eax
    0x08049031 <+49>:     mov     $0x0,%ebx
    0x08049036 <+54>:     int     $0x80
End of assembler dump.
(gdb) █

```

Рис. 2.9: Дисассимилированный код программы

и переключилась на отображение команд с синтаксисом Intel.

```

(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. 2.10: Intel

Отличие заключается в программах: в дисассимилированном отображении в командах используются % и \$, когда в Intel отображении они не используются.

Я включила режим псевдографики.

```

Register group: general
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xffffce0 0 ebp 0x0 0x049000
esi 0x0 0 edi 0x0 0 eip 0x049000 <_start>
eflags 0x282 [ IF ] cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0

0x049000 <_start> mov eax,0x0
0x049005 <_start+5> mov ebx,0x1
0x04900a <_start+10> mov ecx,0x049000
0x04900f <_start+15> mov edx,0x0
0x049014 <_start+20> int 0x0
0x049019 <_start+25> mov ebx,0x0
0x04901e <_start+30> mov ecx,0x049000
0x049023 <_start+35> mov edx,0x7
0x049028 <_start+40> int 0x0
0x04902d <_start+45> mov ebx,0x1
0x049032 <_start+50> mov edx,0x0
0x049037 <_start+55> int 0x0

```

Рис. 2.11: Режим псевдографики

Я установила точку останова по адресу инструкции.

```

Register group: general
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffced0 0xffffced0
ebp 0x0 0x0

0x04902a <_start+42> int 0x80
0x04902c <_start+44> mov eax,0x1
b+ 0x049031 <_start+49> mov ebx,0x0
0x049036 <_start+54> int 0x80
0x049038 add BYTE PTR [eax],al
0x04903a add BYTE PTR [eax],al

native process 5665 (asm) In: _start L9 PC: 0x049000
(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 *0x049031
Breakpoint 2 at 0x049031: file lab09-2.asm, line 20.
(gdb)

```

Рис. 2.12: Точка останова

Далее я посмотрела информацию о всех установленных точках останова.

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffced0 0xffffced0
ebp      0x0      0x0

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

native process 5665 (asm) In: _start          L9    PC: 0x8049000
(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. 2.13: Все точки останова

Я посмотрела содержимое регистров.

```

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
0x8049042      add    BYTE PTR [eax],al
0x8049044      add    BYTE PTR [eax],al
0x8049046      add    BYTE PTR [eax],al
0x8049048      add    BYTE PTR [eax],al

native process 5665 (asm) In: _start
--Type <RET> for more, q to quit, c to continue without paging--
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb)

```

Рис. 2.14: Регистры

Также я посмотрела значение переменной msg1 по имени

```

native process 5665 (asm) In: _start
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb)

```

Рис. 2.15: msg1

и значение переменной msg2 по адресу.

```

(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n"
00f>
(gdb)

```

Рис. 2.16: msg2

Далее я изменила первый символ переменной msg1

```

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

```

Рис. 2.17: Изменение значения для регистра

и символы во второй переменной msg2.

```

(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor d!\n\034"
(gdb)

```

Рис. 2.18: Замена символов

Я вывела значения регистра edx в различных форматах.

```

$2 = 0
(gdb) p/t $eax
$3 = 0
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)

```

Рис. 2.19: Различные форматы edx

С помощью команды set я изменила значение регистра ebx.

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)

```

Рис. 2.20: Изменение значения ebx

Команда выводит разные значения, так как в первом вносится значение 2, а во втором регистр равен двум.

Я скопировала файл lab8-2.asm, создала исполняемый файл и загрузила его в отладчик, указав аргументы.

```
palarionova@linux:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
palarionova@linux:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
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-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. 2.21: Аргументы

Затем я установила точку останова перед первой инструкцией.

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) r
Starting program: /home/palarionova/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)
```

Рис. 2.22: Breakpoint \_start

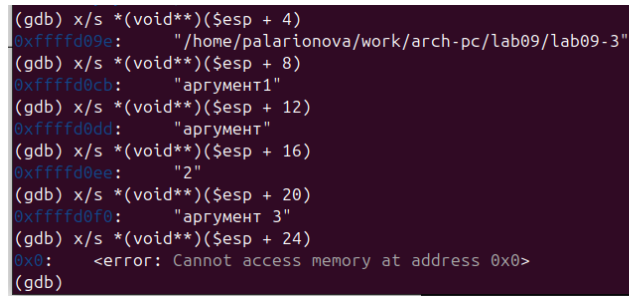
Проверила адрес вершины стека и убедилась, что там хранится 5 элементов.

```
(gdb) x/x $esp
0xffffcea0: 0x00000005
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n)
Please answer y or n.
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/palarionova/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)
```

Рис. 2.23: Вершина стека

Дальше я посмотрела остальные позиции стека.



```
(gdb) x/s *(void**)($esp + 4)
0xffffd09e:  "/home/palarionova/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd0cb:  "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd0dd:  "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd0ee:  "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd0f0:  "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb)
```

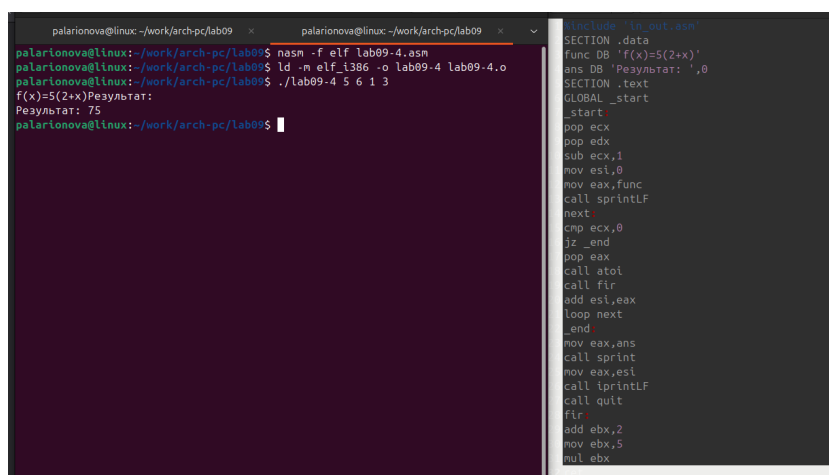
Рис. 2.24: Остальные позиции

По первому адресу хранится адрес, в остальных - элементы. Элементы расположены с интервалом 4 единицы, поэтому компьютер использует новый стек, так как они хранят информацию до 4 байт.



### 3 Задания для самостоятельной работы

- 1) Я преобразовала программу из лабораторной работы №8 и реализовала вычисление как подпрограмму. Создала исполняемый файл и проверила его работу.



```
palarionova@linux: ~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
palarionova@linux: ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
palarionova@linux: ~/work/arch-pc/lab09$ ./lab09-4 5 6 1 3
f(x)=5(2+x)Результат:
Результат: 75
palarionova@linux: ~/work/arch-pc/lab09$
```

```
include "in-out.asm"
SECTION .data
func DB 'f(x)=5(2+x)'
ans DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,func
call sprintf
next:
cmp ecx,0
jz _end
pop eax
call atoi
call fir
add esi,eax
loop next
_end:
mov eax,ans
call sprintf
mov eax,esi
call sprintf
call quit
fir:
add ebx,2
mov ebx,5
mul ebx
```

Рис. 3.1: Преобразованная программа

- 2) Я создала файл lab09-5.asm и ввела в него программу из листинга 9.3



```

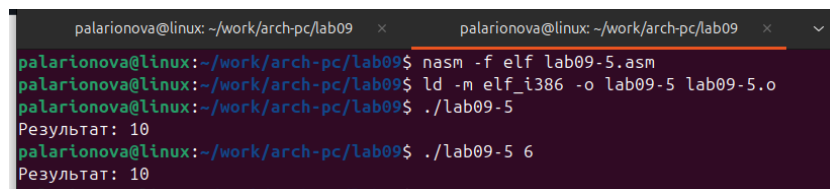
Открыть  ▾  [icon]  *lab09-5.asm
~work/arch-pc/lab09

%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 iprintLF
call quit

```

Рис. 3.2: Листинг 9.3

Проверив его работу, я убедилась, что результат неверный.



```

palarionova@linux: ~/work/arch-pc/lab09  ×  palarionova@linux: ~/work/arch-pc/lab09  ×  ▾
palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
palarionova@linux:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
palarionova@linux:~/work/arch-pc/lab09$ ./lab09-5 6
Результат: 10

```

Рис. 3.3: Исполняемый файл

Я открыла программу в отладчике.

```

palarionova@linux:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
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-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 lab09-5...
(gdb) r
Starting program: /home/palarionova/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
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 0xf7ffc000
Результат: 10
[Inferior 1 (process 5441) exited normally]
(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 <iprintf>
   0x08049111 <+41>:   call   0x80490db <quit>
End of assembler dump.

```

Рис. 3.4: Программа в отладчике

Затем открыла регистры и поняла, что некоторые из них стоят не на своих местах.

The screenshot shows a GDB terminal window with two panes. The top pane, titled 'Registers', displays '[ Register Values Unavailable ]'. The bottom pane shows assembly code with addresses and instructions. The terminal window has tabs for 'palarionova@linux: ~/work/arch-pc/lab09'.

```
Registers
[ Register Values Unavailable ]

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

exec No process (asm) In: L?? PC: ??
(gdb) layout regs
(gdb) b *0x80490e8
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 8.
(gdb)
```

Рис. 3.5: Ошибка в регистрах

Исправив ошибку, программа выдала верный результат.

```

palarionova@linux:~/work/arch-pc/lab09$ gedit lab09-5.asm
palarionova@linux:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.a
m
palarionova@linux:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
palarionova@linux:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
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-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 lab09-5...
(gdb) r
Starting program: /home/palarionova/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Результат: 25
[Inferior 1 (process 6053) exited normally]
(gdb) █

```

Рис. 3.6: Верный ответ

## 4 Выводы

При выполнении лабораторной работы я приобрела навыки написания программ использованием подпрограмм, а также познакомилась с методами отладки при помощи gbd и его основными возможностями.