

# Лабораторная 9

Мазуркевич Анастасия

# Содержание

Цель работы	5
Выполнение лабораторной работы	6
Порядок выполнения лабораторной работы . . . . .	6
Выводы	23
Список литературы	24

# Список иллюстраций

1	создаем каталог . . . . .	6
2	вводим . . . . .	7
3	создаем и запускаем . . . . .	7
4	заменяем строки . . . . .	8
5	проверяем работу файла . . . . .	9
6	создаем . . . . .	9
7	вводим код из листинга . . . . .	10
8	запускаем . . . . .	10
9	запускаем . . . . .	11
10	смотрим . . . . .	11
11	смотрим . . . . .	12
12	включаем . . . . .	13
13	проверяем . . . . .	13
14	проверяем . . . . .	13
15	устанавливаем . . . . .	14
16	выполняем . . . . .	15
17	регистры . . . . .	15
18	смотрим . . . . .	16
19	смотрим . . . . .	16
20	меняем . . . . .	16
21	смотрим . . . . .	16
22	меняем . . . . .	17
23	выходим . . . . .	17
24	копируем . . . . .	17
25	создаем . . . . .	18
26	устанавливаем . . . . .	18
27	смотрим . . . . .	18
28	копируем файл . . . . .	19
29	пишем . . . . .	19
30	проверяем . . . . .	19
31	создаем файл . . . . .	20
32	вводим листинг . . . . .	20
33	запускаем . . . . .	20
34	смотрим . . . . .	21
35	меняем . . . . .	21
36	проверяем . . . . .	22

## Список таблиц

## Цель работы

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

# Выполнение лабораторной работы

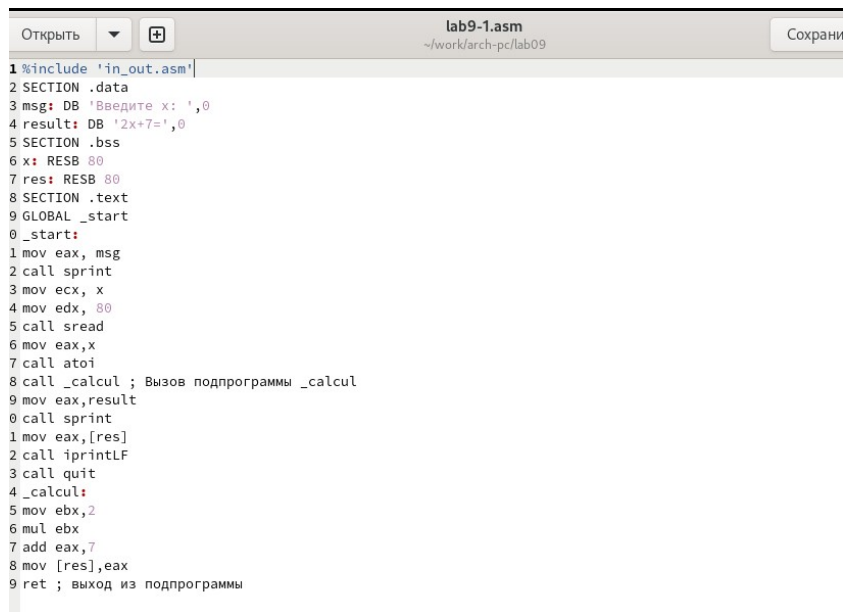
## Порядок выполнения лабораторной работы

Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте файл lab09-1.asm:

```
amazurkevich@vbox:~/work/arch-pc$ cd lab09
amazurkevich@vbox:~/work/arch-pc/lab09$ touch lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ls
lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$
```

Рис. 1: создаем каталог

Введите в файл lab9-1.asm 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 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. 2: вводим

Создайте исполняемый файл и запустите его.



```
amazurkevich@vbox:~/work/arch-pc/lab09$ gedit lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
amazurkevich@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
amazurkevich@vbox:~/work/arch-pc/lab09$
```

Рис. 3: создаем и запускаем

Измените текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения  $\varphi(\varphi(\varphi))$ , где  $\varphi$  вводится с клавиатуры,  $\varphi(\varphi) = 2\varphi + 7$ ,  $\varphi(\varphi) = 3\varphi - 1$ . Т.е.  $\varphi$  передается в подпрограмму `_calcul` из нее в подпрограмму `_subcalcul`, где вычисляется выражение  $\varphi(\varphi)$ , результат возвращается в `_calcul` и вычисляется выражение  $\varphi(\varphi(\varphi))$ . Результат возвращается в основную программу для вывода результата на экран.

```

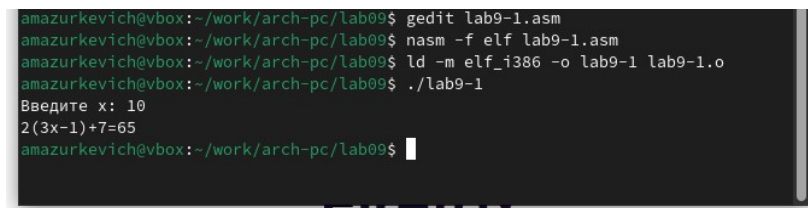
1 %include 'in_out.asm'
2 SECTION .data
3     msg: DB 'Введите x: ',0
4     result: DB '2(3x-1)+7=',0
5 SECTION .bss
6     x: RESB 80
7     res: RESB 80
8 SECTION .text
9 GLOBAL _start
10    _start:
11        mov eax, msg
12        call sprint
13        mov ecx, x
14        mov edx, 80
15        call 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        call _subcalcul
26        mov ebx,2
27        mul ebx
28        add eax,7
29        mov [res],eax
30        ret
31    _subcalcul:
32        mov ebx,3
33        mul ebx
34        sub eax,1
35        ret

```

Рис. 4: заменяем строки



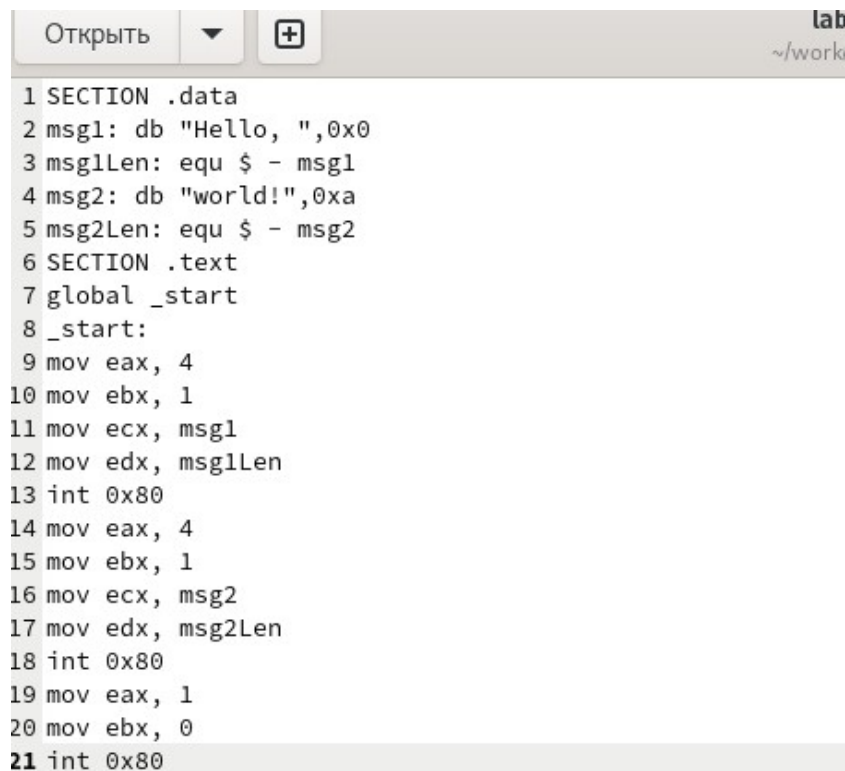
Создайте исполняемый файл и запустите его.



```
amazurkevich@vbox:~/work/arch-pc/lab09$ gedit lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
amazurkevich@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2(3x-1)+7=65
amazurkevich@vbox:~/work/arch-pc/lab09$
```

Рис. 5: проверяем работу файла

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!):



```
Открыть  [icon]  lab
~/work
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. 6: создаем

```

amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
amazurkevich@vbox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab9-2...
(gdb)

```

Рис. 7: вводим код из листинга

Получите исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. Загрузите исполняемый файл в отладчик gdb:

```

amazurkevich@vbox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab9-2...
(gdb) run
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4609) exited normally]
(gdb) run
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4613) exited normally]
(gdb)

```

Рис. 8: запускаем

Проверьте работу программы, запустив ее в оболочке GDB с помощью команды

run (сокращённо r):

```
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4609) exited normally]
(gdb) run
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4613) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 9: запускаем

Посмотрите дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`.

```
Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(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)
```

```
mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 10: смотрим

Переключитесь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`

```
End of assembler dump.
(gdb) set disassemble-flavor intel
No symbol "disassemble" in current context.
(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. 11: смотрим

Перечислите различия отображения синтаксиса машинных команд в режимах ATТ и Intel.

1. Порядок операндов
2. Синтаксис регистров
3. Синтаксис немедленных значений
4. Синтаксис адресов памяти
5. Команды для переходов и вызовов
6. Комментарии
7. Символы для указания на размер данных

Включите режим псевдографики для более удобного анализа программы (рис. 9.2):

```

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
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
0x804904a add    BYTE PTR [eax],al
native process 4624 In: _start L9 PC: 0x8049000

```

Рис. 12: включаем

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверьте это с помощью команды info breakpoints (кратко i b)

```

native process 4624 In: _start L9 P
(gdb) layuot regs
Undefined command: "layuot". Try "help".
(gdb) layout regs
(gdb) into breakpoints
Undefined command: "into". Try "help".
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb)

```

Рис. 13: проверяем

Проверим с помощью i b

```

      breakpoint already hit 1 time
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb)

```

Рис. 14: проверяем

Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции (см. рис. 9.3). Определите адрес предпоследней инструкции (`mov ebx,0x0`) и установите точку останова.

```

[ Register Values Unavailable ]

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

native process 4832 In: _start          L9    PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
2     breakpoint       keep y   0x8049031 lab9-2.asm:20
(gdb)

```

Рис. 15: устанавливаем

Выполните 5 инструкций с помощью команды `stepi` (или `si`) и проследите за изменением значений регистров.

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd080 0xffffd080
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35

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

native process 4832 In: _start L14 PC: 0x8049016
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num    Type             Disp Enb Address      What
1      breakpoint        keep y 0x08049000 lab9-2.asm:9
       breakpoint already hit 1 time
2      breakpoint        keep y 0x08049031 lab9-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 16: выполняем

Значения каких регистров изменяются? Во время выполнения команд менялись регистры: ebx, ecx, edx, eax, eip.

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd080 0xffffd080
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 17: регистры

Смотрим msg1

```

gs      0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb)

```

Рис. 18: смотрим

Смотрим msg1

```

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

```

Рис. 19: смотрим

Изменим первые символы

```

(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg2='L'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor!d!\n\034"
(gdb)

```

Рис. 20: меняем

Чтобы посмотреть значения регистров используется команда `print /F` (перед именем регистра обязательно ставится префикс `$`) (рис. 9.6):

```

(gdb) p/t $eax
$1 = 1000
(gdb) p/t $edx
$2 = 1000
(gdb) p/s $edx
$3 = 8
(gdb) p/x $edx
$4 = 0x8
(gdb) set $ebx='?'

```

Рис. 21: смотрим



изменяя ebx

Выводятся разные значения команда без кавычек присваивает регистру вводимое значение.

```
$3 = 8
(gdb) p/x $edx
$4 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)
```

Рис. 22: меняем

завершаем программу и выходим

```
(gdb) c
Continuing.
World!

Breakpoint 2, _start () at lab9-2.asm:20
(gdb)
```

Рис. 23: выходим

Скопируйте файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm:

```
amazurkevich@vbox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис. 24: копируем

Создайте исполняемый файл

```
amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
amazurkevich@vbox:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '4'
```

Рис. 25: создаем

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

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

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) b _start
Note: breakpoint 1 also set at pc 0x80490e8.
Breakpoint 2 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab09-3 2 3 4

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd070: 0x00000004
(gdb)
```

Рис. 26: устанавливаем

Посмотрите остальные позиции стека – по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д.

```
(gdb) x/x $esp
0xffffd070: 0x00000004
(gdb) x/s *(void**)(esp + 4)
0xffffd233: "/home/amazurkevich/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd261: "2"
(gdb) x/s *(void**)(esp + 12)
0xffffd263: "3"
(gdb) x/s *(void**)(esp + 16)
0xffffd265: "4"
(gdb) x/s *(void**)(esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 27: смотрим

Адресные регистры имеют размерность 32 бита, соответственно 4 байта

## Задание для самостоятельной работы

Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму

```
Quit anyway? (y or n) y
amazurkevich@vbox: ~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab09-4.asm
amazurkevich@vbox: ~/work/arch-pc/lab09$
```

Рис. 28: копируем файл

Вводим программу

```
~/work/arch-pc/lab09
1 %include 'in_out.asm'
2 SECTION .data
3     msg: DB 'Введите x: ',0
4     result: DB '3(10+x)=',0
5 SECTION .bss
6     x: RESB 80
7     res: RESB 90
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     add eax, 10
26     mov ebx, 3
27     mul ebx
28     mov [res], eax
29     ret
```

Рис. 29: пишем

Запускаем

```
amazurkevich@vbox: ~/work/arch-pc/lab09$ ./lab9-4
Введите x: 10
3(10+x)=60
amazurkevich@vbox: ~/work/arch-pc/lab09$
```

Рис. 30: проверяем

В листинге 9.3 приведена программа вычисления выражения  $(3 + 2) \cdot 4 + 5$ . При

запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

```
amazurkevich@vbox:~/work/arch-pc/lab09$ touch lab9-5.asm
amazurkevich@vbox:~/work/arch-pc/lab09$
```

Рис. 31: создаем файл

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB '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. 32: вводим листинг

Создаем файл и запускаем

```
amazurkevich@vbox:~/work/arch-pc/lab09$ gedit lab9-5.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
amazurkevich@vbox:~/work/arch-pc/lab09$ ./lab9-5
Результат: 10
amazurkevich@vbox:~/work/arch-pc/lab09$
```

Рис. 33: запускаем

Запускаем его в отладчике GDB и смотрим на изменение регистров командой si

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd080 gister Val0xffffd080lable ]
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490e8  0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35

3> 0x80490e8 <_start> mov     $0x3,%eax
0x80490ed <_start+5> mov     $0x2,%ebx
0x80490f2 <_start+10> add     %ebx,%eax
0x80490f4 <_start+12> mov     $0x4,%ecx
0x80490f9 <_start+17> mul     %ecx,ly Available ]
0x80490fb <_start+19> add     $0x5,%eax
0x80490fe <_start+22> mov     %eax,%edi
0x8049100 <_start+24> mov     $0x040800,%eax
0x8049105 <_start+29> call    0x804908f <sprint>
0x804910a <_start+34> mov     %edi,%eax
0x804910c <_start+36> call    0x804908e <iprintf>

xec No process In: L?? PC: ??
active process 5554 In: _start L?? PC: 0x80490e8
<https://debuginfod.fedoraproject.org/>
enable debuginfod for this session? (y or [n]) n
debuginfod has been disabled.
to make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
езультат: 25
Inferior 1 (process 5546) exited normally]
(gdb) break _start
Breakpoint 2 at 0x80490e8
(gdb) run
Starting program: /home/amazurkevich/work/arch-pc/lab09/lab9-5

Breakpoint 2, 0x80490e8 in _start ()
(gdb)

```

Рис. 34: смотрим

Изменяем программу

```

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

```

Рис. 35: меняем

Создаем исполняемый файл и запускаем его

```
amazurkevich@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
amazurkevich@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
amazurkevich@vbox:~/work/arch-pc/lab09$ ./lab9-5
Результат: 25
```

Рис. 36: проверяем

## Выводы

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

## Список литературы