

# **Отчет по лабораторной работе №10**

**Понятие подпрограммы. Отладчик GDB**

Мокочунина Влада Сергеевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	17

# Список иллюстраций

3.1	Создание файла . . . . .	7
3.2	Ввод текста . . . . .	7
3.3	Запуск программы . . . . .	8
3.4	Замена текста . . . . .	8
3.5	Запуск программы . . . . .	8
3.6	Ввод текста . . . . .	9
3.7	Загрузка файла и проверка работы . . . . .	9
3.8	Установка брейкпоинта . . . . .	10
3.9	Код программы . . . . .	10
3.10	Другой синтаксис . . . . .	10
3.11	Режим псевдографики . . . . .	11
3.12	Точка останова . . . . .	11
3.13	Просмотр регистров . . . . .	12
3.14	Просмотр значения переменной . . . . .	12
3.15	Просмотр значения переменной . . . . .	12
3.16	Замена первого символа . . . . .	12
3.17	Замена символа . . . . .	13
3.18	Значения регистров . . . . .	13
3.19	Замена значения регистра . . . . .	14
3.20	Выход . . . . .	14
3.21	Копия файла . . . . .	15
3.22	Загрузка файла . . . . .	15
3.23	Точка останова . . . . .	16
3.24	Адрес вершины стека . . . . .	16
3.25	Позиции стека . . . . .	16
4.1	Преобразование программы . . . . .	17
4.2	Запуск файла . . . . .	17
4.3	Ввод текста . . . . .	18
4.4	Запуск файла . . . . .	18
4.5	Запуск файла . . . . .	19
4.6	Запуск файла . . . . .	19
4.7	Исправление ошибок . . . . .	20
4.8	Запуск файла . . . . .	20

## List of Tables

# 1 Цель работы

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

## 2 Задание

Приобрести навыки написания программ с использованием подпрограмм, познакомиться с методами отладки при помощи GDB и его основными возможностями.

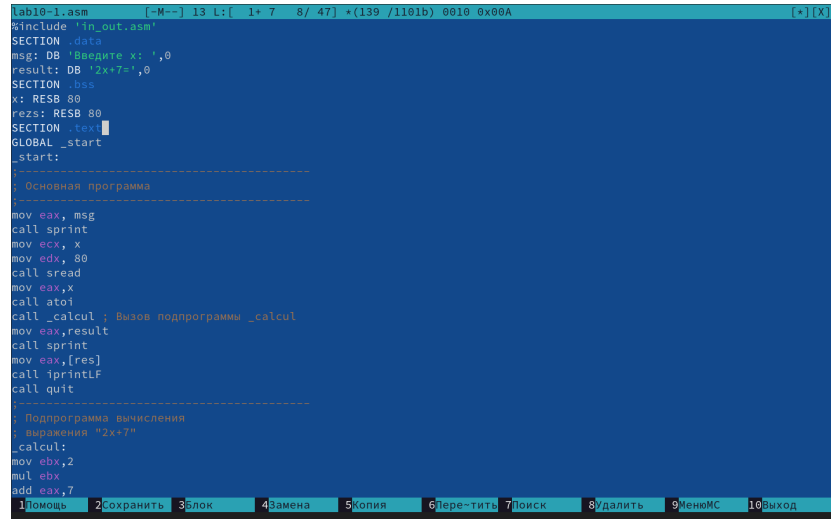
### 3 Выполнение лабораторной работы

1. Я создала каталог для лабораторной работы №10 и создала файл lab10-1.asm (рис. 3.1)

```
[vsmokochunina@10 lab10]$ touch lab10-1.asm  
[vsmokochunina@10 lab10]$
```

Рис. 3.1: Создание файла

2. Я ввела в него текст из листинга 1



```
lab10-1.asm  [-M--] 13 L: [ 1+ 7 8/ 47] +(139 /1101b) 0010 0x00A  [*] [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 iprintf  
call quit  
-----  
; Подпрограмма вычисления  
; выражения "2x+7"  
_calcul:  
mov ebx, 2  
mul ebx  
add eax, 7  
-----  
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Пере-тить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 3.2: Ввод текста

3. Я создала файл и проверила его работу

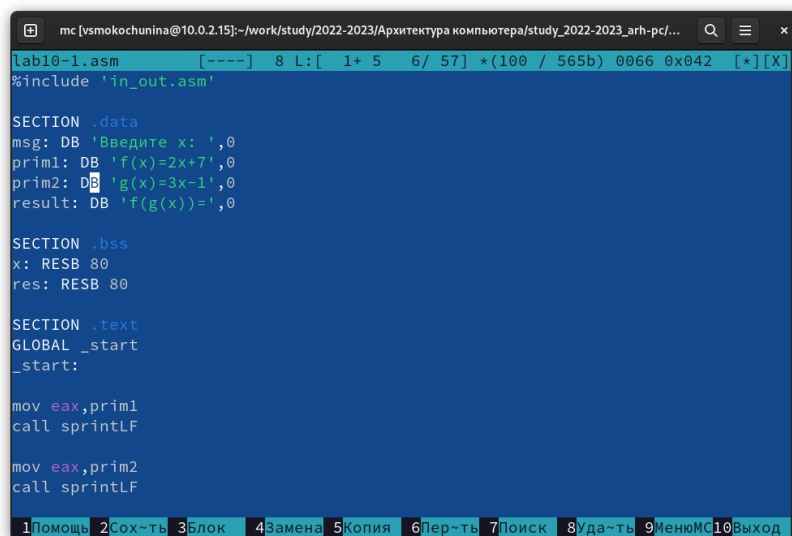
```

[vsmokochunina@10 lab10]$ nasm -f elf lab10-1.asm
[vsmokochunina@10 lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[vsmokochunina@10 lab10]$ ./lab10-1
Введите x: 7
2x+7=21

```

Рис. 3.3: Запуск программы

4. Я изменила текст программы для вычисления заданного выражения



```

lab10-1.asm  [----]  8 L: [ 1+ 5 6/ 57] *(100 / 565b) 0066 0x042 [*] [X]
#include 'in_out.asm'

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

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,prim1
call sprintLF

mov eax,prim2
call sprintLF

```

Рис. 3.4: Замена текста

5. Я создала и запустила файл

```

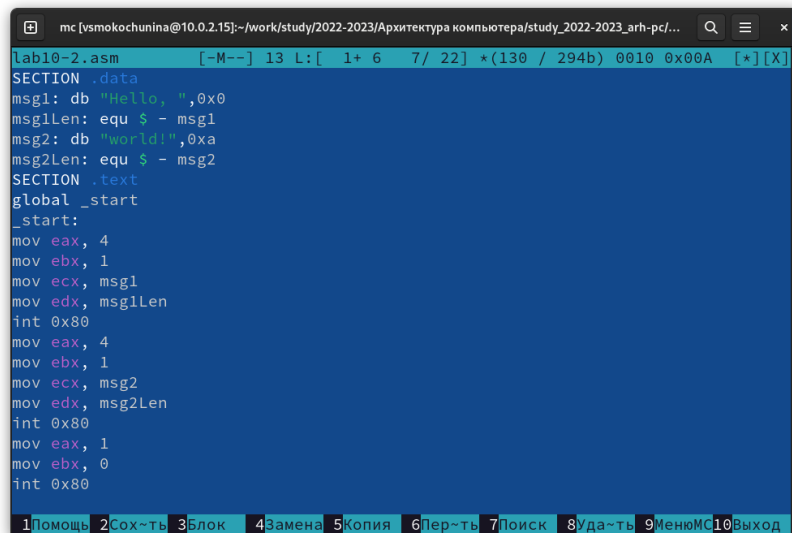
[vsmokochunina@10 lab10]$ nasm -f elf lab10-1.asm
[vsmokochunina@10 lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[vsmokochunina@10 lab10]$ ./lab10-1
f(x)=2x+7
g(x)=3x-1
Введите x: 2
f(g(x))=17

```

Рис. 3.5: Запуск программы

6. Я создала файл lab10-2.asm и ввела в него текст из листинга 2

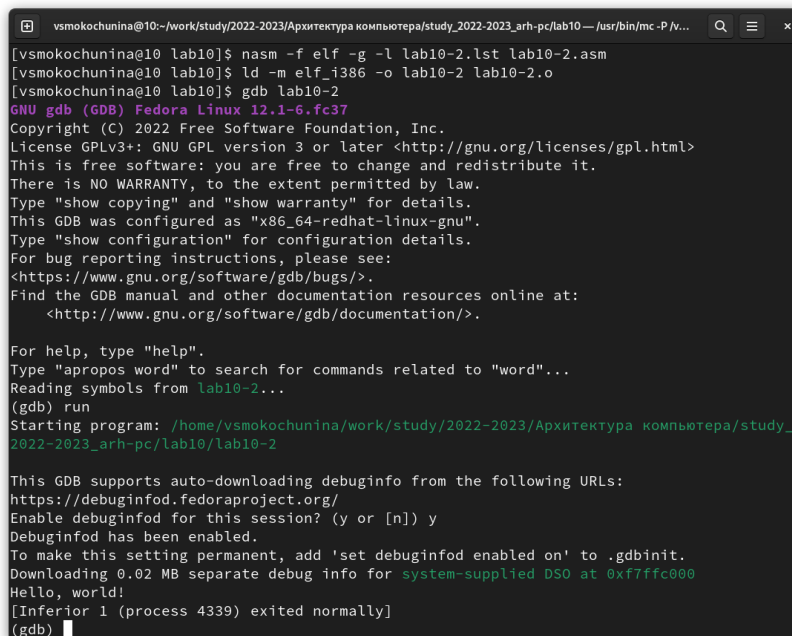




```
lab10-2.asm [-M--] 13 L: [ 1+ 6 7/ 22] *(130 / 294b) 0010 0x00A [*][X]
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 3.6: Ввод текста

7. Я получила исполняемый файл, загрузила его в отладчик gdb и проверила работу программы, запустив ее в оболочке GDB



```
vsmokochunina@10:~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 --- /usr/bin/mc -P /v...
[vsmokochunina@10 lab10]$ nasm -f elf -g -l lab10-2.lst lab10-2.asm
[vsmokochunina@10 lab10]$ ld -m elf_i386 -o lab10-2 lab10-2.o
[vsmokochunina@10 lab10]$ gdb lab10-2
GNU gdb (GDB) Fedora Linux 12.1-6.fc37
Copyright (C) 2022 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 lab10-2...
(gdb) run
Starting program: /home/vsmokochunina/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10/lab10-2

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.
Downloading 0.02 MB separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4339) exited normally]
(gdb)
```

Рис. 3.7: Загрузка файла и проверка работы

8. Я установила брейкпоинт на метку \_start

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 9.
(gdb) run
Starting program: /home/vsmokochunina/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10/lab10-2
Breakpoint 1, _start () at lab10-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 3.8: Установка брейкпоинта

## 9. Я посмотрела дисассимилированный код программы

```
Breakpoint 1, _start () at lab10-2.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.
```

Рис. 3.9: Код программы

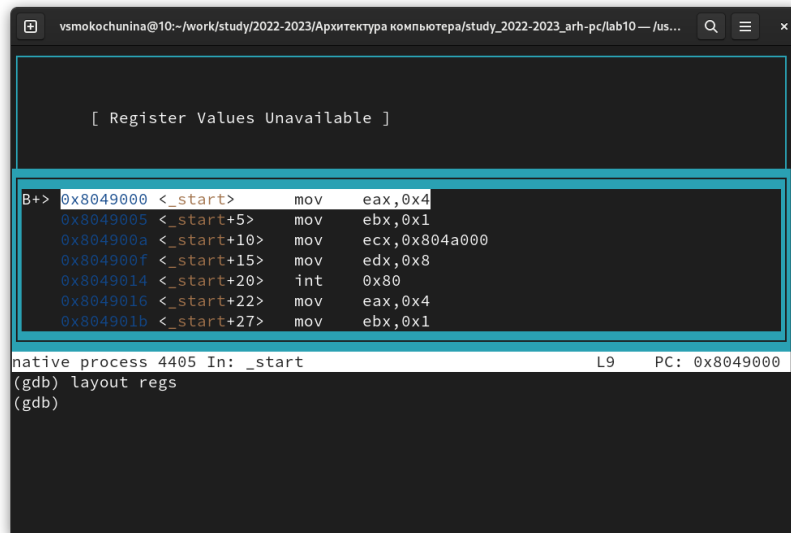
## 10. Я переключилась на отображение команд с Intel'овским синтаксисом

```
End of assembler dump.
(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. 3.10: Другой синтаксис

Различие только в отсутствии символов % и \$

## 11. Я включила режим псевдографики



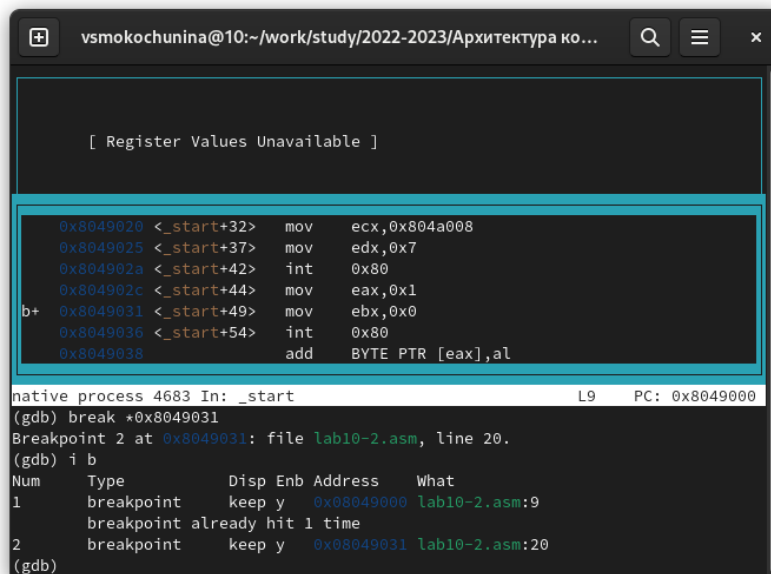
```
vsmokochunina@10:~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 — /us...
[ Register Values Unavailable ]

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

native process 4405 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 3.11: Режим псевдографики

## 12. Я проверила точку останова и установила свою



```
vsmokochunina@10:~/work/study/2022-2023/Архитектура ко...
[ Register Values Unavailable ]

      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

native process 4683 In: _start L9 PC: 0x8049000
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 20.
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint      keep y   0x08049000 lab10-2.asm:9
      breakpoint already hit 1 time
2     breakpoint      keep y   0x08049031 lab10-2.asm:20
(gdb)
```

Рис. 3.12: Точка останова

## 13. Я посмотрела регистры

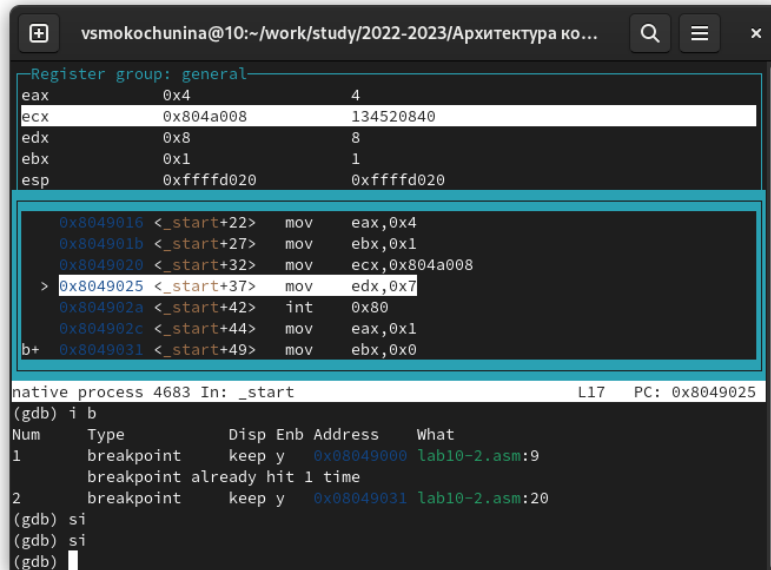


Рис. 3.13: Просмотр регистров

14. Я посмотрела значение переменной msg1

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

Рис. 3.14: Просмотр значения переменной

15. Я посмотрела значение переменной msg2

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

Рис. 3.15: Просмотр значения переменной

16. Я изменила первый символ переменной msg1

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hhello, "
```

Рис. 3.16: Замена первого символа

17. Я заменила символ переменной msg2

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

Рис. 3.17: Замена символа

18. Я вывела значения регистров

```
(gdb) p/s $eax  
$2 = 4  
(gdb) p/t $eax  
$3 = 100  
(gdb) p/c $ecx  
$4 = 8 '\b'  
(gdb) p/x $ecx  
$5 = 0x804a008  
(gdb)
```

Рис. 3.18: Значения регистров

19. Я изменила значение регистра ebx

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$1 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$2 = 2  
(gdb) 
```

Рис. 3.19: Замена значения регистра

Выводится два разных значения, так как в первом случае мы вносим двойку, а во втором сам регистр равен двум

20. Я завершила выполнение программы и вышла

```
(gdb) quit  
A debugging session is active.  
  
Inferior 1 [process 3193] will be killed.  
Quit anyway? (y or n) 
```

Рис. 3.20: Выход

21. Я скопировала файл из лабораторной №9 в лабораторную №10

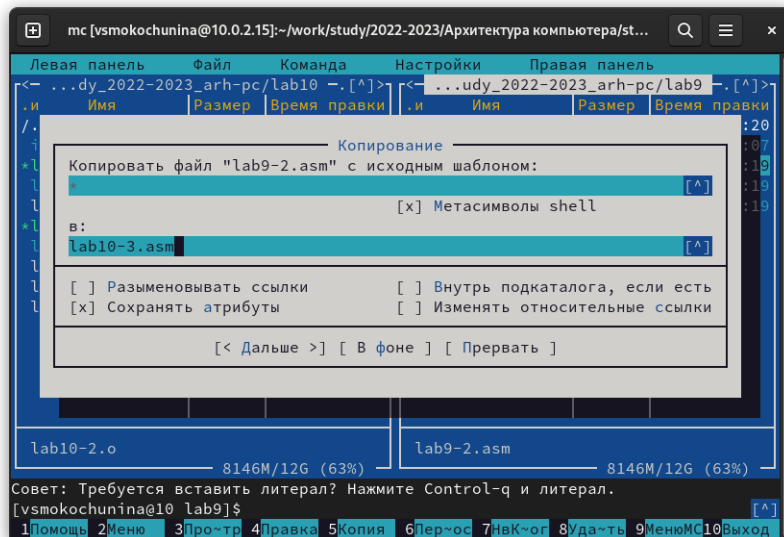


Рис. 3.21: Копия файла

22. Я создала файл и загрузила,указав аргументы

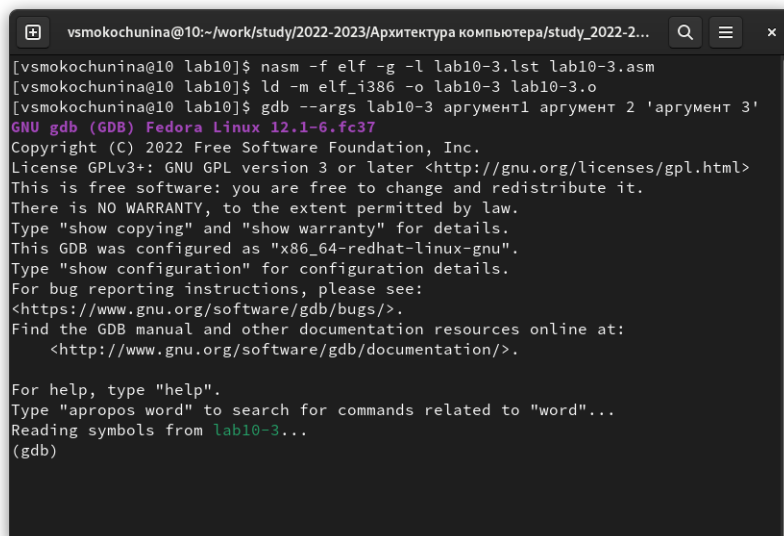


Рис. 3.22: Загрузка файла

23. Я установила точку останова

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) run
Starting program: /home/vsmokochunina/work/study/2022-2023/Архитектура компьютера
a/study_2022-2023_arh-pc/lab10/lab10-3 аргумент1 аргумент 2 аргумент\ 3

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.

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)

```

Рис. 3.23: Точка останова

24. Я посмотрела адрес вершины стека

```

(gdb) x/x $esp
0xffffcfd0:      0x00000005
(gdb)

```

Рис. 3.24: Адрес вершины стека

25. Я посмотрела остальные позиции стека

```

(gdb) x/s *(void**)(esp + 4)
0xffffd187:      "/home/vsmokochunina/work/study/2022-2023/Архитектура компьютера
/study_2022-2023_arh-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd201:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd213:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd224:      "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd226:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb)

```

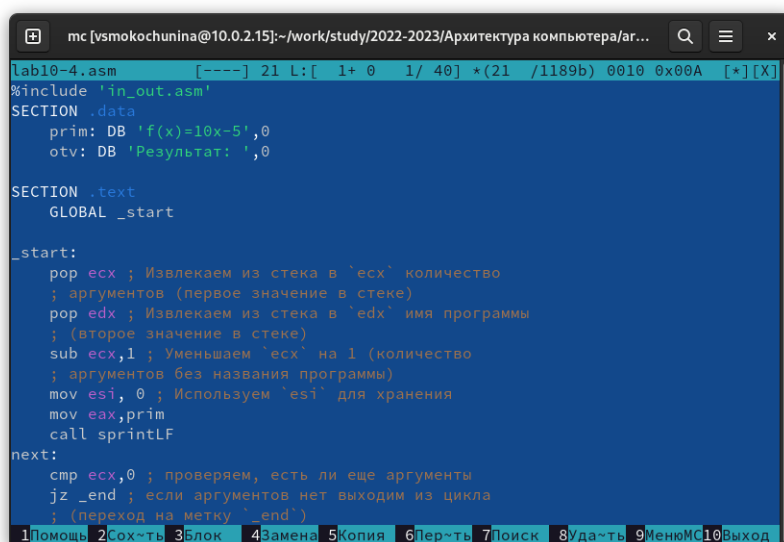
Рис. 3.25: Позиции стека

Интервал между элементами 4, так как в стеке хранится только до 4 байт



## 4 Самостоятельная работа

1. Я преобразовала программу из лабораторной работы No9, реализовав вычисление значения функции  $f(x) = 10x - 5$  как подпрограмму.



```
lab10-4.asm [----] 21 L: [ 1+ 0 1/ 40] *(21 /1189b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
    prim: DB 'f(x)=10x-5',0
    otv: DB 'Результат: ',0
SECTION .text
    GLOBAL _start
_start:
    pop ecx ; Извлекаем из стека в 'ecx' количество
             ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в 'edx' имя программы
             ; (второе значение в стеке)
    sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
             ; аргументов без названия программы)
    mov esi, 0 ; Используем 'esi' для хранения
    mov eax,prim
    call sprintf
next:
    cmp ecx,0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку '_end')
```

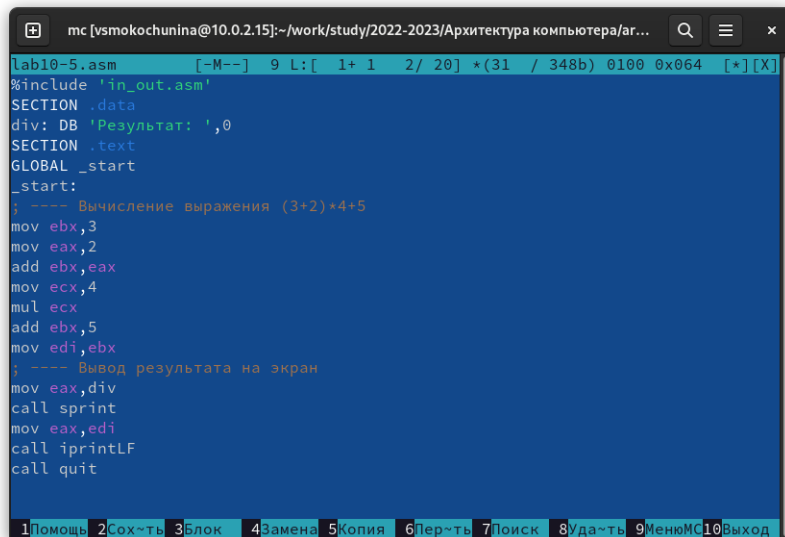
Рис. 4.1: Преобразование программы

2. Я создала и запустила файл

```
[vsmokochunina@10 lab10]$ nasm -f elf lab10-4.asm
lab10-4.asm:18: error: symbol 'sprintf' not defined
[vsmokochunina@10 lab10]$ nasm -f elf lab10-4.asm
[vsmokochunina@10 lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o
[vsmokochunina@10 lab10]$ ./lab10-4 1 2 3 4
f(x)=10x-5
Результат: 80
```

Рис. 4.2: Запуск файла

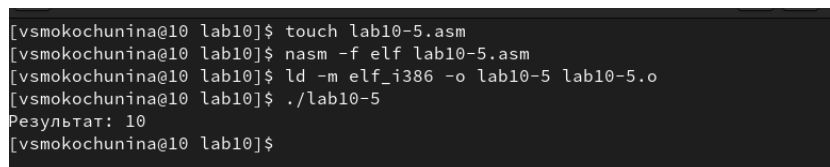
3. Я создала файл и ввела из него программу из листинга 3



```
lab10-5.asm [-M--] 9 L: [ 1+ 1 2/ 20] *(31 / 348b) 0100 0x064 [*][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 sprintf
mov eax,edi
call iprintLF
call quit
```

Рис. 4.3: Ввод текста

4. Я запустила файл и увидела, что выдает ошибку



```
[vsmokochunina@i0 lab10]$ touch lab10-5.asm
[vsmokochunina@i0 lab10]$ nasm -f elf lab10-5.asm
[vsmokochunina@i0 lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[vsmokochunina@i0 lab10]$ ./lab10-5
Результат: 10
[vsmokochunina@i0 lab10]$
```

Рис. 4.4: Запуск файла

5. Я запустила файл с помощью отладчика gdb

```
vsmokochunina@10:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab10 — /usr/bin/mc -P /var/tmp/mc-vsmoko...
[vsmokochunina@10 lab10]$ nasm -f elf -g -l lab10-5.lst lab10-5.asm
[vsmokochunina@10 lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[vsmokochunina@10 lab10]$ gdb lab10-5
GNU gdb (GDB) Fedora Linux 12.1-6.fc37
Copyright (C) 2022 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 lab10-5...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 8.
(gdb) run
Starting program: /home/vsmokochunina/work/study/2022-2023/Архитектура компьютера/arch-pc/lab10/lab10-5

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.
```

Рис. 4.5: Запуск файла

```
vsmokochunina@10:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab10 — /usr/bin/mc -P /var/tmp/mc-vsmoko...
Starting program: /home/vsmokochunina/work/study/2022-2023/Архитектура компьютера/arch-pc/lab10/lab10-5

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.

Breakpoint 1, _start () at lab10-5.asm:8
8      mov ebx,3
(gdb) set disassembly-flavor intel
No symbol "disassembly" in current context.
(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
    0x08049108 <+24>: mov     eax,0x804a000
    0x08049105 <+29>: call    0x080490ef <sprint>
    0x0804910a <+34>: mov     eax,edi
    0x0804910c <+36>: call    0x08049085 <printfLF>
    0x08049111 <+41>: call    0x080490db <quit>
End of assembler dump.
(gdb)
```

Рис. 4.6: Запуск файла

6. Я посмотрела, что регистры стоят не на своих местах и изменила это

Register group: general			
eax	0x0	0	
ecx	0x0	0	
edx	0x0	0	
ebx	0x3	3	
esp	0xffffd050	0xffffd050	
ebp	0x0	0x0	
esi	0x0	0	

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

Рис. 4.7: Исправление ошибок

## 7. Я проверила работу программы

```
[vsmokochunina@i0 lab10]$ nasm -f elf lab10-5.asm
[vsmokochunina@i0 lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[vsmokochunina@i0 lab10]$ ./lab10-5
Результат: 25
[vsmokochunina@i0 lab10]$
```

Рис. 4.8: Запуск файла

## #Вывод

Я приобрела навыки написания программ с использованием подпрограмм, познакомилась с методами отладки при помощи GDB и его основными возможностями.