

Отчёт по лабораторной работе 9

Понятие подпрограммы. Отладчик GDB.

Тукаев Тимур Ильшатovich НММбд-03-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	35

Список иллюстраций

2.1	Программа в файле lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	9
2.3	Программа в файле lab9-1.asm	10
2.4	Запуск программы lab9-1.asm	12
2.5	Программа в файле lab9-2.asm	13
2.6	Запуск программы lab9-2.asm в отладчике	15
2.7	Дизассимилированный код	16
2.8	Дизассимилированный код в режиме интел	17
2.9	Точка остановки	18
2.10	Изменение регистров	19
2.11	Изменение регистров	20
2.12	Изменение значения переменной	21
2.13	Вывод значения регистра	22
2.14	Вывод значения регистра	23
2.15	Вывод значения регистра	25
2.16	Программа в файле lab9-4.asm	26
2.17	Запуск программы lab9-4.asm	28
2.18	Код с ошибкой	30
2.19	Отладка	31
2.20	Код исправлен	33
2.21	Проверка работы	34

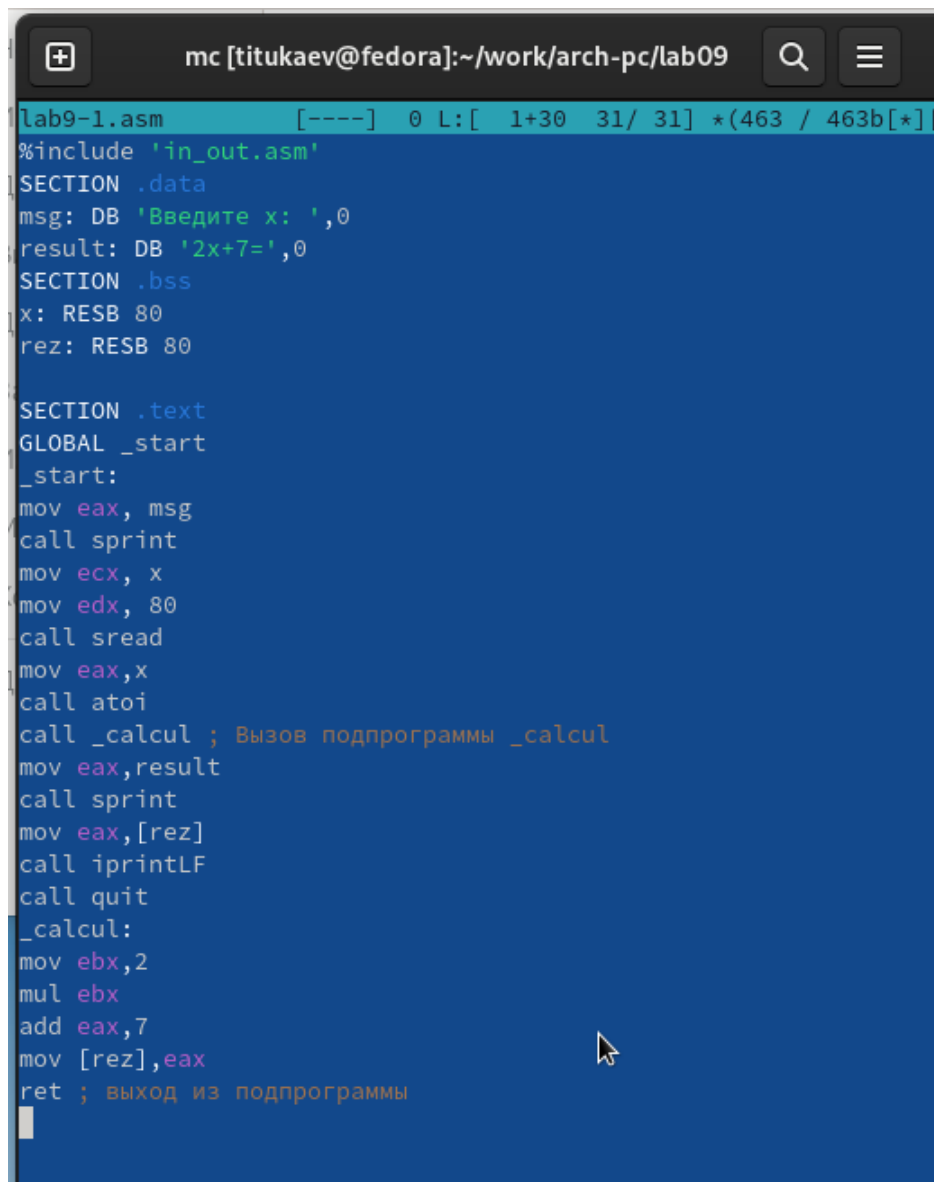
Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



```
lab9-1.asm [----] 0 L:[ 1+30 31/ 31] *(463 / 463b[*]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Рис. 2.1: Программа в файле lab9-1.asm

Также размещаю код программы в отчете.

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
```

```

x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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,[rez]
call iprintLF
call quit
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [rez],eax
ret ; выход из подпрограммы

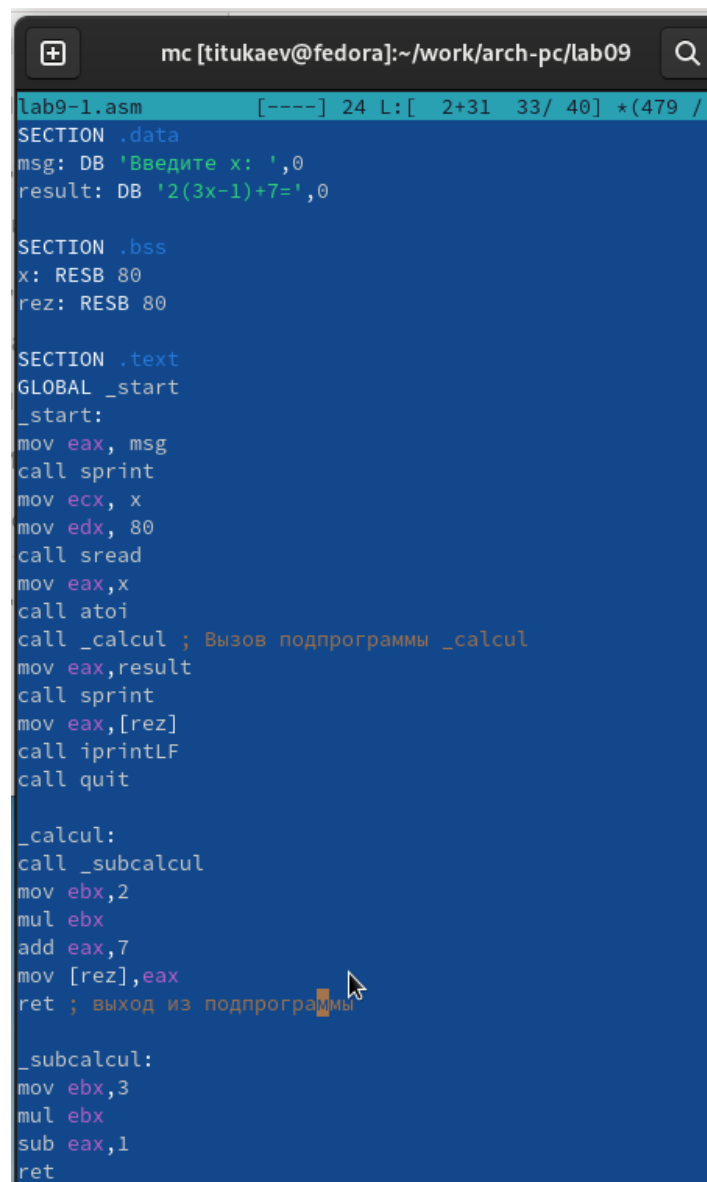
```



```
[titukaev@fedora lab09]$  
[titukaev@fedora lab09]$ nasm -f elf lab9-1.asm  
[titukaev@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[titukaev@fedora lab09]$ ./lab9-1  
Введите x: 6  
2x+7=19  
[titukaev@fedora lab09]$
```

Рис. 2.2: Запуск программы lab9-1.asm

3. Изменил текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$.



```
lab9-1.asm [----] 24 L:[ 2+31 33/ 40] *(479 /
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 2.3: Программа в файле lab9-1.asm

Код программы

```
%include 'in_out.asm'

SECTION .data

msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0
```

```

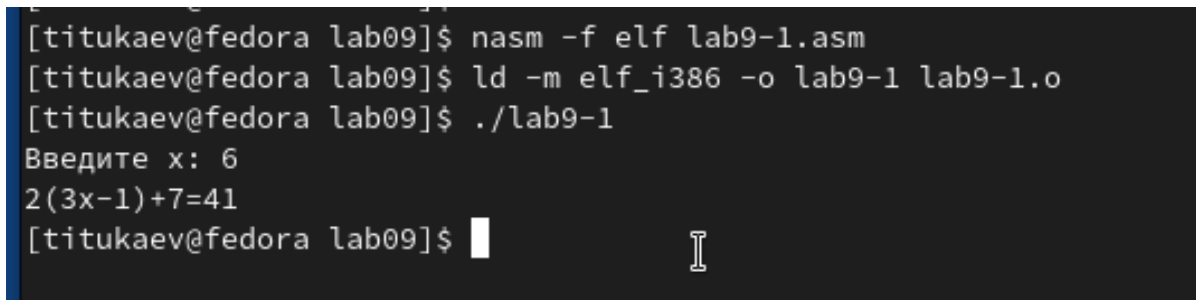
SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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,[rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [rez],eax
ret ; выход из подпрограммы

```

```
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret
```



```
[titukaev@fedora lab09]$ nasm -f elf lab9-1.asm
[titukaev@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[titukaev@fedora lab09]$ ./lab9-1
Введите x: 6
2(3x-1)+7=41
[titukaev@fedora lab09]$
```

Рис. 2.4: Запуск программы lab9-1.asm

4. Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).



```
mc [titukaev@fedora]:~/work/a
lab9-2.asm [----] 10 L: [ 1+15
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. 2.5: Программа в файле lab9-2.asm

Также размещаю код программы в отчете.

```
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
```

Получил исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'.

Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).

```

[titukaev@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[titukaev@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[titukaev@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora 12.1-2.fc36
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 lab9-2...
(gdb) r
Starting program: /home/titukaev/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])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 3550) exited normally]
(gdb)

```

Рис. 2.6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы.

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/titukaev/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      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)

```

Рис. 2.7: Дизассимилированный код


```

(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. 2.8: Дизассимилированный код в режиме интел

На предыдущих шагах была установлена точка остановки по имени метки (`_start`). Проверил это с помощью команды `info breakpoints` (кратко `i b`). Установил еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установил точку.

```
(gdb) ib
Undefined command: "ib". Try "help".
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:11
          breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:22
(gdb) 
```

Рис. 2.9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды `stepi` (или `si`) и проследил за изменением значений регистров.

```
titukaev@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd220 0xffffd220
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
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 3561 In: _start L12 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--cs 0
x23      35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) █
```

Рис. 2.10: Изменение регистров

```
titukaev@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd220 0xffffd220
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 3561 In: _start L16 PC: 0x8049016
gs      0x0      0
(gdb) si
(gdb) ib
Undefined command: "ib". Try "help".
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:11
         breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:22
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.11: Изменение регистров

Посмотрел значение переменной msg1 по имени. Посмотрел значение переменной msg2 по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменил первый символ переменной msg1.

```

native process 3561 In: _start
(gdb) si
(gdb) si
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "Lor!d!\n\034"
(gdb)

```

Рис. 2.12: Изменение значения переменной

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра `edx`.

```
native process 3561 In: _start
```

```
$1 = 8
```

```
(gdb) p/t $eax
```

```
$2 = 1000
```

```
(gdb) p/s $ecx
```

```
$3 = 134520832
```

```
(gdb) p/x $ecx
```

```
$4 = 0x804a000
```

```
(gdb) p/s $edx
```

```
$5 = 8
```

```
(gdb) p/t $edx
```

```
$6 = 1000
```

```
(gdb) p/x $edx
```

```
$7 = 0x8
```

```
(gdb) 
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

```
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb) 
```

Рис. 2.14: Вывод значения регистра

5. Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы.

Код программы

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

```

next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call printf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit

```

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

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы `lab9-3` и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрел 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.


```
titukaev@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 ar...
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/titukaev/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

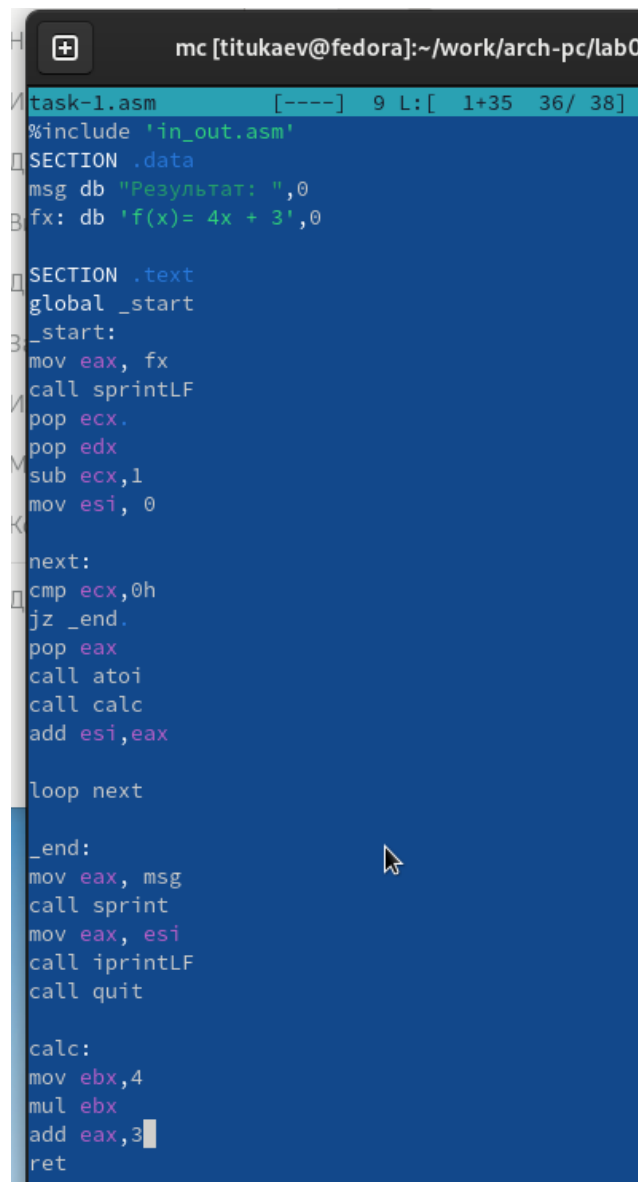
This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1f0:      0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd3ad:      "/home/titukaev/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3d6:      "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3df:      "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3e1:      "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3ea:      "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3ec:      "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] - шаг равен размеру переменной - 4 байтам.

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



```
mc [titukaev@fedora]:~/work/arch-pc/lab0
task-1.asm [----] 9 L: [ 1+35 36/ 38]
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 4x + 3',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

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

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

calc:
mov ebx,4
mul ebx
add eax,3
ret
```

Рис. 2.16: Программа в файле lab9-4.asm

Также размещаю код программы в отчете.

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 4x + 3',0
```

```

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

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

loop next

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

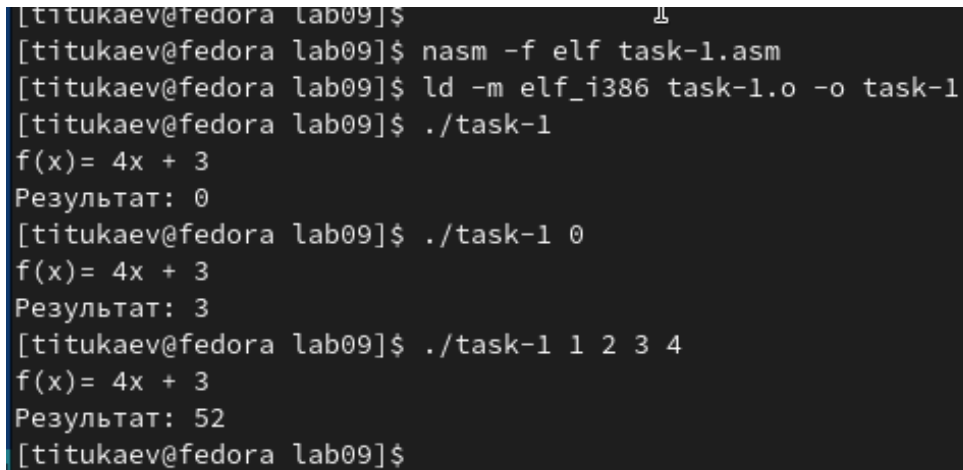
calc:
mov ebx,4

```

```

mul ebx
add eax,3
ret

```



```

[titukaev@fedora lab09]$
[titukaev@fedora lab09]$ nasm -f elf task-1.asm
[titukaev@fedora lab09]$ ld -m elf_i386 task-1.o -o task-1
[titukaev@fedora lab09]$ ./task-1
f(x)= 4x + 3
Результат: 0
[titukaev@fedora lab09]$ ./task-1 0
f(x)= 4x + 3
Результат: 3
[titukaev@fedora lab09]$ ./task-1 1 2 3 4
f(x)= 4x + 3
Результат: 52
[titukaev@fedora lab09]$

```

Рис. 2.17: Запуск программы lab9-4.asm

- В листинге приведена программа вычисления выражения $(3 + 2) * 4 + 5$. При запуске данная программа дает неверный результат. Проверил это. С помощью отладчика GDB, анализируя изменения значений регистров, определяю ошибку и исправляю ее.

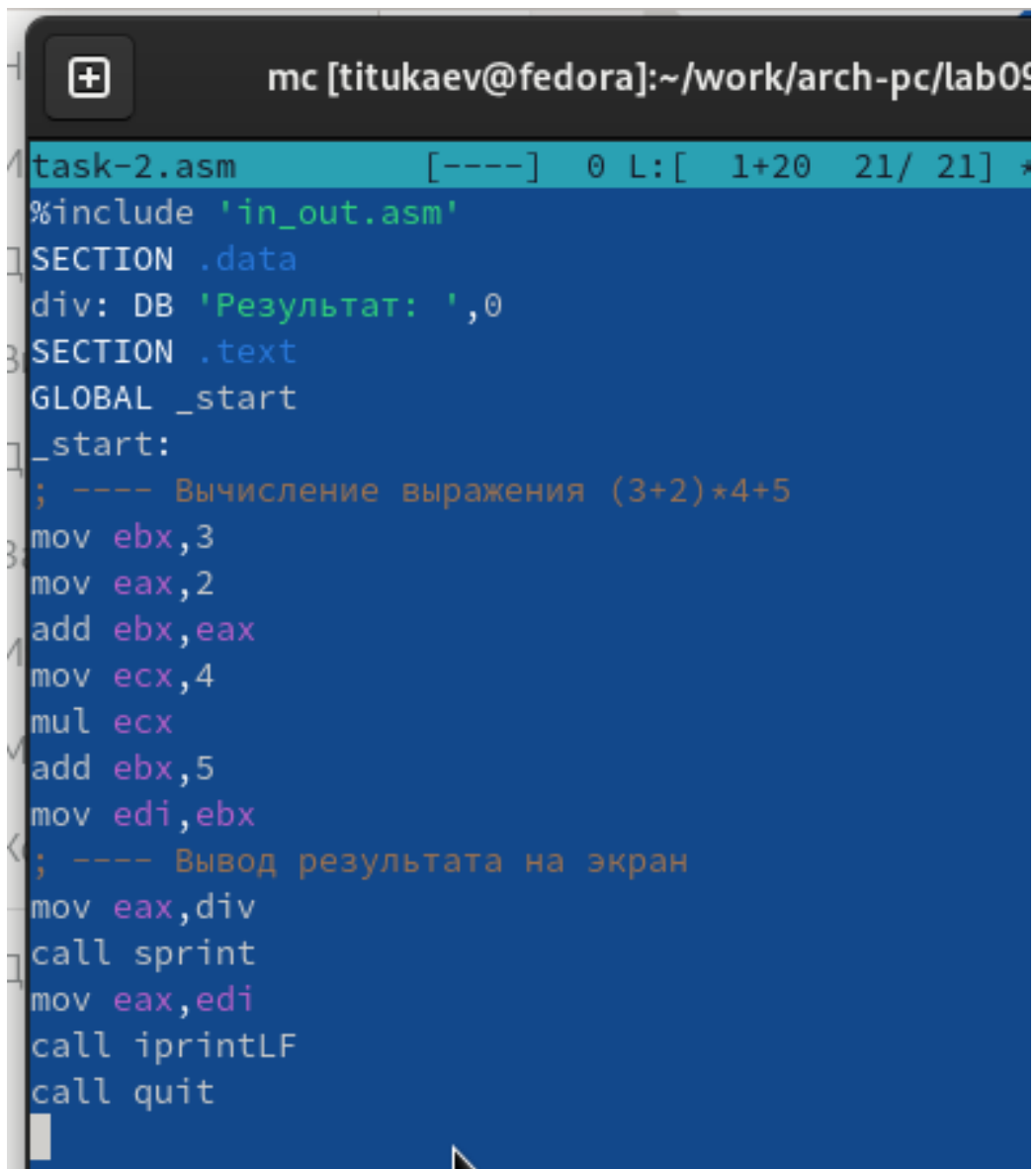
Также размещаю код программы в отчете.

```

%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
```



```
mc [titukaev@fedora]:~/work/arch-pc/lab09
task-2.asm [----] 0 L:[ 1+20 21/ 21] *
#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. 2.18: Код с ошибкой

```

titukaev@fedora:~/work/arch-pc/lab09 — gdb task-2

eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd220 0xffffd220
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>
eflags   0x206    [ PF IF ]
cs       0x23     35

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x8049100 <_start+24>    mov     eax,0x804a000
0x8049105 <_start+29>    call   0x804900f <sprint>
0x804910a <_start+34>    mov     eax,edi
0x804910c <_start+36>    call   0x8049086 <iprintLF>
0x8049111 <_start+41>    call   0x80490db <quit>

>                                04a000
                                rint>

                                86 <iprintLF>

native process 3833 In: _start L16 PC: 0x8049100
To makeNo process In: , add 'set debuginfod enabled off' to .gdbinit. L?? PC: ??
Breakpoint 1, _start () at task-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 10
[Inferior 1 (process 3833) exited normally]
(gdb)

```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

Исправленный код программы

```

#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text

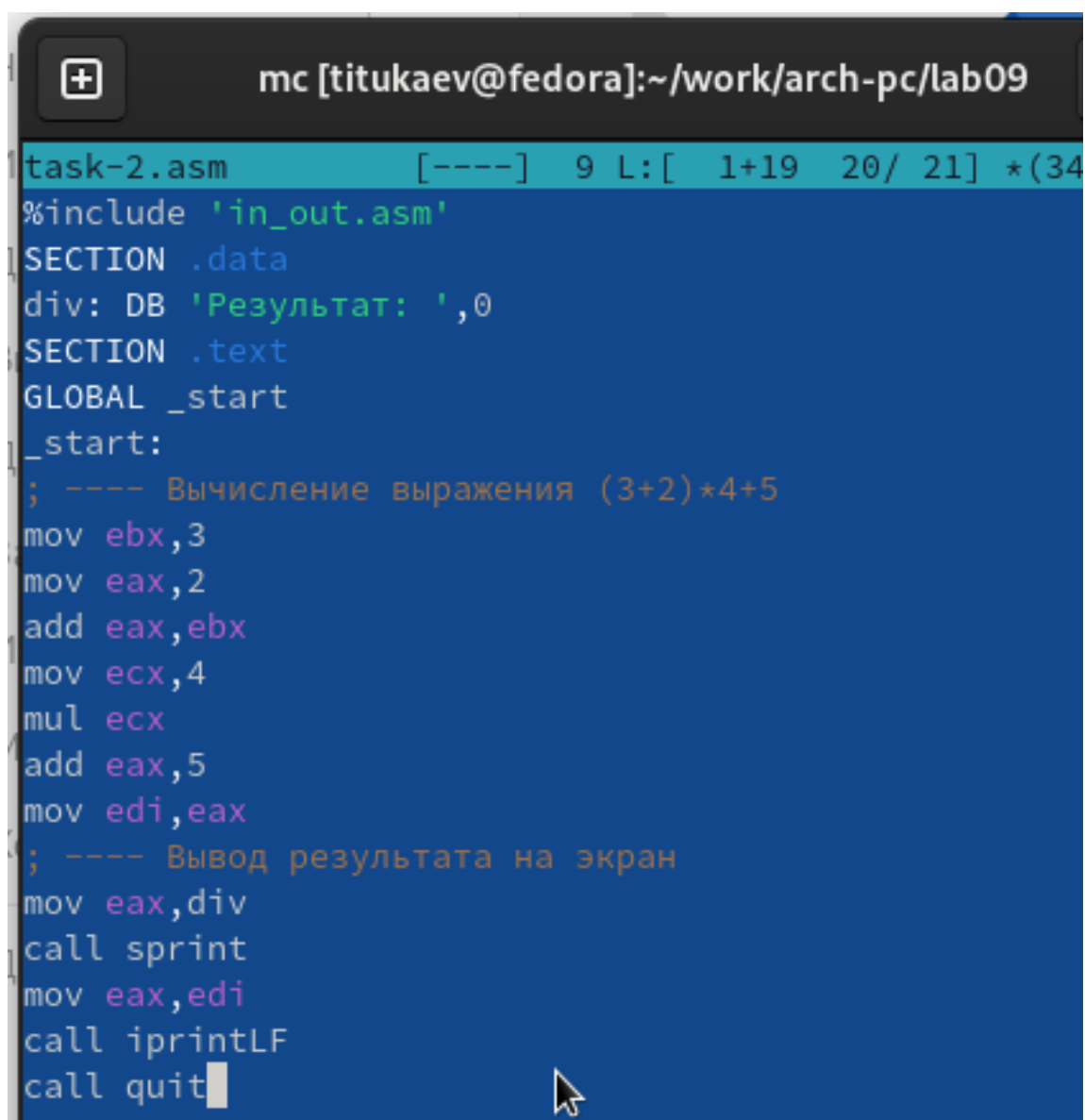
```

```

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

```

```
mc [titukaev@fedora]:~/work/arch-pc/lab09
task-2.asm [----] 9 L:[ 1+19 20/ 21] *(34
#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
```

Рис. 2.20: Код исправлен

```
titukaev@fedora:~/work/arch-pc/lab09 — gdb task-2

eax      0x804a000      134520832
ecx      0x4            4
edx      0x0            0
ebx      0x3            3
esp      0xffffd220     0xffffd220
ebp      0x0            0x0
esi      0x0            0
edi      0x19           25
eip      0x8049105     0x8049105 <_start+29>
eflags   0x202         [ IF ]
cs       0x23           35

B+ 0x80490e8 <_start>      mov     ebx,0x3
0x8049105 <_start+29>    call    0x804900f <sprint>
0x804910a <_start+34>    add     eax,edi
0x804910c <_start+36>    call    0x8049086 <iprintLF>
0x8049111 <_start+41>    call    0x80490db <quit>
                                ,0x5

                                04a000
>                                rint>

                                86 <iprintLF>

native process 3888 In: _start L17 PC: 0x8049105
(gdb) sNo process In: L?? PC: ??
Undefined command: "su". Try "help".
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 3888) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.