

Отчёт по лабораторной работе 9

Понятие подпрограммы. Отладчик GDB.

Останин Владислав Александрович НПМбв-01-21


Содержание

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. Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте 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. Внимательно изучите текст программы (Листинг 10.1).

Открыть ▾ 

lab9-1.asm
~/work/arch-pc/lab09

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80

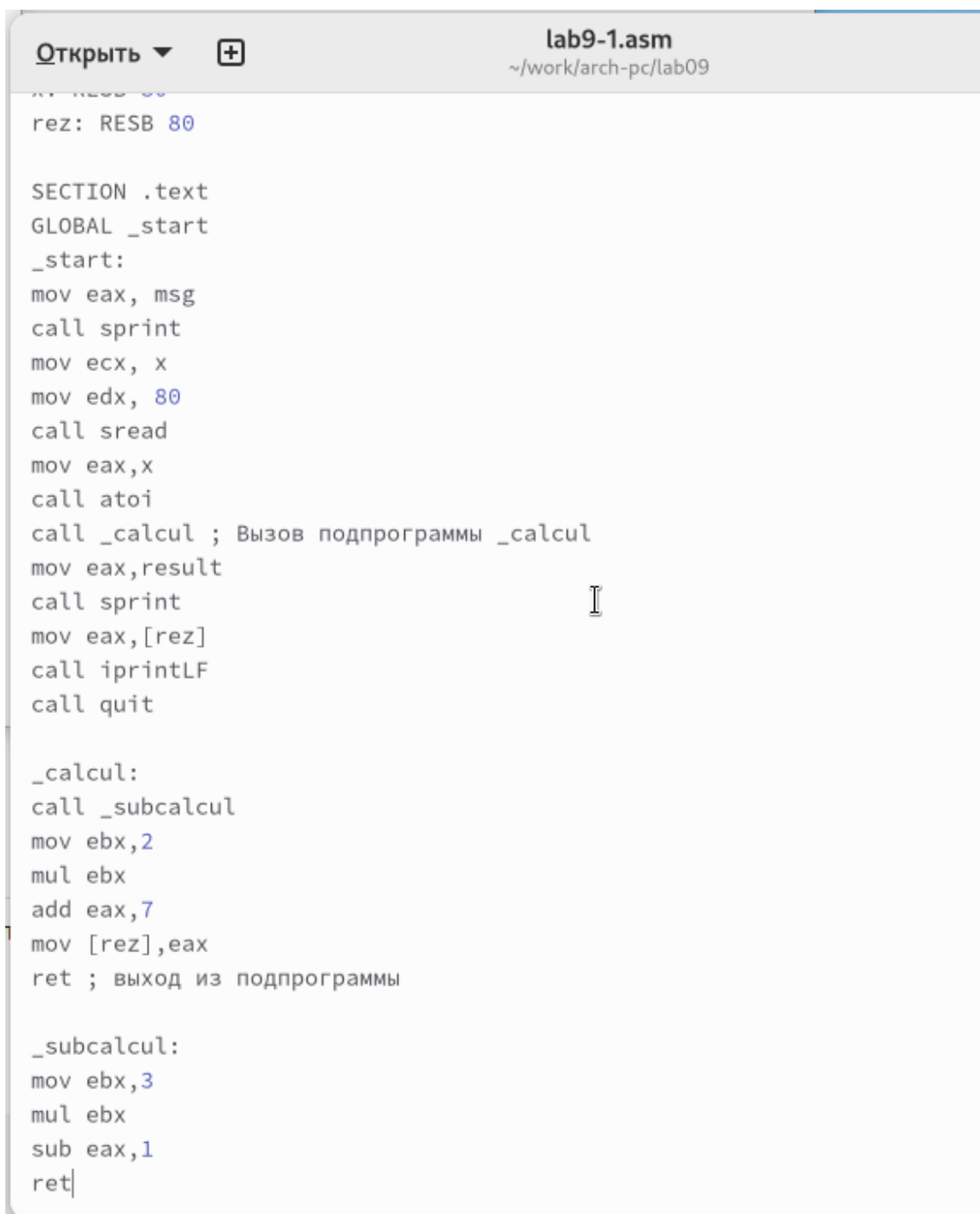
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Figure 1: Файл lab9-1.asm

```
[vlad-o-.o-.o@fedora lab09]$ nasm -f elf lab9-1.asm
[vlad-o-.o-.o@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[vlad-o-.o-.o@fedora lab09]$ ./lab9-1
Введите x: 3
2x+7=13
[vlad-o-.o-.o@fedora lab09]$
```

Figure 2: Работа программы lab9-1.asm

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



```

rez: RESB 80

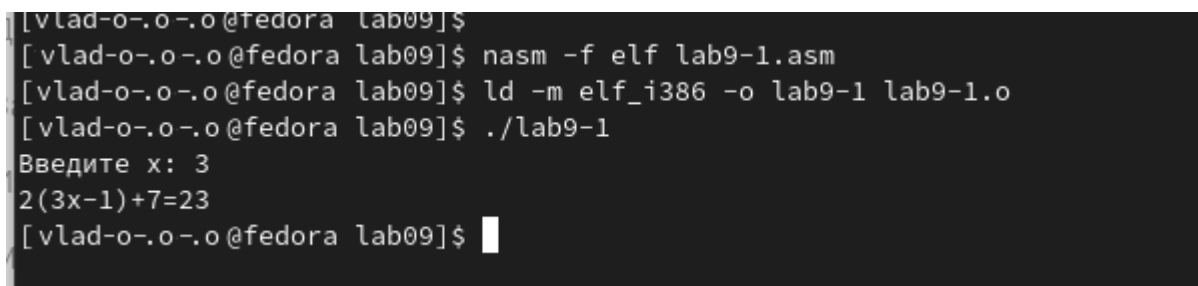
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit

_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

```

Figure 3: Файл lab9-1.asm



```

[vlad-o-o-o@fedora lab09]$
[vlad-o-o-o@fedora lab09]$ nasm -f elf lab9-1.asm
[vlad-o-o-o@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[vlad-o-o-o@fedora lab09]$ ./lab9-1
Введите x: 3
2(3x-1)+7=23
[vlad-o-o-o@fedora lab09]$

```

Figure 4: Работа программы lab9-1.asm

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



```
lab9-2.asm
~/work/arch-pc/lab09

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
```

Figure 5: Файл lab9-2.asm

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: Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r):(рис. [6])

```

[vlad-o-o-o@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[vlad-o-o-o@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[vlad-o-o-o@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)
(gdb) r
Starting program: /home/vlad/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 3657) exited normally]
(gdb)

```

Figure 6: Работа 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.

```

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 3657) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/vlad/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)

```

Figure 7: дисассимилированный 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)

```

Figure 8: дисассимилированный код в режиме интел

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверьте это с помощью команды `info breakpoints` (кратко `i b`) Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции (`mov ebx,0x0`) и установите точку.(рис. [9])

The screenshot shows a GDB debugger window with two main panes. The top pane displays assembly code with addresses, disassembled instructions, and comments. The bottom pane shows register information and the current instruction pointer (PC).

```

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
      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 3661 In: _start                                     L11  PC: 0x8049000
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202   [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--sics 0x23
35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb)

```

Figure 9: точка остановки

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


```
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
    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 3661 In: _start                                L15    PC: 0x8049014
eflags      0x202      [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--sics      0x23
35
ss          0x2b      43
ds          0x2b      43
es          0x2b      43
fs          0x0       0
gs          0x0       0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Figure 10: изменение регистров

```
(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>: "Lorld!\n\034"
(gdb)
```

Figure 11: изменение регистров

Посмотрите значение переменной msg1 по имени Посмотрите 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. Измените первый символ переменной msg1 Замените любой символ во второй переменной msg2.

```

process 3666 In: _start
(gdb) p/t $eax
$2 = 100
(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)

```

Figure 12: изменение значения переменной

Выведите в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. С помощью команды set измените значение регистра ebx:

```

native process 3661 In: _start
process 3666 In: _start
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb)

```

Figure 13: вывод значения регистра

С помощью команды set измените значение регистра ebx

```

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
    0x8049014 <_start+20> int     0x800x4
    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
B > 0x8049031 <_start+49> mov     ebx,0x0
    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 3661 In: _start L15 PC: 0x8049014
process 3666 In: _start L15 PC: 0x8049014
(gdb) p/t $edx 22 9031
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb) cont
Continuing.
hello, World!
Breakpoint 2, _start () at lab9-2.asm:22
(gdb)

```

Figure 14: вывод значения регистра

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

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

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.

```

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) r
Starting program: /home/vlad/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]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

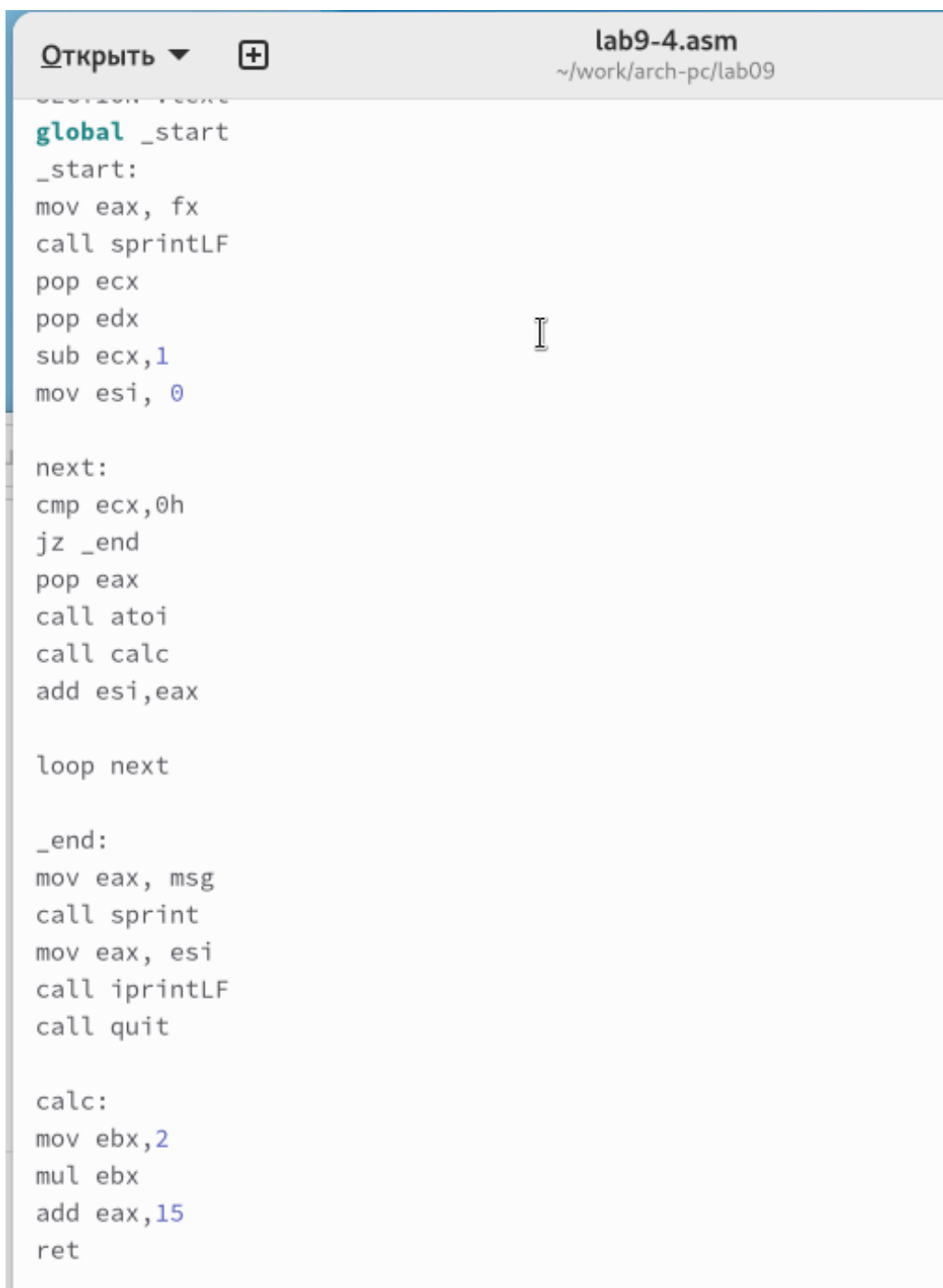
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1d0: 0x00000006
(gdb) x/s *(void*)($esp + 4)
0xffffd381: "/home/yusufsubanov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void*)($esp + 8)
0xffffd3ae: "argument"
(gdb) x/s *(void*)($esp + 12)
0xffffd3b7: "1"
(gdb) x/s *(void*)($esp + 16)
0xffffd3b9: "argument"
(gdb) x/s *(void*)($esp + 20)
0xffffd3c2: "2"
(gdb) x/s *(void*)($esp + 24)
0xffffd3c4: "argument 3"
(gdb)

```

Figure 15: вывод значения регистра

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

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



```
Открыть ▾ + lab9-4.asm
~/work/arch-pc/lab09

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

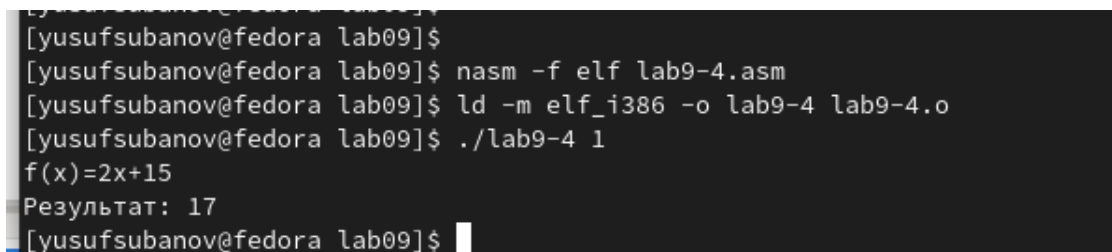
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
call calc
add esi, eax

loop next

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

calc:
mov ebx, 2
mul ebx
add eax, 15
ret
```

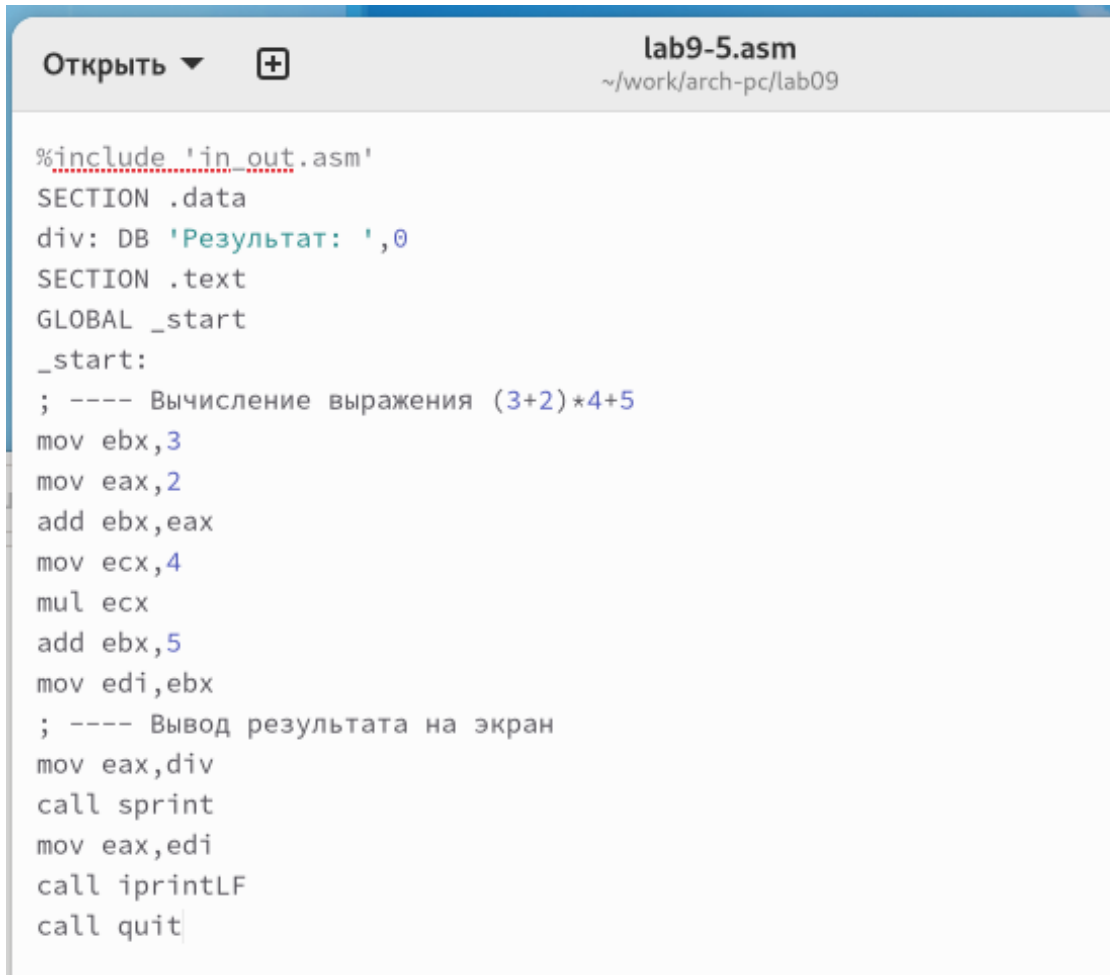
Figure 16: Файл lab9-4.asm



```
[yusufsubanov@fedora lab09]$
[yusufsubanov@fedora lab09]$ nasm -f elf lab9-4.asm
[yusufsubanov@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[yusufsubanov@fedora lab09]$ ./lab9-4 1
f(x)=2x+15
Результат: 17
[yusufsubanov@fedora lab09]$
```

Figure 17: Работа программы lab9-4.asm

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



```
Открыть ▾ + lab9-5.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
```

Figure 18: код с ошибкой

```
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x5      5
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+12>
eflags   0x206     [ PF IF ]

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
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>

native process 3848 In: _start L11 PC: 0x80490f4
No process In: L?? PC: ??
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 lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 10
[Inferior 1 (process 3848) exited normally]
(gdb) 
```

Figure 19: отладка

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

Открыть ▾

+

lab9-5.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 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
```

Figure 20: код исправлен


```
eax      0x5      5
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffd200 0xffffd200
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+12>
eflags   0x206    [ PF IF ]

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490f4 <_start+12>  mov     ecx,0x4
0x80490fb <_start+19>  add     eax,0x5
0x80490fe <_start+22>  mov     edi,eax
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>

native process 3921 In: _start L11 PC: 0x80490f4
No process In: L?? PC: ??
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 lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 3921) exited normally]
(gdb)
```

Figure 21: проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.