

Отчёта по лабораторной работе №9

Отладчик GDB

Гомес Лопес Теофания

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
2.1	Реализация подпрограмм в NASM	7
2.2	Отладка программ с помощью GDB	9
3	Выводы	20
	Список литературы	21

Список иллюстраций

2.1	Создала каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	7
2.2	Заполняла файл	8
2.3	Запускала файл и проверяла его работу	8
2.4	Изменяла файл, добавляя еще одну подпрограмму	9
2.5	Запускала файл и смотрела на его работу	9
2.6	Создала файл	9
2.7	Заполняла файл	10
2.8	Загружала исходный файл в отладчик	10
2.9	Запускала программу командой <code>run</code>	10
2.10	Запускала программу с брейкпоинтом	11
2.11	Смотрела дисассимилированный код программы	11
2.12	Переключалась на синтаксис Intel	11
2.13	Включала отображение регистров, их значений и результат дисассимилирования программы	12
2.14	Использовала команду <code>info breakpoints</code> и создала новую точку останова	13
2.15	Смотрела информацию	13
2.16	Отслеживала регистры	14
2.17	Смотрела значение переменной	14
2.18	Смотрела значение переменной	14
2.19	Меняла символ	14
2.20	Меняла символ	15
2.21	Смотрела значение регистра	15
2.22	Изменяла регистр командой <code>set</code>	15
2.23	Прописывала команды <code>c</code> и <code>quit</code>	15
2.24	Копировала файл	15
2.25	Создала и запускала в отладчике файл	16
2.26	Устанавливала точку останова	16
2.27	Изучала полученные данные	16
2.28	Копировала файл	16
2.29	Изменяла файл	17
2.30	Проверяем работу программы	17
2.31	Создала файл	18
2.32	Изменяла файл	18
2.33	Создала и смотрим на работу программы	18
2.34	Ищем ошибку регистров в отладчике	19
2.35	Меняла файл	19

2.36 Создала и запускала файл	19
---	----

Список таблиц

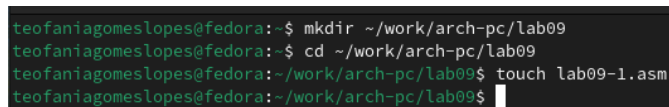
1 Цель работы

Познакомиться с методами отладки при помощи GDB, его возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

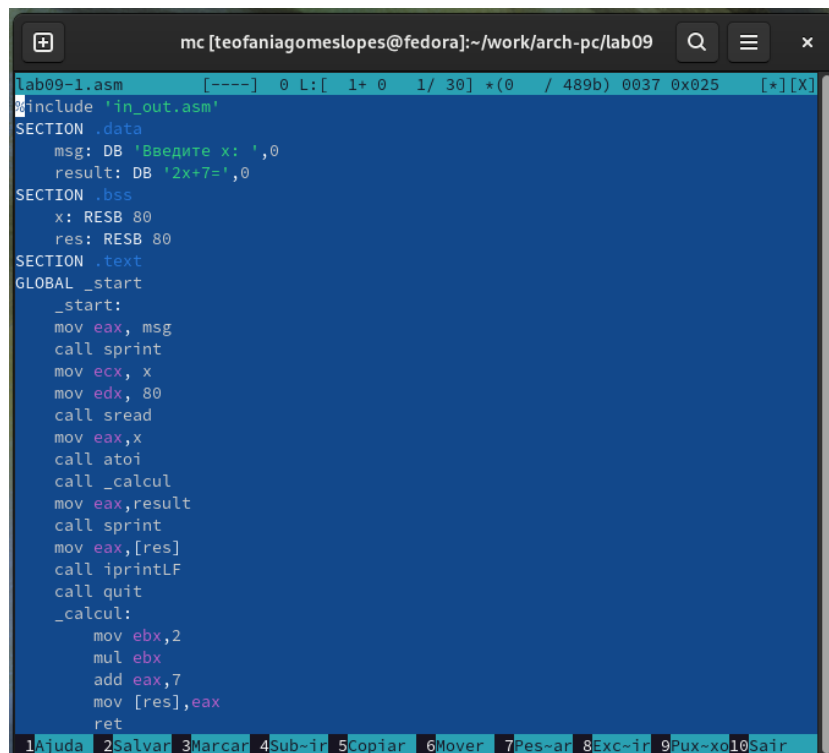
Создала каталог для программ ЛБ9, и в нем создала файл (рис. 2.1).



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

Рис. 2.1: Создала каталог с помощью команды `mkdir` и файл с помощью команды `touch`

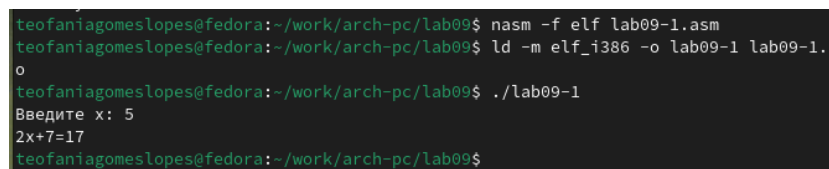
Открывала файл в Midnight Commander и заполняла его в соответствии с листингом 9.1 (рис. 2.2).



```
lab09-1.asm [-----] 0 L: [ 1+ 0 1/ 30] *(0 / 489b) 0037 0x025 [*][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:
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    call _calcul
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF
    call quit
_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret
```

Рис. 2.2: Заполняла файл

Создала исполняемый файл и запускала его (рис. 2.3).



```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$
```

Рис. 2.3: Запускала файл и проверяла его работу

Снова открывала файл для редактирования и изменяла его, добавив подпрограмму в подпрограмму(по условию) (рис. 2.4).


```

lab09-1.asm  [----]  0 L:[ 1+ 0 1/ 36] +(0 / 613b) 0037 0x025  [*][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:
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    call _calcul
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF
    call quit
_calcul:
    call _subcalcul
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret
_subcalcul:
    mov ebx, 3
    mul ebx
    sub eax, 1
    ret

```

Рис. 2.4: Изменяла файл, добавляя еще одну подпрограмму

Создала исполняемый файл и запускала его (рис. 2.5).

```

teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2(3x-1)+7=35
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$

```

Рис. 2.5: Запускала файл и смотрела на его работу

2.2 Отладка программ с помощью GDB

Создала новый файл в каталоге (рис. 2.6).

```

teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$

```

Рис. 2.6: Создала файл

Открывала файл в Midnight Commander и заполняла его в соответствии с листингом 9.2 (рис. 2.7).

```
lab09-2.asm [----] 0 L: [ 1+ 0 1/ 22] *(0 / 366b) 0083 0x053 [*] [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. 2.7: Заполняла файл

Получала исходный файл с использованием отладчика gdb (рис. 2.8).

```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ gdb lab09-2
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 lab09-2...
(gdb)
```

Рис. 2.8: Загружала исходный файл в отладчик

Запускала команду в отладчике (рис. 2.9).

```
[Inferior 1 (process 4200) exited normally]
(gdb) run
Starting program: /home/teofaniagomeslopes/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4200) exited normally]
(gdb)
```

Рис. 2.9: Запускала программу командой run

Устанавливала брейкпоинт на метку _start и запускала программу (рис. 2.10).

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

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

Рис. 2.10: Запускала программу с брейкпоинтом

Смотрела дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. 2.11).

```
(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. 2.11: Смотрела дисассимилированный код программы

Переключаемся на отображение команд с Intel'овским синтаксисом (рис. 2.12).

```
(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. 2.12: Переключалась на синтаксис Intel

Различия отображения синтаксиса машинных команд в режимах АТТ и Intel:

1.Порядок операндов: В АТТ синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel син-

таксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.

2.Разделители: В АТТ синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).

3.Префиксы размера операндов: В АТТ синтаксисе размер операнда указывается перед операндом с использованием префиксов, таких как “b” (byte), “w” (word), “l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.

4.Знак операндов: В АТТ синтаксисе операнды с позитивными значениями предваряются символом “”.*Intel*”.

5.Обозначение адресов: В АТТ синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.

Включала режим псевдографики (рис. 2.13).

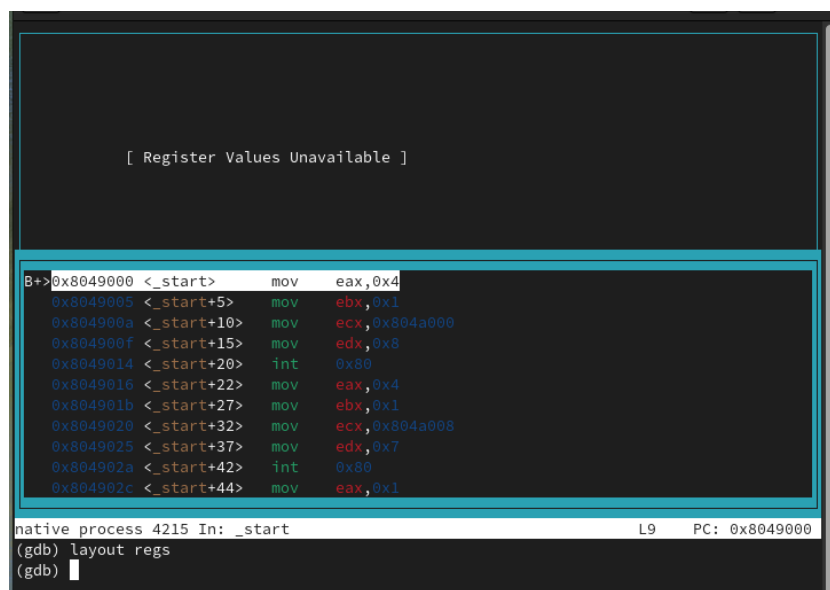


Рис. 2.13: Включала отображение регистров, их значений и результат дисассимилирования программы

Проверяла была ли установлена точка останова и устанавливала точку останова предпоследней инструкции (рис. 2.14).

```

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 4215 In: _start          L9      PC: 0x8049000
(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 *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb)

```

Рис. 2.14: Использовала команду info breakpoints и создала новую точку останова

Посмотрела информацию о всех установленных точках останова (рис. 2.15).

```

(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. 2.15: Смотрела информацию

Выполняла 5 инструкций командой si (рис. 2.16).

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd040 0xffffd040
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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 4215 In: _start L14 PC: 0x8049016
(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) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.16: Отслеживала регистры

Смотрела значение переменной msg1 по имени (рис. 2.17).

```

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

```

Рис. 2.17: Смотрела значение переменной

Смотрела значение переменной msg2 по адресу (рис. 2.18).

```

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

```

Рис. 2.18: Смотрела значение переменной

Изменила первый символ переменной msg1 (рис. 2.19).

```

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

```

Рис. 2.19: Меняла символ

Изменила первый символ переменной msg2 (рис. 2.20).

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

Рис. 2.20: Меняла символ

Смотрела значение регистра edx в разных форматах (рис. 2.21).

```
(gdb) p/t $edx
$1 = 1000
(gdb) p/s $edx
$2 = 8
(gdb) p/x $edx
$3 = 0x8
(gdb)
```

Рис. 2.21: Смотрела значение регистра

Изменяла регистр ebx (рис. 2.22).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)
```

Рис. 2.22: Изменяла регистр командой set

Прописывала команды для завершения программы и выхода из GDB (рис. 2.23).

```
(gdb) c
Continuing.
Lorld!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb)
```

Рис. 2.23: Прописывала команды c и quit

Копировала файл lab8-2.asm в файл с именем lab09-3.asm (рис. 2.24).

```
(gdb) layout asm
teofaniagomeslopes@fedora: ~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
teofaniagomeslopes@fedora: ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.as
```

Рис. 2.24: Копировала файл

Создала исполняемый файл и запускала его в отладчике GDB (рис. 2.25).

```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '5'
```

Рис. 2.25: Создала и запускала в отладчике файл

Установила точку останова перед первой инструкцией в программе и запустим ее (рис. 2.26).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/teofaniagomeslopes/work/arch-pc/lab09/lab09-3 2 3 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 lab09-3.asm:5
5      pop ecx
(gdb) x/x $esp
0xffffd030: 0x00000004
(gdb)
```

Рис. 2.26: Устанавливала точку останова

Смотрела позиции стека по разным адресам (рис. 2.27).

```
(gdb) x/s *(void**)(esp + 4)
0xffffd1f7: "/home/teofaniagomeslopes/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd22b: "2"
(gdb) x/s *(void**)(esp + 12)
0xffffd22d: "3"
(gdb) x/s *(void**)(esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.27: Изучала полученные данные

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

##Задание для самостоятельной работы

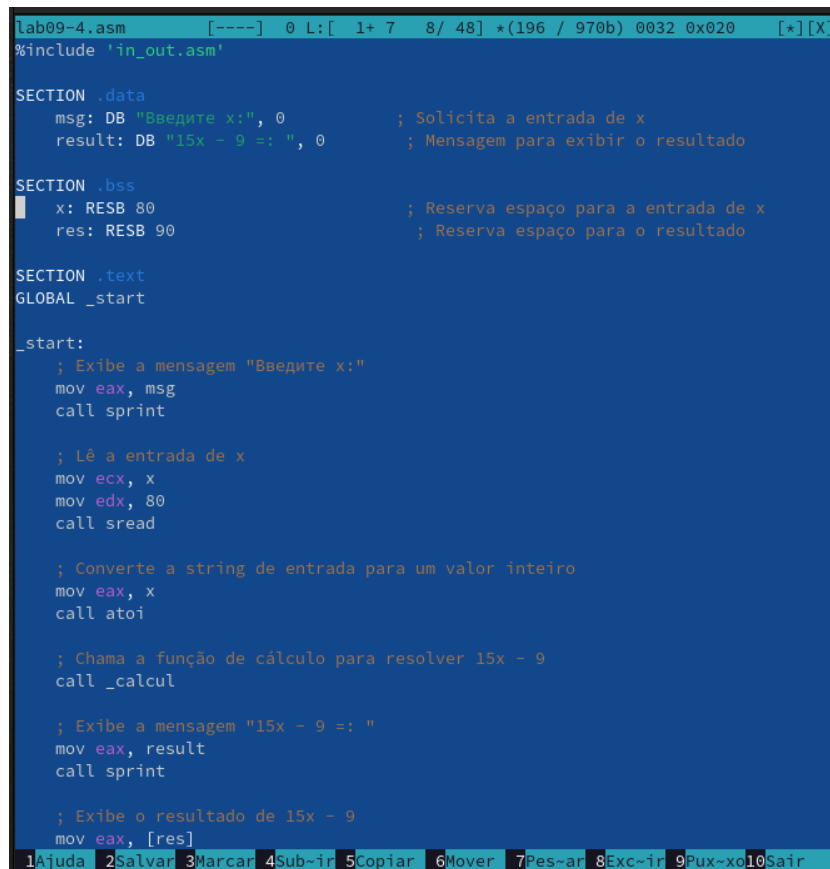
###Задание 1

Копировала файл lab8-4.asm(ср №1 в ЛБ8) в файл с именем lab09-3.asm (рис. 2.28).

```
(gdb) layout asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
```

Рис. 2.28: Копировала файл

Открывала файл в Midnight Commander и меняла его, создавая подпрограмму (рис. 2.29).



```
lab09-4.asm  [----]  0 L: [ 1+ 7 8/ 48] *(196 / 970b) 0032 0x020  [*][X]
#include 'in_out.asm'

SECTION .data
    msg: DB "Введите x:", 0          ; Solicita a entrada de x
    result: DB "15x - 9 =: ", 0      ; Mensagem para exibir o resultado

SECTION .bss
    x: RESB 80                      ; Reserva espaço para a entrada de x
    res: RESB 90                    ; Reserva espaço para o resultado

SECTION .text
GLOBAL _start

_start:
    ; Exibe a mensagem "Введите x:"
    mov eax, msg
    call sprint

    ; Lê a entrada de x
    mov ecx, x
    mov edx, 80
    call sread

    ; Converte a string de entrada para um valor inteiro
    mov eax, x
    call atoi

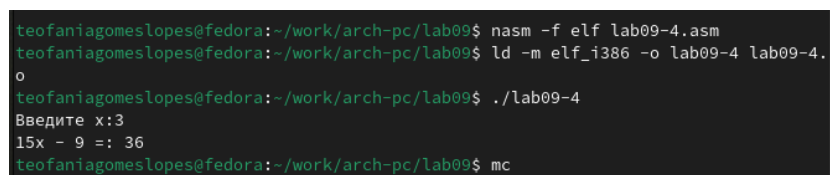
    ; Chama a função de cálculo para resolver 15x - 9
    call _calcul

    ; Exibe a mensagem "15x - 9 =: "
    mov eax, result
    call sprint

    ; Exibe o resultado de 15x - 9
    mov eax, [res]
```

Рис. 2.29: Изменяла файл

Создала исполняемый файл и запускаем его (рис. 2.30).



```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ./lab09-4
Введите x:3
15x - 9 =: 36
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ mc
```

Рис. 2.30: Проверяем работу программы

###Задание 2

Создала новый файл в дирректории (рис. 2.31).

```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ touch lab09-5.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$
```

Рис. 2.31: Создала файл

Открывала файл в Midnight Commander и заполняем его в соответствии с листингом 9.3 (рис. 2.32).

```
lab09-5.asm [----] 0 L: 1+ 0 1/ 19] *(0 / 285b) 0037 0x025 [*][X]
%include 'in_out.asm'
SECTION .data
div: DB 'Результат:', 0
SECTION .text
GLOBAL _start
_start:
    mov ebx,3
    mov eax,2
    add ebx,eax
    mov ecx,4
    mul ecx
    add ebx,5
    mov edi,ebx
    mov eax,dib
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис. 2.32: Изменяла файл

Создала исполняемый файл и запускала его (рис. 2.33).

```
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат:10
teofaniagomeslopes@fedora:~/work/arch-pc/lab09$
```

Рис. 2.33: Создала и смотрим на работу программы

Создала исполняемый файл и запускала его в отладчике GDB и смотрела на изменение регистров командой si (рис. 2.34).

The screenshot shows a debugger window with the following assembly code:

```

B+ 0x80490e8 <_start>    mov     ebx,0x1
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f9 <_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
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
0x804910c <_start+36>   call   0x8049086 <iprintfLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al

```

Below the assembly code, the register values are displayed:

Register	Value
esp	0xffffd0b0
ebp	0x0
esi	0x0
edi	0x0
eip	0x80490f4
eflags	0x206
cs	0x23
ss	0x2b
ds	0x2b
es	0x2b

The status bar at the bottom indicates: native process 8870 In: _start L11 PC: 0x80490f9

Рис. 2.34: Ищем ошибку регистров в отладчике

Изменяла программу для корректной работы (рис. 2.35)

The screenshot shows a text editor window with the following assembly code:

```

lab09-5.asm  [----]  0 L: [ 1+ 0 1/ 27 ] *(0 / 966b) 0037 0x025 [*][X]
#include 'in_out.asm'

SECTION .data
    div: DB 'Результат:', 0 ; Definir a string "Результат:" corretamente

SECTION .text
GLOBAL _start

_start:
    mov ebx, 3 ; Carrega 3 em EBX
    mov eax, 2 ; Carrega 2 em EAX
    add ebx, ebx ; Soma EBX e EAX (EBX = 3 + 2 = 5)
    mov ecx, 4 ; Carrega 4 em ECX
    mul ecx ; Multiplica EAX por ECX (EAX = 5 * 4 = 20)
    add ebx, 5 ; Soma 5 em EBX (EBX = 5 + 5 = 10)
    mov edi, eax ; Copia o resultado para EDI (EDI = 10)

    ; Exibe a mensagem "Результат:"
    mov eax, div ; Carrega o endereço da string "Результат:" em EAX
    call sprint ; Exibe a string

    ; Exibe o valor armazenado em EDI
    mov eax, edi ; Carrega o valor de EDI em EAX (EAX = 10)
    call iprintfLF ; Exibe o valor em EAX

    call quit ; Encerra o programa

```

Рис. 2.35: Меняла файл

Создала исполняемый файл и запускала его (рис. 2.36)

Создала и запускала файл

Рис. 2.36: Создала и запускала файл

3 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.

Список литературы