

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Чернятьева Олеся Олеговна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Самостоятельная работа	17
3.1	Здание №1	17
4	Вывод	24

Список иллюстраций

2.1	Создание директории	5
2.2	Редактирование файла	6
2.3	Запуск исполняемого файла	6
2.4	Редоктирование	7
2.5	Запуск исполняемого файла	7
2.6	Создание исполняемого файла	9
2.7	Работа с отладчиком	9
2.8	Дисассамблирование кода	10
2.9	Синтаксис Intel	10
2.10	Режим псевдографики	11
2.11	Просмотр точек останова	12
2.12	Вывод значений регистров	13
2.13	Вывод значений переменных	13
2.14	Изменение значений переменных	14
2.15	Изменение значений переменных	14
2.16	Запуск исполняемого файла	14
2.17	Вывод значений переменных	14
2.18	Вывод значений переменных	15
2.19	копирование файла	15
2.20	Запуск отладчика	15
2.21	Изменение значений переменных	16
2.22	Просмотр содержимого в esp	16
2.23	Вывод значений переменных	16
3.1	Копирование файла	17
3.2	Изменение программы	18
3.3	Запуск исполняемого файла	18
3.4	Программа вычисления выражения $(3 + 2) * 4 + 5$	19
3.5	Запуск файла в отладчике	20
3.6	Запуск программы	20
3.7	Работа с отладчиком	21
3.8	Проверка значений регистров	22
3.9	Выявление главных ошибок	22
3.10	Выявление главных ошибок	23
3.11	Исправление ошибок в программе	23
3.12	Запуск исполняемого файла	23

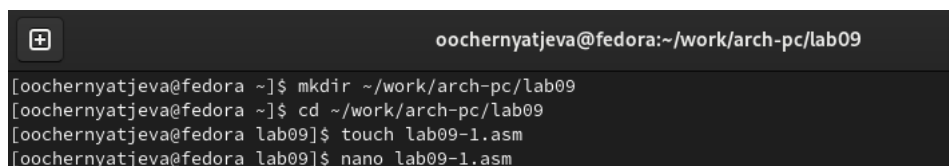
1 Цель работы

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

2 Выполнение лабораторной работы

Шаг 1

С помощью утилиты `mkdir` создаю директорию `lab09`, перехожу в нее и создаю файл для работы. (рис. [2.1])

A screenshot of a terminal window with a dark background. The title bar at the top reads "oochernyatjeva@fedora:~/work/arch-pc/lab09". The terminal shows four lines of commands and their outputs: 1. "[oochernyatjeva@fedora ~]\$ mkdir ~/work/arch-pc/lab09" 2. "[oochernyatjeva@fedora ~]\$ cd ~/work/arch-pc/lab09" 3. "[oochernyatjeva@fedora lab09]\$ touch lab09-1.asm" 4. "[oochernyatjeva@fedora lab09]\$ nano lab09-1.asm" The cursor is at the end of the fourth line.

```
oochernyatjeva@fedora:~/work/arch-pc/lab09
[oochernyatjeva@fedora ~]$ mkdir ~/work/arch-pc/lab09
[oochernyatjeva@fedora ~]$ cd ~/work/arch-pc/lab09
[oochernyatjeva@fedora lab09]$ touch lab09-1.asm
[oochernyatjeva@fedora lab09]$ nano lab09-1.asm
```

Рис. 2.1: Создание директории

Шаг 2

Открываю созданный файл `lab9-1.asm`, вставляю в него программу с использованием подпрограммы(рис.[2.2]).

```
oochernyatjeva@fedora:~/work/arch-pc/lab09 — nano lab09-1.asm
GNU nano 7.2 lab09-1.asm
#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 iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
```

Рис. 2.2: Редактирование файла

Шаг 3

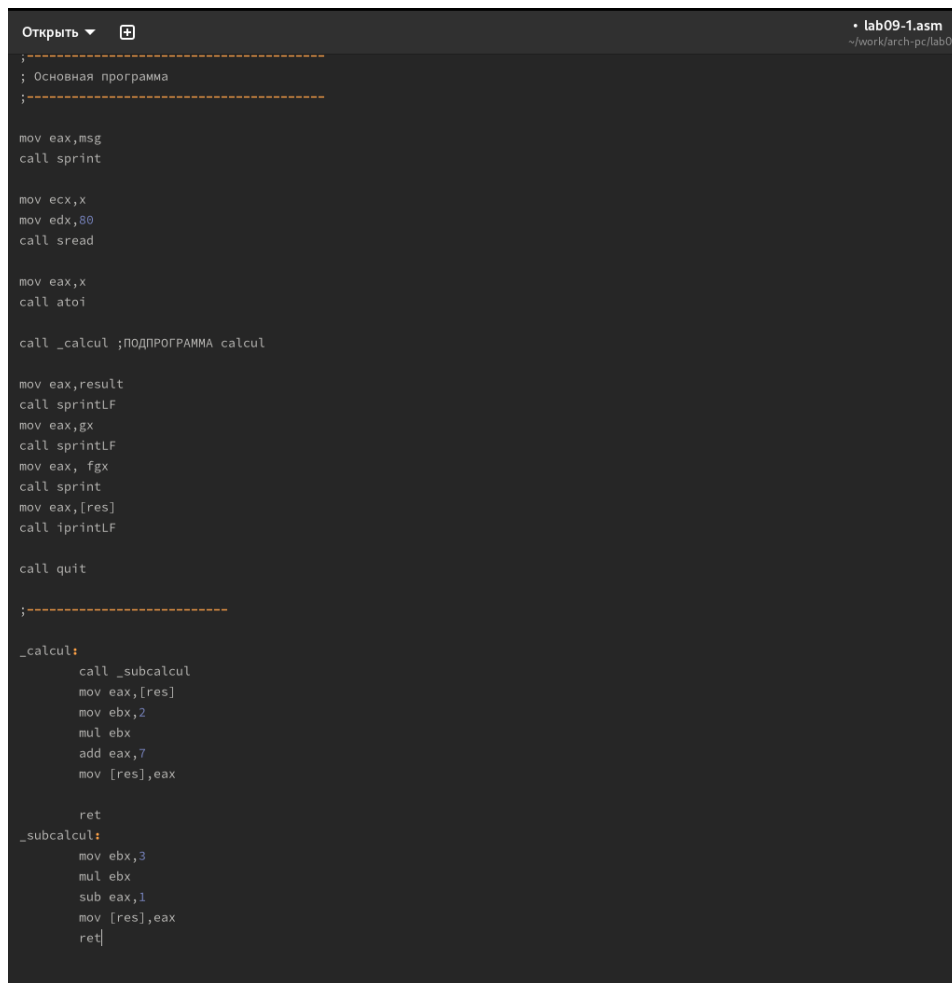
Создаю исполняемый файл программы и запускаю его (рис. [2.3]).

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

Рис. 2.3: Запуск исполняемого файла

Шаг 4

Изменяю текст программы для вычисления композиции f от g , при $g(x) = 3x - 1$.
Создаю новую подпрограмму `_subcalcul` для вычисления функции g (рис. [2.4]).




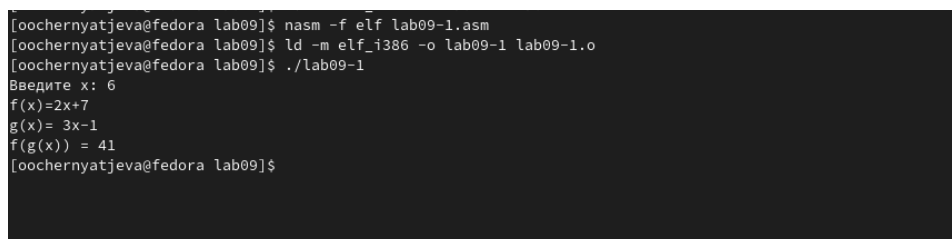
```
Открыть ▾  lab09-1.asm  
~\work\arch-pc\lab09  
;-----  
; Основная программа  
;-----  
  
mov eax,msg  
call sprint  
  
mov ecx,x  
mov edx,80  
call sread  
  
mov eax,x  
call atoi  
  
call _calcul ;ПОДПРОГРАММА calcul  
  
mov eax,result  
call sprintLF  
mov eax,gx  
call sprintLF  
mov eax,fgx  
call sprint  
mov eax,[res]  
call iprintLF  
  
call quit  
  
;-----  
  
_calcul:  
    call _subcalcul  
    mov eax,[res]  
    mov ebx,2  
    mul ebx  
    add eax,7  
    mov [res],eax  
  
    ret  
_subcalcul:  
    mov ebx,3  
    mul ebx  
    sub eax,1  
    mov [res],eax  
    ret
```

Рис. 2.4: Редоктирование

Шаг 5

Создаю исполняемый файл и проверяю работу программы (рис. [2.5]).



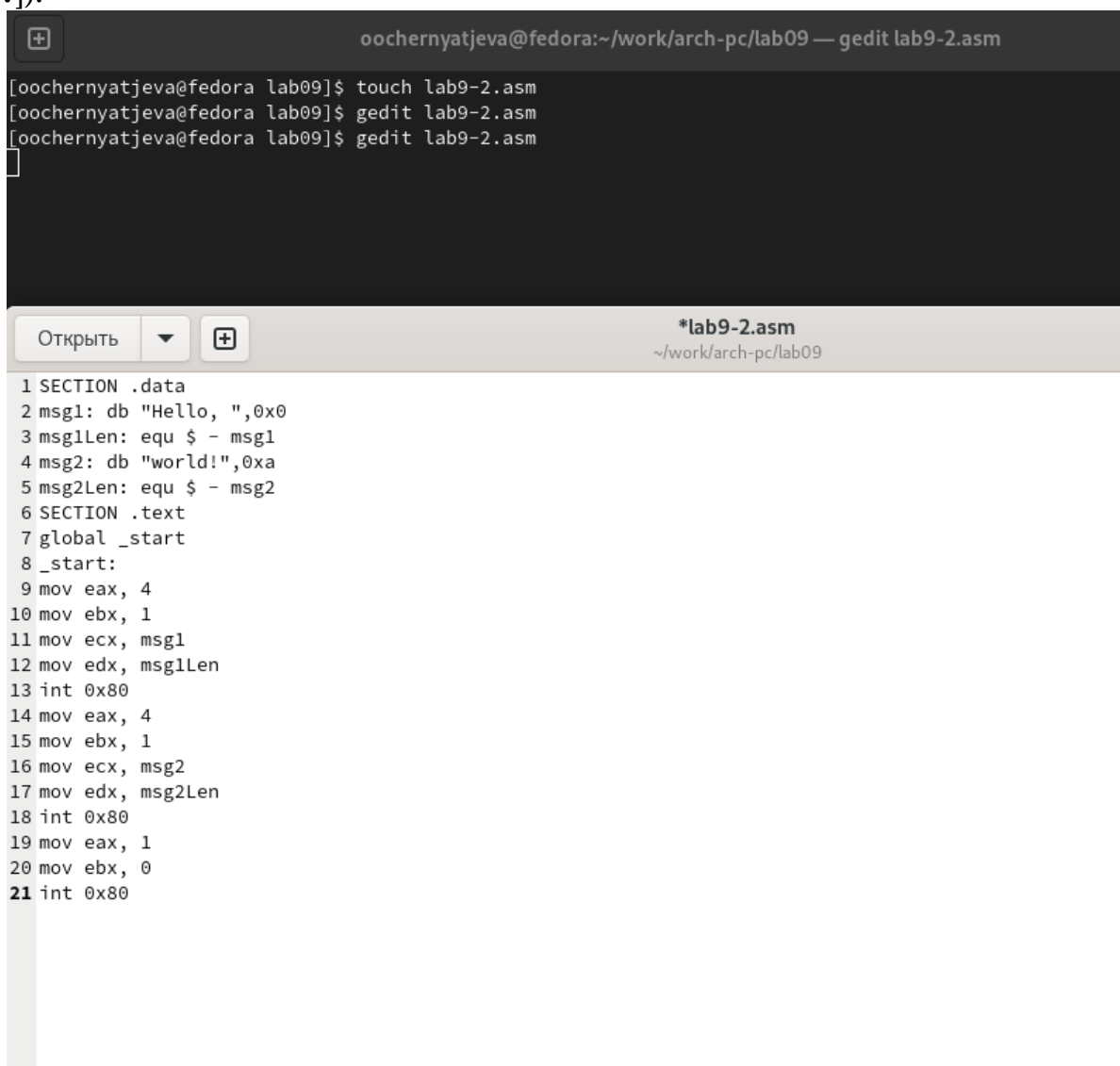
```
[oochernyatjeva@fedora lab09]$ nasm -f elf lab09-1.asm  
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o  
[oochernyatjeva@fedora lab09]$ ./lab09-1  
Введите x: 6  
f(x)=2x+7  
g(x)= 3x-1  
f(g(x)) = 41  
[oochernyatjeva@fedora lab09]$
```

Рис. 2.5: Запуск исполняемого файла

- Программа отработала верно!!

Шаг 6

Создаю новый файл lab9-2.asm и вставляю в него текст из Листинга 9.2 (рис. [??]).



The image shows two screenshots. The top screenshot is a terminal window with the title bar "oochernyatjeva@fedora:~/work/arch-pc/lab09 — gedit lab9-2.asm". It contains three lines of commands: `touch lab9-2.asm`, `gedit lab9-2.asm`, and `gedit lab9-2.asm`. The bottom screenshot is a text editor window titled "*lab9-2.asm" with the path "~/work/arch-pc/lab09". It displays the following assembly code:

```
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
```

{ #fig:006 width=80% }

Шаг 7

Создаю исполняемый файл, файл листинга для работы с отладчиком GDB (рис. [2.6]).


```
[oochernyatjeva@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[oochernyatjeva@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
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. 2.6: Создание исполняемого файла

Шаг 8

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды `run`, и для более подробного анализа программы, вставляю брэйкпоинт на метку `_start` (рис. [2.7]).

```
(gdb) run
Starting program: /home/darfonos/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]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Hello, world!
[Inferior 1 (process 4971) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8040000: file lab9-2.asm, line 9.
(gdb)
```

Рис. 2.7: Работа с отладчиком

Шаг 9

Посмотрим дизассемблированный код, начиная с этой метки. (рис. [2.8]).

```

(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. 2.8: Дисассамблирование кода

Шаг 10

Так же посмотрим как выглядит дизассемблированный код с синтаксисом Intel (рис. [2.9]).

```

(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. 2.9: Синтаксис Intel

- В представлении АТТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении intel так записываются адреса вторых аргументов.

Шаг 11

Включим режим псевдографики, с помощью которого отображается код программы и содержимое регистров (рис. [2.10]).

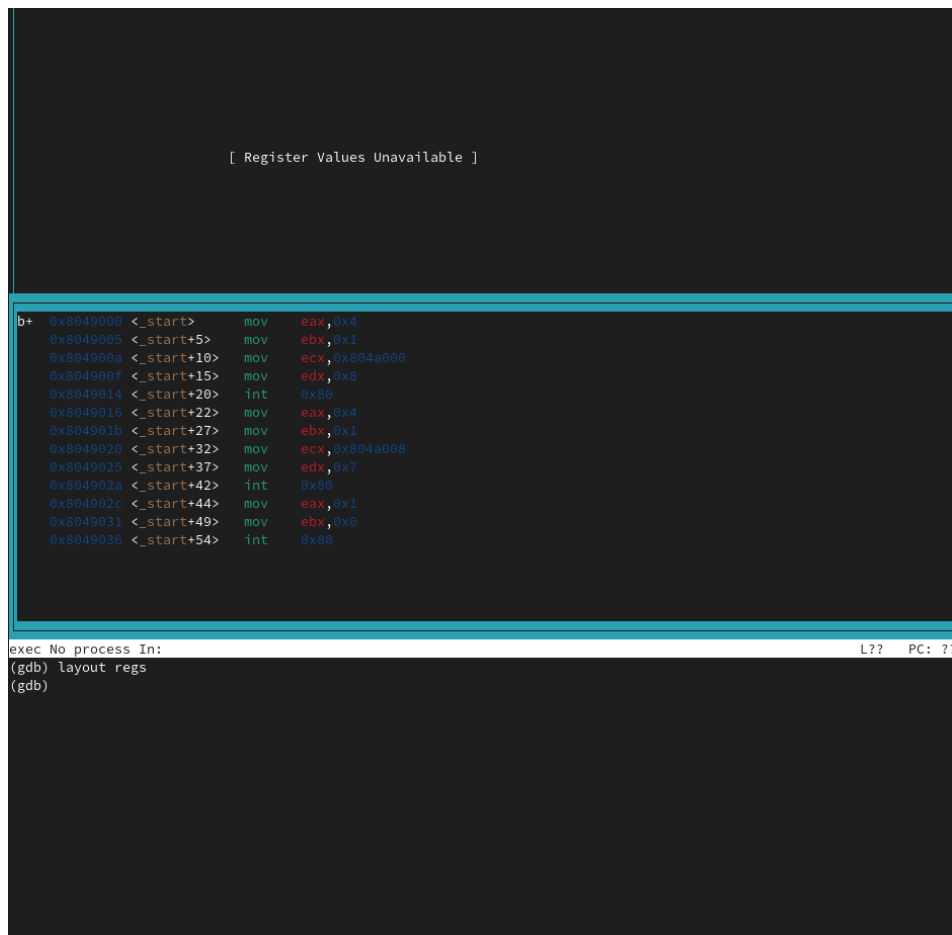


Рис. 2.10: Режим псевдографики

Шаг 12

Посмотрим информацию о наших точках останова и сразу добавим еще одну точку .(рис. [2.11]).

```
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

exec No process in:
(gdb) layout regs
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
(gdb) b *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
2        breakpoint keep y  0x08049031 lab9-2.asm:20
(gdb)
```

Рис. 2.11: Просмотр точек останова

Шаг 13

Так же можно выводить значения регистров. Делается это командой `i r`. Псевдографика предствалена на (рис. [2.12]).

```

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags   0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

lab9-2.asm
B+> 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
b+ 21  int 0x80
22
23
24
25

native process 5061 In: _start L9 PC: 0x8049000
(gdb) i r
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb)

```

Рис. 2.12: Вывод значений регистров

Шаг 14

В отладчике можно вывести текущее значение переменных. Сделать это можно по имени или по адресу: выводим значения переменных msg1 и msg2 (рис. [2.13]).

```

(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 2.13: Вывод значений переменных

Шаг 15

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. [2.14]).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x604a000 <msg1>:      "hello, "
(gdb) █
```

Рис. 2.14: Изменение значений переменных

- Заменяю первый символ 'H' на 'h'

Шаг 16

Заменяю первый символ переменной msg2 на символ j. (рис. [2.15]).

```
(gdb) set {char}&msg2='k'
(gdb) x/1sb &msg2
0x604a008 <msg2>:      "korld!\n\034"
(gdb) █
```

Рис. 2.15: Изменение значений переменных

Шаг 17

Выводить можно так же содержимое регистров. Выведем значение ebx в разных форматах: строчном, 16-ричном . (рис. [2.16]).

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

Рис. 2.16: Запуск исполняемого файла

Шаг 18

Как и переменным, регистрам можно задавать значения (рис. [2.17]).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) p/t $ebx
$2 = 110010
(gdb) p/x $ebx
$3 = 0x32
(gdb) █
```

Рис. 2.17: Вывод значений переменных

Шаг 19

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. [2.18]).

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

Рис. 2.18: Вывод значений переменных

- Однако при попытке задать строчное значение, происходит ошибка.

Завершим работу в gdb командами `continue`, она закончит выполнение программы, и `exit`, она завершит сеанс gdb

Шаг 20

Скопируем файл из лабораторной 9, переименуем её и создадим исполняемый файл.(рис. [2.19]).

```
[oochernyatjeva@fedora lab09]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
[oochernyatjeva@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[oochernyatjeva@fedora lab09]$
```

Рис. 2.19: копирование файла

Шаг 21

Откроем отладчик и зададим аргументы. (рис. [2.20]).

```
[oochernyatjeva@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
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 lab09-3...
(gdb) run
Starting program: /home/darfonos/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3
```

Рис. 2.20: Запуск отладчика

Шаг 22

Создадим точку останова на метке `_start` и запустим программу(рис. [2.21]).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/darfonos/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) |
```

Рис. 2.21: Изменение значений переменных

Шаг 23

Посмотрим на содержимое стека, что расположено по адресу, находящемуся в регистре `esp`(рис. [2.22]).

```
(gdb) x/x $esp
0xffffd160: 0x00000005
(gdb)
```

Рис. 2.22: Просмотр содержимого в `esp`

Шаг 24

Далее посмотрим на все остальные аргументы в стеке. Их адреса располагаются в 4 байтах друг от друга(именно столько занимает элемент стека) (рис. [2.23]).

```
(gdb) x/s *(void**)(esp + 4)
0xffffd31b: "/home/darfonos/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp)
0x5: <error: Cannot access memory at address 0x5>
(gdb) x/s *(void**)(esp + 4)
0xffffd31b: "/home/darfonos/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd345: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd357: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd368: "2"
(gdb) x/s *(void**)(esp + 20)
A syntax error in expression, near `'.
(gdb) |
```

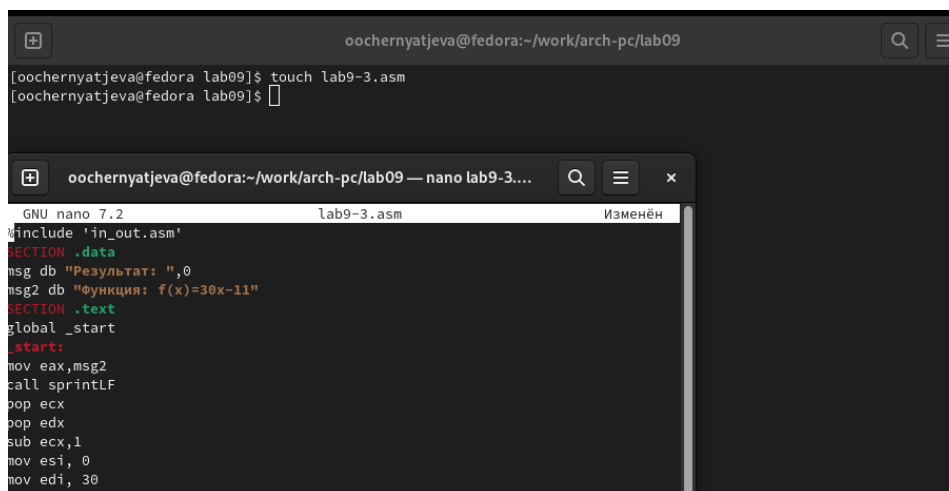
Рис. 2.23: Вывод значений переменных

3 Самостоятельная работа

3.1 Задание №1

Шаг 1

Копирую программу из лабораторной 8 и переименовываю его в lab9-4 (рис. [3.1]).



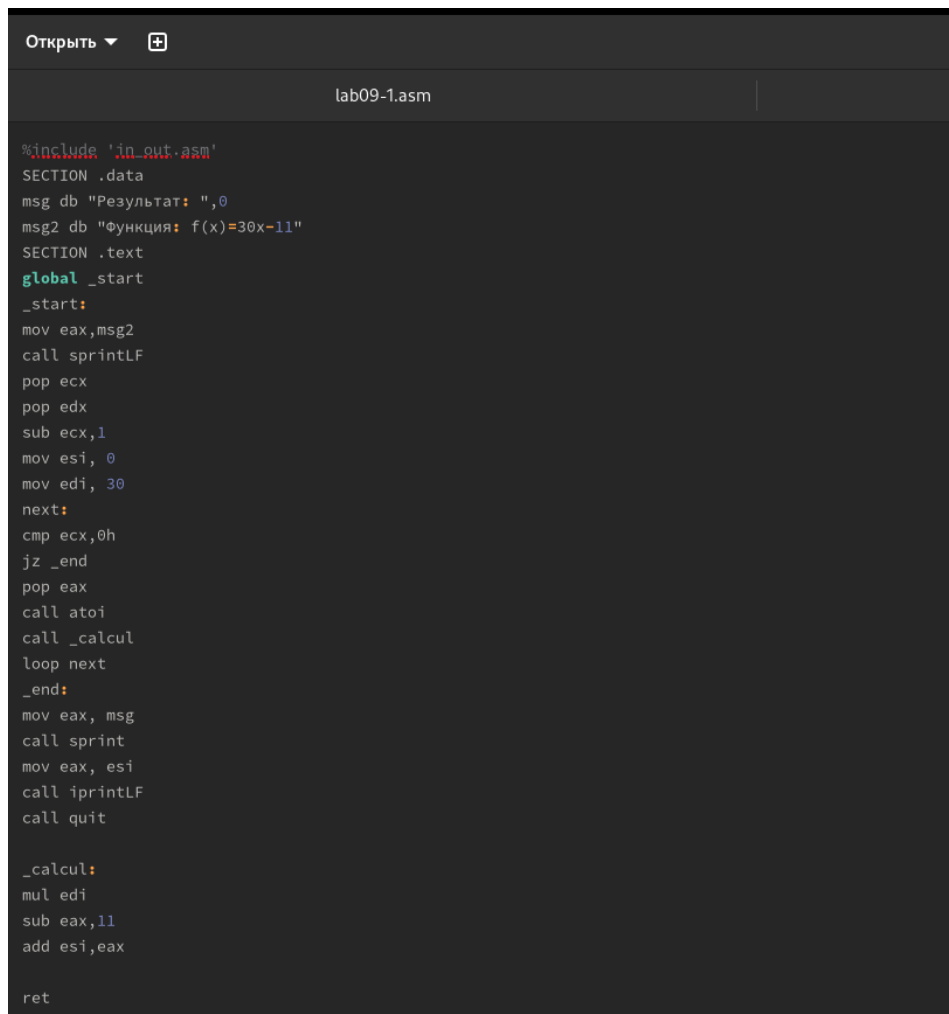
The image shows a terminal window and a nano editor window. The terminal window shows the command `touch lab9-3.asm` being executed. The nano editor window shows the contents of `lab9-3.asm`, which includes assembly code for a program that prints a message and calls a function.

```
GNU nano 7.2 lab9-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=30x-11"
SECTION .text
global _start
_start:
mov eax,msg2
call sprintf
pop ecx
pop edx
sub ecx,1
mov esi, 0
mov edi, 30
```

Рис. 3.1: Копирование файла

Шаг 2

Изменяю текст программы с использованием подпрограмм (рис. [3.2]).



```
Открыть ▾ +
lab09-1.asm

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=30x-11"
SECTION .text
global _start
_start:
mov eax,msg2
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0
mov edi, 30
next:
cmp ecx,0h
jz _end
pop eax
call atoi
call _calcul
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

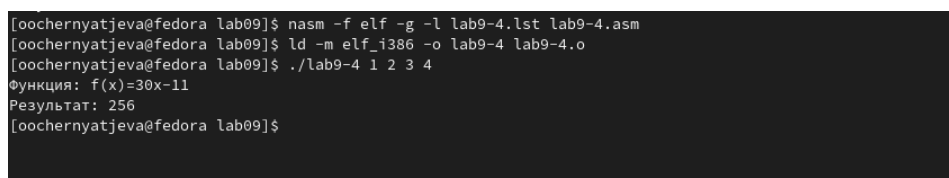
_calcul:
mul edi
sub eax,11
add esi,eax

ret
```

Рис. 3.2: Изменение программы

Шаг 3

Создаю исполняемый файл и проверяю работу изменённой программы .(рис. [3.3]).



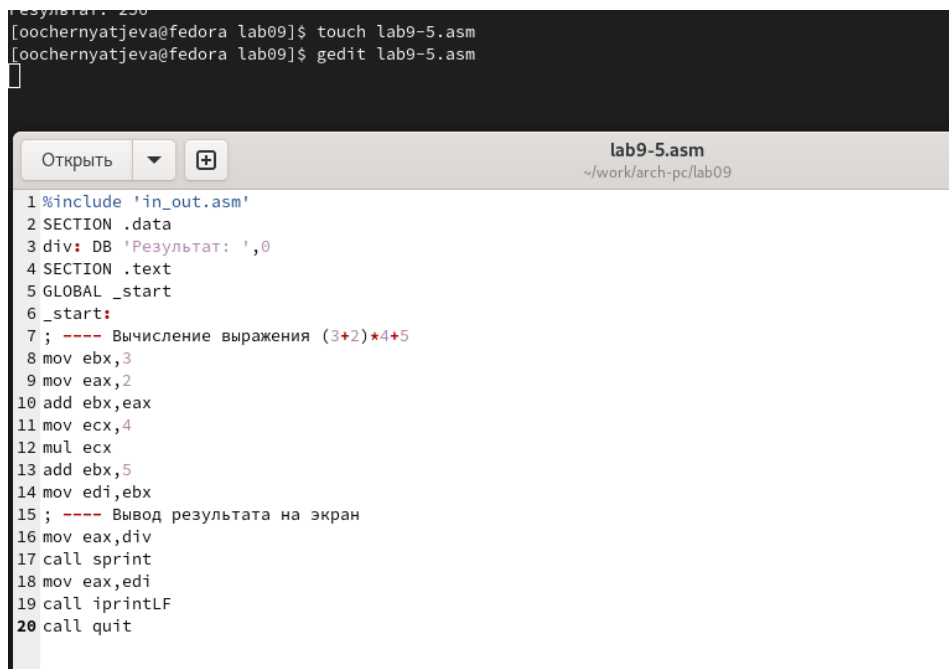
```
[oochernyatjeva@fedora lab09]$ nasm -f elf -g -l lab9-4.lst lab9-4.asm
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[oochernyatjeva@fedora lab09]$ ./lab9-4 1 2 3 4
Функция: f(x)=30x-11
Результат: 256
[oochernyatjeva@fedora lab09]$
```

Рис. 3.3: Запуск исполняемого файла

Программа отработала верно ## Задание №2

Шаг 1

Создаю новый файл и вставляю в него программу из листинга (рис. [3.4]).



```
[oochernyatjeva@fedora lab09]$ touch lab9-5.asm
[oochernyatjeva@fedora lab09]$ gedit lab9-5.asm
```

lab9-5.asm
~/work/arch-pc/lab09

```
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. 3.4: Программа вычисления выражения $(3 + 2) * 4 + 5$

Шаг 2

Запускаю программу и проверяю его работу и вижу, что результат вычисления неправильный. (рис. [3.6]).

```
[oochernyatjeva@fedora lab09]$ nasm -f elf -g -l lab9-5.lst lab9-5.asm
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab9-5 lab9-5.o
[oochernyatjeva@fedora lab09]$ gdb lab9-5
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
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 lab9-5...
(gdb)
```

Рис. 3.5: Запуск файла в отладчике

```
(gdb) run
Starting program: /home/darfonos/work/arch-pc/lab09/lab9-5

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.
Результат: 10
```

Рис. 3.6: Запуск программы

Шаг 3

Далее открываю программу в отладчике. Для того, чтобы найти ошибку дисасемблирую программу и добавляю брейкпоинты в основной части программы (рис. [3.7]).

```

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
0x08049100 <+24>:   mov     eax,0x804a000
0x08049105 <+29>:   call    0x804900f <sprint>
0x0804910a <+34>:   mov     eax,edi
0x0804910c <+36>:   call    0x8049086 <iprintLF>
0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb) b *0x080490f4
Breakpoint 1 at 0x80490f4: file lab9-5.asm, line 11.
(gdb) b *0x08049100
Breakpoint 2 at 0x8049100: file lab9-5.asm, line 16.
(gdb)

```

Рис. 3.7: Работа с отладчиком

Шаг 4

Запускаю программу до первой точки останова, и проверяю значения регистров.

- Замечаю, что результат сложения записывается в регистр ebx. (рис. [3.8]).

```

Register group: general
eax      0x2          2          ecx      0x0          0
edx      0x0          0          ebx      0x5          5
esp      0xffffd1b0   0xffffd1b0   ebp      0x0          0x0
esi      0x0          0          edi      0x0          0
eip      0x80490f4    0x80490f4 <_start+12> eflags   0x206        [ PF IF ]
cs       0x23         35          ss       0x2b         43
ds       0x2b         43          es       0x2b         43
fs       0x0          0          gs       0x0          0

0x80490e8 <_start>      mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
B+> 0x80490f4 <_start+12> mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
b+ 0x8049100 <_start+24> mov     eax,0x804a000
0x8049105 <_start+29>   call    0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
0x804910c <_start+36>   call    0x8049086 <printf>
0x8049111 <_start+41>   call    0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al
0x8049118             add     BYTE PTR [eax],al
0x804911a             add     BYTE PTR [eax],al
0x804911c             add     BYTE PTR [eax],al
0x804911e             add     BYTE PTR [eax],al

native process 6762 In: _start L11 PC: 0x80490f4
(gdb) layout regs
(gdb) run
Starting program: /home/darfonos/work/arch-pc/lab09/lab9-5

Breakpoint 1, _start () at lab9-5.asm:11

```

Рис. 3.8: Проверка значений регистров

Шаг 5

Перехожу к следующему брейкпоинту и снова проверяю какие значения принимают регистры. (рис. [3.10]).

- Замечаю, что умножение регистра ebx происходит на регистр eax(4*2), а к регистру ebx плюсуется 5 (5+5) и его значение записывается в результат программы.

```

Register group: general
eax      0x8          8          ecx      0x4          4
edx      0x0          0          ebx      0xa         10
esp      0xffffd1b0   0xffffd1b0   ebp      0x0          0x0
esi      0x0          0          edi      0xa         10

```

Рис. 3.9: Выявление главных ошибок

```

Breakpoint 2, _start () at lab9-5.asm:16
(gdb) p/s $ebx
$1 = 10
(gdb) p/s $edi
$2 = 10
(gdb) p/s $eax
$3 = 8
(gdb)

```

Рис. 3.10: Выявление главных ошибок

Шаг 6

Исправляю основные ошибки выявленные с помощью отладчика GDB. (рис. [3.11]).

```

; ---- 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

```

Рис. 3.11: Исправление ошибок в программе

Шаг 7

Создаю исполняемый файл и проверяю работу программы. (рис. [3.12]).

```

[oochernyatjeva@fedora lab09]$ nasm -f elf -g -l lab9-5.lst lab9-5.asm
[oochernyatjeva@fedora lab09]$ ld -m elf_i386 -o lab9-5 lab9-5.o
[oochernyatjeva@fedora lab09]$ ./lab9-5
Результат: 25
[oochernyatjeva@fedora lab09]$

```

Рис. 3.12: Запуск исполняемого файла

Программа отработала без ошибок!!

4 Вывод

В результате выполнения лабораторной работы, я научилась организовывать код в подпрограммы и познакомилась с базовыми функциями отладчика GDB.