

Отчёт по лабораторной работе №9

Архитектура компьютера НММбд-03-24

Туева Анастасия Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	22
5	Выводы	28

Список иллюстраций

3.1	Создание файла	7
3.2	Редактирование файла	8
3.3	Запуск исполняемого файла	9
3.4	Редактирование файла	10
3.5	Запуск исполняемого файла	11
3.6	Редактирование файла lab09-2.asm	12
3.7	Исполнение программы	13
3.8	Исполнение программы брейкпойнт	13
3.9	Просмотр дисассимилированного кода программы	14
3.10	Просмотр дисассимилированного кода программы с синтаксисом Intel	15
3.11	Переход в режим псевдографики	16
3.12	Наличие меток	17
3.13	Значение переменной msg1	17
3.14	Значение переменной msg2	18
3.15	Изменение значения переменной msg1	18
3.16	Изменение значения переменной msg2	18
3.17	Значение регистра edx	19
3.18	Загрузка файла lab09-3.asm в отладчик	20
3.19	Запуск файла lab09-3 через метку	20
3.20	Адрес вершины стека	20
3.21	Проверка остальных позиций стека	21
4.1	Текст программы lab09-4.asm	23
4.2	Запуск программы	24
4.3	Текст программы	25
4.4	Запуск программы	26
4.5	Запуск файла lab09-3 через метку	26
4.6	Повторный запуск программы	27

Список таблиц

1 Цель работы

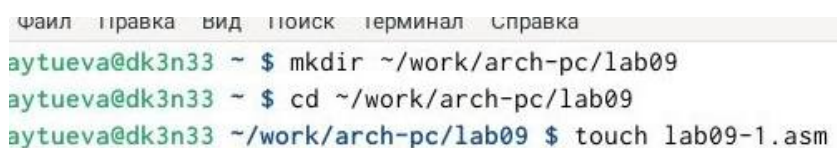
Приобретение навыков написания программ с использованием подпрограмм.
Знакомство с методами отладки при помощи GDB и его основными возможностями

2 Задание

1. Выполнение лабораторной работы
2. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.
3. В листинге 9.3 приведена программа вычисления выражения $(3 + 2) \cdot 4 + 5$. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

3 Выполнение лабораторной работы

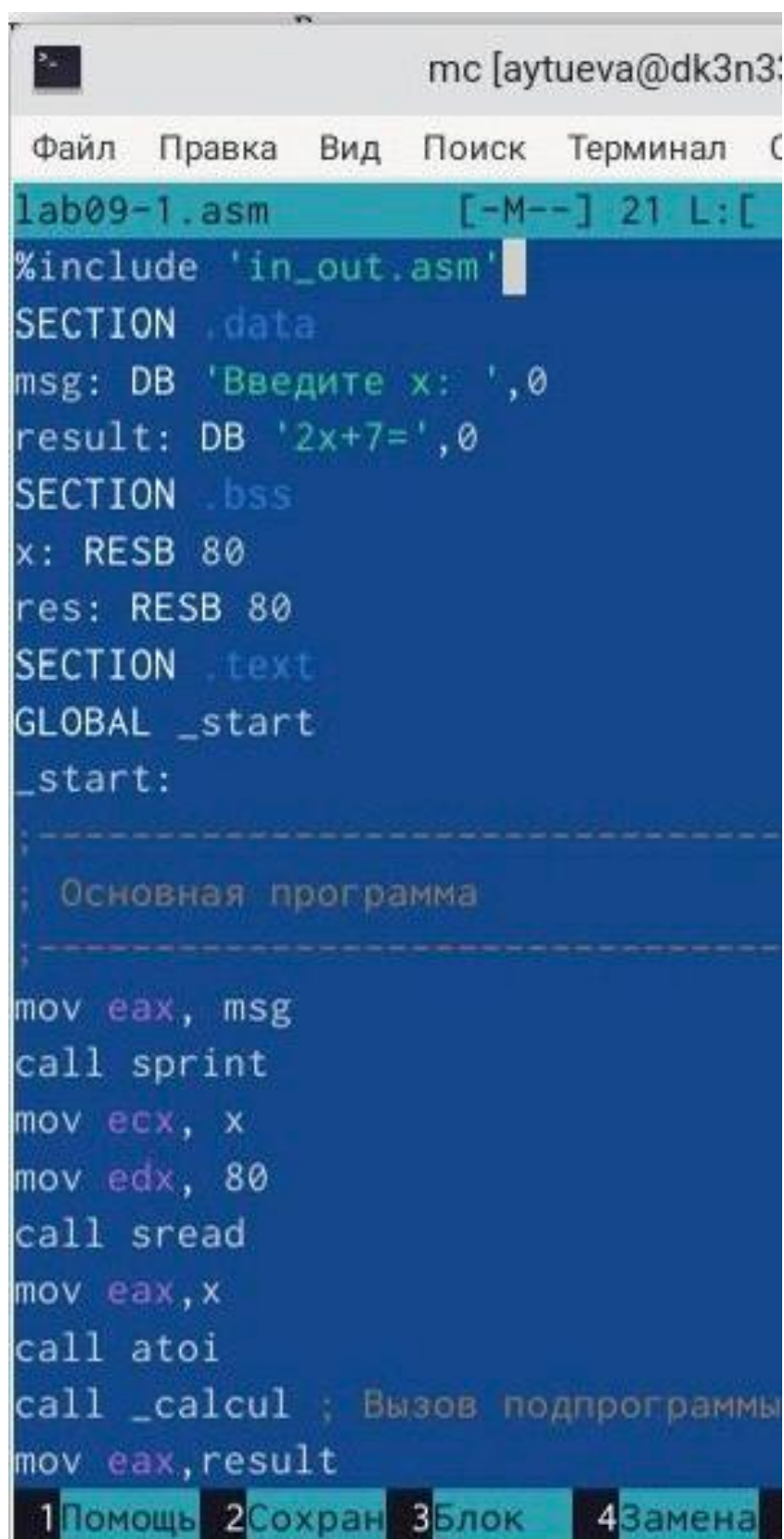
Создаём каталог для программ лабораторной работы №9, переходим в него и создаём файл lab09-1.asm. (рис. 3.1).



```
Файл  Правка  Вид  Поиск  Терминал  Справка
aytueva@dk3n33 ~ $ mkdir ~/work/arch-pc/lab09
aytueva@dk3n33 ~ $ cd ~/work/arch-pc/lab09
aytueva@dk3n33 ~/work/arch-pc/lab09 $ touch lab09-1.asm
```

Рис. 3.1: Создание файла

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. Вводим в файл `lab09-1.asm` текст программы из листинга 9.1. (рис. 3.2).



The screenshot shows a text editor window with a menu bar (Файл, Правка, Вид, Поиск, Терминал) and a title bar (mc [aytueva@dk3n3:]). The editor is open to a file named 'lab09-1.asm'. The code is written in assembly language and includes comments in Russian. The code defines data and bss sections, then uses system calls to read input and calculate the result.

```
lab09-1.asm [-M--] 21 L:[
#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
mov eax, result
```

At the bottom of the window, there is a status bar with four buttons: 1Помощь, 2Сохран, 3Блок, and 4Замена.

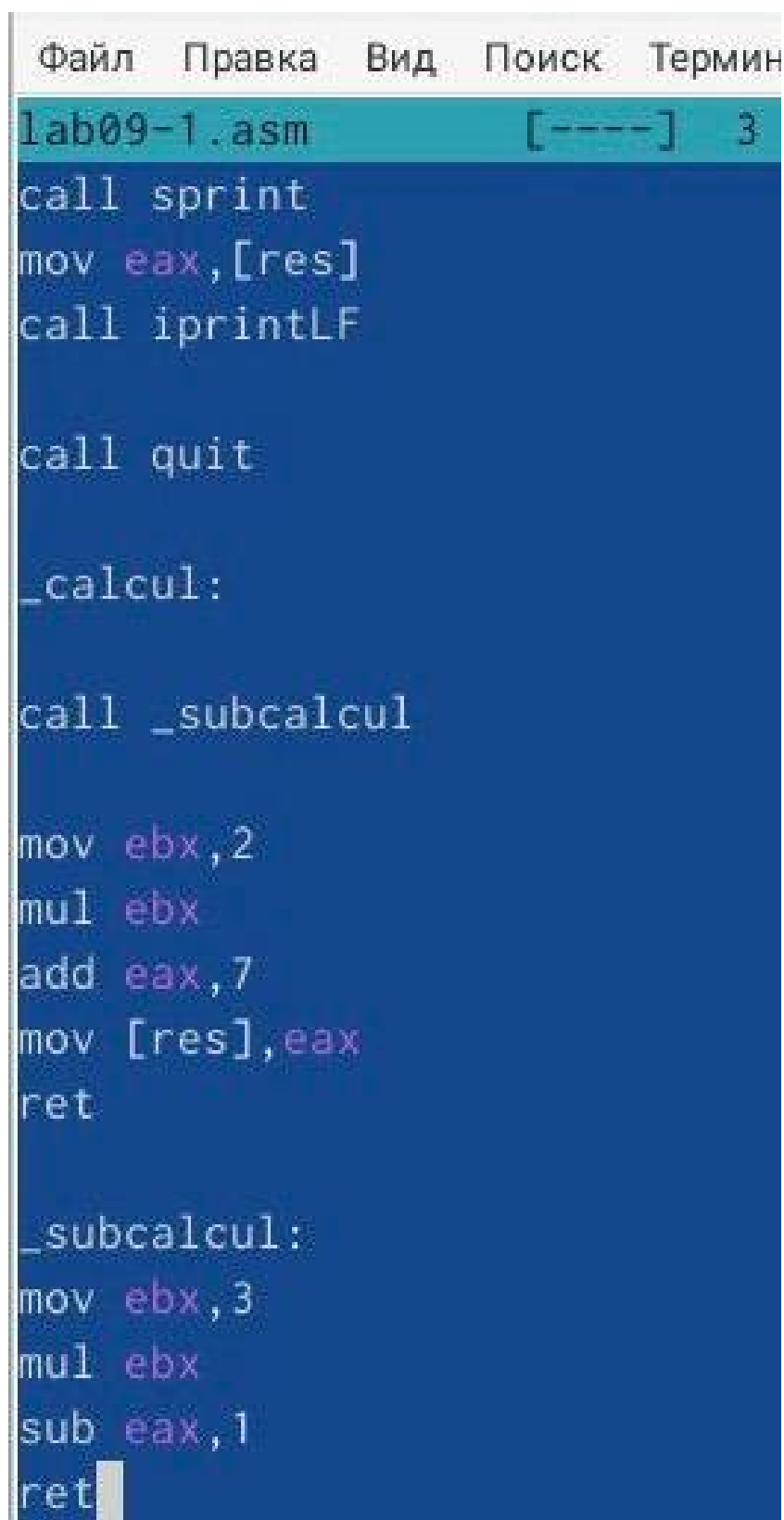
Рис. 3.2: Редактирование файла

Создаем исполняемый файл программы и запускаем его (рис. 3.3).

```
aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
2x+7=13
```

Рис. 3.3: Запуск исполняемого файла

Изменим текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. (рис. 3.4).



```
Файл  Правка  Вид  Поиск  Термин
lab09-1.asm  [-----]  3
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
```

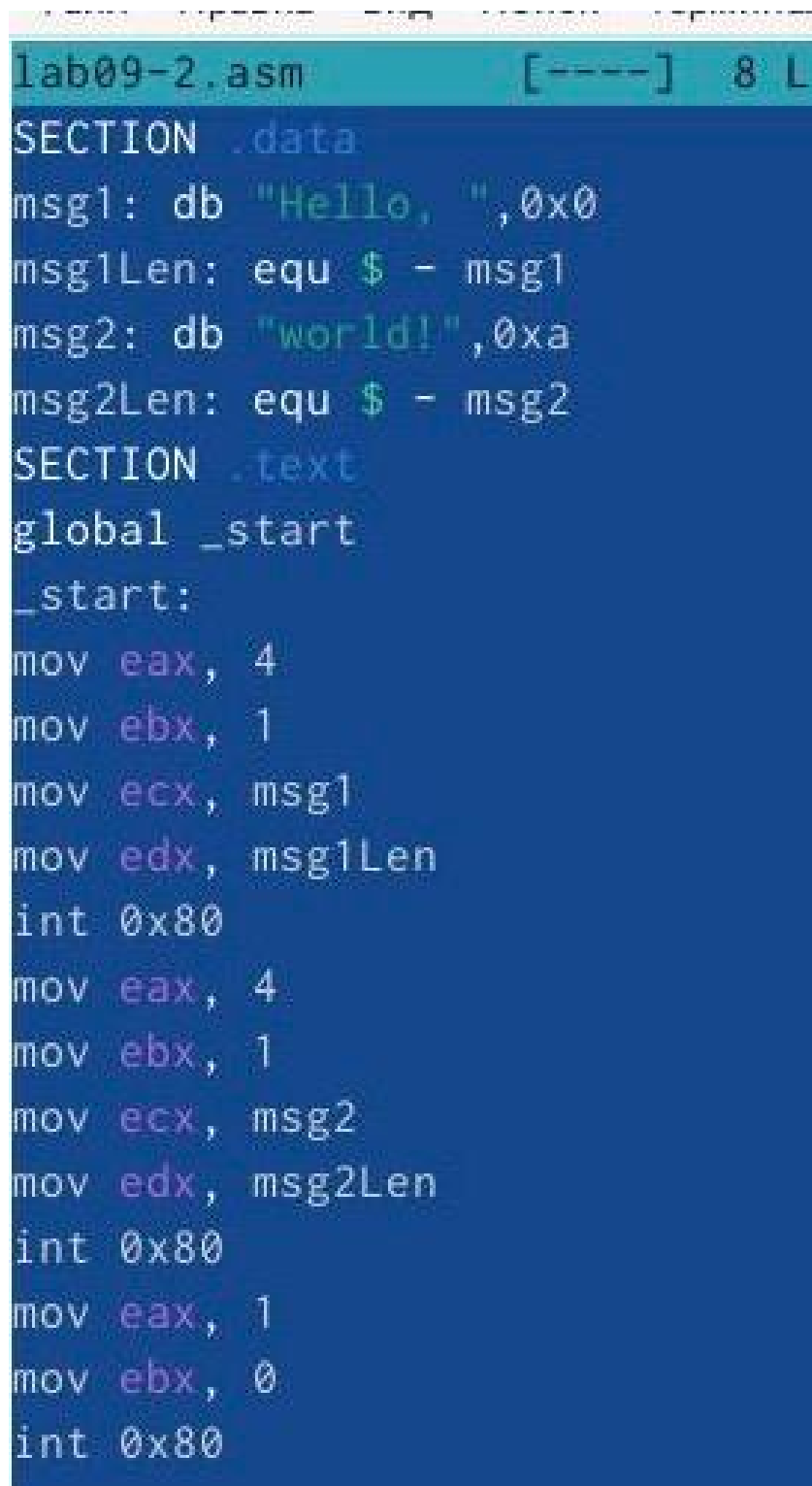
Рис. 3.4: Редактирование файла

Создаем новый исполняемый файл программы и запускаем его (рис. 3.5).

```
aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 3
f(g(x))= 23
```

Рис. 3.5: Запуск исполняемого файла

Создаем файл lab09-2.asm. Вводим в него программу из листинга 9.2. (рис. 3.6).



```
lab09-2.asm      [-----]  8 L
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80
mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 3.6: Редактирование файла lab09-2.asm

Транслируем текст программы с ключом '-g'. Загружаем исполняемый файл в

gdb (рис. 3.7).

```
aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 10458) exited normally]
(gdb)
```

Рис. 3.7: Исполнение программы

Для более подробного анализа программы установим брейкпойнт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустим её. (рис. 3.8).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/lab09-2

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

Рис. 3.8: Исполнение программы брейкпойнт

Посмотрим дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`. (рис. 3.9).

```

(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. 3.9: Просмотр дисассимилированного кода программы

Переключимся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. Различия между синтаксисом ATТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как `ax`, `eax`, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов). (рис. 3.10).

```

End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
      0x08049005 <+5>:      mov     ebx,0x1
      0x0804900a <+10>:     mov     ecx,0x804a000
      0x0804900f <+15>:     mov     edx,0x8
      0x08049014 <+20>:     int     0x80
      0x08049016 <+22>:     mov     eax,0x4
      0x0804901b <+27>:     mov     ebx,0x1
      0x08049020 <+32>:     mov     ecx,0x804a008
      0x08049025 <+37>:     mov     edx,0x7
      0x0804902a <+42>:     int     0x80
      0x0804902c <+44>:     mov     eax,0x1
      0x08049031 <+49>:     mov     ebx,0x0
      0x08049036 <+54>:     int     0x80
End of assembler dump.

```

Рис. 3.10: Просмотр дисассимилированного кода программы с синтаксисом Intel

Включим режим псевдографики для более удобного анализа программы. (рис. 3.11).

```
aytueva@dk3n33 - lab09
Файл  Правка  Вид  Поиск  Терминал  Справка

[ Register Values Unavailable ]

B+> 0x8049000 <_start>  mov  eax,0x4
      0x8049005 <_start+5>  mov  ebx,0x1
      0x804900a <_start+10> mov  ecx,0x804a000
      0x804900f <_start+15> mov  edx,0x8
      0x8049014 <_start+20> int  0x80
      0x8049016 <_start+22> mov  eax,0x4
      0x804901b <_start+27> mov  ebx,0x1
      0x8049020 <_start+32> mov  ecx,0x804a008

native process 10533 In: _start  L9
(gdb) layout regs
(gdb)
```

Рис. 3.11: Переход в режим псевдографики

Посмотрим наличие меток и добавим еще одну метку на предпоследнюю инструкцию. (рис. 3.12).


```

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

native process 10533 In: _start L9
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) 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. 3.12: Наличие меток

С помощью команды посмотрим значение переменной msg1. (рис. 3.13).

```

    0x8049036 <_start+54>  int      0:
native process 11691 In: _start
(gdb) x/1sb &msg1
0x804a000 <msg1>:         "Hello, "
(gdb)

```

Рис. 3.13: Значение переменной msg1

Посмотрим значение второй переменной msg2. (рис. 3.14).

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

Рис. 3.14: Значение переменной msg2

С помощью команды set изменим значение переменной msg1. (рис. 3.15).

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hh1lo, "
(gdb)
```

Рис. 3.15: Изменение значения переменной msg1

Изменим переменную msg2. (рис. 3.16).

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

Рис. 3.16: Изменение значения переменной msg2

Выведем в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.(рис. 3.17).

```

Register group: general—
eax      0x8
ecx      0x804a000
edx      0x8
ebx      0x1
esp      0xffffc4c0
ebp      0x0
esi      0x0

0x804900a <_start+10>
0x804900f <_start+15>
0x8049014 <_start+20>
>0x8049016 <_start+22>
0x804901b <_start+27>
0x8049020 <_start+32>
0x8049025 <_start+37>

native process 4097 In: _st
(gdb) p/s $edx
$1 = 8
(gdb) p/t $edx
$2 = 1000
(gdb) p/x $edx
$3 = 0x8
(gdb) █

```

Рис. 3.17: Значение регистра edx

Скопируем файл lab8-2.asm в файл с именем lab09-3.asm. Создадим исполняемый файл. Загрузим исполняемый файл в отладчик, указав аргументы. (рис. 3.18).

```
aytueva@dk3n33 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'а
ргумент 3'
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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)
```

Рис. 3.18: Загрузка файла lab09-3.asm в отладчик

Поставим метку на _start и запустим файл. (рис. 3.19).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/la
b09-3 аргумент1 аргумент 2 аргумент\ 3

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

Рис. 3.19: Запуск файла lab09-3 через метку

Проверим адрес вершины стека и убедимся, что там хранится 5 элементов. (рис. 3.20).

```
(gdb) x/x $esp
0xfffffc4a0:      0x00000005
(gdb)
```

Рис. 3.20: Адрес вершины стека

Посмотрим 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. Шаг изменения адреса равен 4 байтам, потому что мы работаем с 32-битной системой (x86), а указатели (void **) в такой системе занимают 4 байта. Ошибка Cannot access memory at address 0x0 на \$esp + 24 указывает на то, что закончились аргументы командной строки. (рис. 3.21).

```
(gdb) x/s *(void**)(esp + 4)
0xfffffc70d:      "/afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/lab
09-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc751:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc763:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc774:      "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc776:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 3.21: Проверка остальных позиций стека

4 Выполнение самостоятельной работы

Преобразем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.(рис. 4.1).

```
lab09-4.asm [----] 2
%include 'in_out.asm'

SECTION .data
prim DB 'f(x)=15x+2',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end

pop eax
call atoi
call fir
add esi,eax

loop next
```

Рис. 4.1: Текст программы lab09-4.asm

```

aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-4
f(x)=15x+2
Результат: 0
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-4 1 2
f(x)=15x+2
Результат: 49
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-4 1 2 4
f(x)=15x+2
Результат: 111
aytueva@dk3n33 ~/work/arch-pc/lab09 $

```

Рис. 4.2: Запуск программы

Перепишем программу и попробуем запустить ее, чтобы увидеть ошибку. Ошибка арифметическая, так как вместо 25, программа выводит 10. (рис. 4.3).


```
lab09-5.asm      [----]  9 L:[  1+2
%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,div
call sprint
mov eax,edi
call iprintLF

call quit
```

Рис. 4.3: Текст программы

```

aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ./lab09-5
Результат: 10
aytueva@dk3n33 ~/work/arch-pc/lab09 $

```

Рис. 4.4: Запуск программы

После появления ошибки, я запустила программу в отладчике. (рис. 4.5).

```

aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) b _start
Breakpoint 1 at 0x080490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/lab09-5

Breakpoint 1, 0x080490e8 in _start ()
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:    mov     ebx,0x3
      0x080490ed <+5>:    mov     eax,0x2
      0x080490f2 <+10>:   add     ebx,eax
      0x080490f4 <+12>:   mov     ecx,0x4
      0x080490f9 <+17>:   mul     ecx
      0x080490fb <+19>:   add     ebx,0x5
      0x080490fe <+22>:   mov     edi,ebx
      0x08049100 <+24>:   mov     eax,0x804a000
      0x08049105 <+29>:   call   0x804900f <sprint>
      0x0804910a <+34>:   mov     eax,edi
      0x0804910c <+36>:   call   0x8049086 <iprintf>
      0x08049111 <+41>:   call   0x80490db <quit>
End of assembler dump.
(gdb)

```

Рис. 4.5: Запуск файла lab09-3 через метку

Откроем регистры и проанализируем их. Некоторые регистры стоят не на своих местах. Исправим это. Изменим регистры и запустим программу. Программа вывела ответ 25, то есть все работает правильно.(рис. 4.6).

```

aytueva@dk3n33 ~/work/arch-pc/lab09 $ nasm -f elf lab09-5.asm
aytueva@dk3n33 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-5 lab09-5.o
aytueva@dk3n33 ~/work/arch-pc/lab09 $ gdb lab09-5
GNU gdb (Gentoo 14.2 vanilla) 14.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aytueva/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 17979) exited normally]
(gdb)

```

Рис. 4.6: Повторный запуск программы

5 Выводы

В результате выполнения лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки в NASM.