

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Кириянова Екатерина Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Реализация подпрограмм в NASM	8
4.2	Отладка программ с помощью GDB	9
4.3	Задание для самостоятельной работы	20
5	Вывод	24

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Текст программы	8
4.3	Работа программы	9
4.4	Измененный текст программы	9
4.5	Проверка работы программы	9
4.6	Текст второй программы	10
4.7	Отладка второго файла	10
4.8	Брекпоинт на метку _start	11
4.9	Дисассимплированный код	11
4.10	Intel’овское отображение	12
4.11	Псевдографика	12
4.12	Наличие меток	13
4.13	Просмотр регистров	13
4.14	Просмотри значения переменной	14
4.15	Значение переменной msg2	15
4.16	Изменение значения переменной	15
4.17	Изменение msg2	16
4.18	Значение регистров еsx и еax	17
4.19	Значение регистров ebx	18
4.20	Копирование	18
4.21	Запуск файла в отладчике	18
4.22	Запуск файла lab9-3 через метку	19
4.23	Адрес вершины стека	19
4.24	Все позиции стека	19
4.25	Текст программы	20
4.26	Запуск программы	20
4.27	Текст программы	21
4.28	Запуск программы	21
4.29	Запуск программы в отладчике	22
4.30	Запуск	22
4.31	Анализ регистров	23
4.32	Повторный запуск программы	23

1 Цель работы

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

2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Задание для самостоятельной работы

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: обнаружение ошибки; поиск её местонахождения; определение причины ошибки; исправление ошибки. Можно выделить следующие типы ошибок: синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново. Наиболее часто применяют следующие методы отладки: создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения); использование специальных программ-отладчиков. Отладчики позволяют управлять ходом выполнения

программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программаотладчик приостанавливает выполнение программы и ждёт команд. GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки.

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

Создаю каталог lab09 и файл lab9-1.asm (рис. 4.1).

```
eakirjyanova@user:~$ mkdir ~/work/arch-pc/lab09
eakirjyanova@user:~$ cd ~/work/arch-pc/lab09
eakirjyanova@user:~/work/arch-pc/lab09$ touch lab9-1.asm
```

Рис. 4.1: Создание каталога и файла

Ввожу текст листинга в файл (рис. 4.2) и запускаю программу (рис. 4.3).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab09/lab9-1.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
```

Рис. 4.2: Текст программы


```
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
eakirjyanova@user:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 5
2x+7=17
```

Рис. 4.3: Работа программы

Меняю текст программы (рис. 4.4), чтобы она решала выражение $f(g(x))$ (рис. 4.5).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab09/lab9-1.asm *
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
prim1: DB 'f(x) = 2x+7',0
prim2: DB 'g(x) = 3x-1',0
result: DB 'f(g(x))= ',0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,prim1
call sprintLF

mov eax,prim2
call sprintLF
```

Рис. 4.4: Измененный текст программы

```
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
eakirjyanova@user:~/work/arch-pc/lab09$ ./lab9-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 5
f(g(x))= 35
```

Рис. 4.5: Проверка работы программы

4.2 Отладка программ с помощью GDB

Создаю файл lab9-2.asm и ввожу туда программу (рис. 4.6).

```
eakirjyanova@user: ~/work/arch-pc/lab09
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab09/lab9-2.asm *
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. 4.6: Текст второй программы

Запускаю файл второй программы в отладчик gdb (рис. 4.7).

```
eakirjyanova@user: ~/work/arch-pc/lab09
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
eakirjyanova@user:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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) run
Starting program: /home/eakirjyanova/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 5486) exited normally]
(gdb)
```

Рис. 4.7: Отладка второго файла

Ставлю брекпоинт на метку `_start` и запускаю программу (рис. 4.8).

```
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/eakirjyanova/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
```

Рис. 4.8: Брекпоинт на метку `_start`

Просматриваю дисассимплированный код программы начиная с метки (рис. 4.9).

```
(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.
```

Рис. 4.9: Дисассимплированный код

С помощью команды переключаюсь на intel'овское отображение синтаксиса (рис. 4.10). Отличие заключается в командах, в дисассимилированном отображении в командах используют `%` и `$`, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

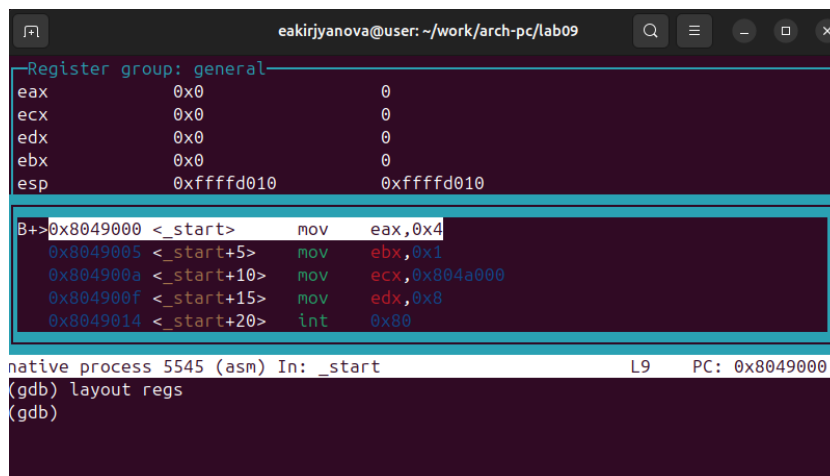
```

(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. 4.10: Intel'овское отображение

Включаю режим псевдографики (рис. 4.11).



```

eakiryanova@user: ~/work/arch-pc/lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010

B+>0x08049000 <_start>  mov     eax,0x4
0x08049005 <_start+5>  mov     ebx,0x1
0x0804900a <_start+10> mov     ecx,0x804a000
0x0804900f <_start+15> mov     edx,0x8
0x08049014 <_start+20> int     0x80

native process 5545 (asm) In: _start L9 PC: 0x08049000
(gdb) layout regs
(gdb)

```

Рис. 4.11: Псевдографика

Смотрю наличие меток и добавляю еще одну метку на предпоследнюю инструкцию (рис. 4.12).

```

eakirjyanova@user: ~/work/arch-pc/lab09

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0

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

native process 5545 (asm) In: _start L9 PC: 0x804900
(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
          breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:20
(gdb)

```

Рис. 4.12: Наличие меток

С помощью команды si смотрю регистры (рис. 4.13).

```

eakirjyanova@user: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

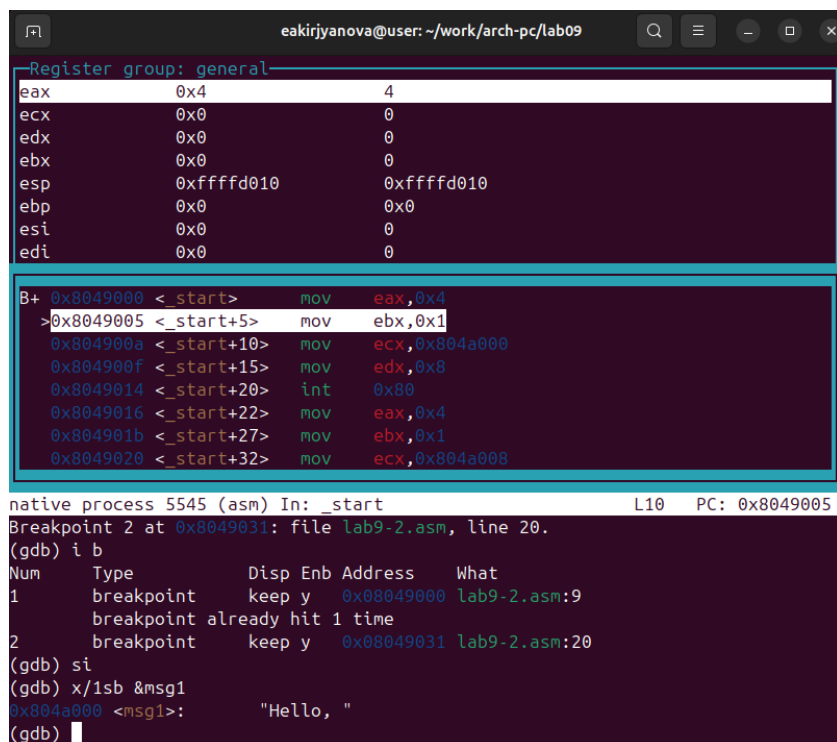
B+ 0x8049000 <_start>    mov     eax,0x4
>0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int     0x80
0x8049016 <_start+22>   mov     eax,0x4
0x804901b <_start+27>   mov     ebx,0x1
0x8049020 <_start+32>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

native process 5545 (asm) In: _start L10 PC: 0x8049005
(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
          breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:20
(gdb) si
(gdb)

```

Рис. 4.13: Просмотр регистров

С помощью команды я смотрю значение переменной msg1 (рис. 4.14).



The screenshot shows a GDB debugger window with the following content:

```
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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 5545 (asm) In: _start L10 PC: 0x8049005
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
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab9-2.asm:20
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)
```

Рис. 4.14: Просмотри значения переменной

Смотрю значение второй переменной msg2 (рис. 4.15).

```

eakiryanova@user: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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 5545 (asm) In: _start L10 PC: 0x80490
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab9-2.asm:20
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"

```

Рис. 4.15: Значение переменной msg2

С помощью команды set меняю значение переменной msg1 (рис. 4.16).

```

eakiryanova@user: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

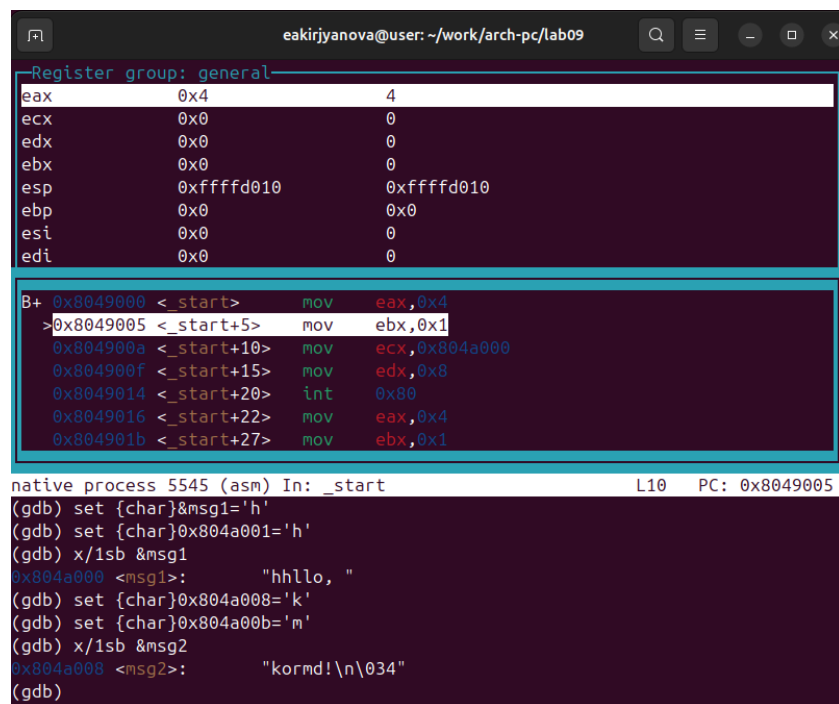
B+ 0x8049000 <_start> mov eax,0x4
>0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1

native process 5545 (asm) In: _start L10 PC: 0x8049005
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hhlllo, "
(gdb)

```

Рис. 4.16: Изменение значения переменной

Меняю переменную msg2 (рис. 4.17).



```
eakirjyanova@user: ~/work/arch-pc/lab09
Register group: general
eax    0x4      4
ecx    0x0      0
edx    0x0      0
ebx    0x0      0
esp    0xffffd010 0xffffd010
ebp    0x0      0x0
esi    0x0      0
edi    0x0      0

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

native process 5545 (asm) In: _start      L10    PC: 0x8049005
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hhlllo, "
(gdb) set {char}0x804a008='k'
(gdb) set {char}0x804a00b='m'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "kormd!\n\034"
(gdb)
```

Рис. 4.17: Изменение msg2

Вывожу значение регистров esx и eax (рис. 4.18).


```
eakirjyanova@user: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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 5545 (asm) In: _start L10 PC: 0x8049005
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)
```

Рис. 4.18: Значение регистров ecx и eax

Меняю значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные (рис. 4.19).

```
eakirjyanova@user: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x2      2
esp      0xffffd010 0xffffd010
ebp      0x0      0x0

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

native process 5545 (asm) In: _start L10 PC: 0x8049005
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb) █
```

Рис. 4.19: Значение регистров ebx

Копирую файл lab8-2.asm и переименовываю его (рис. 4.20). Запускаю файл в отладчике, указав аргументы (рис. 4.21).

```
eakirjyanova@user:~$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab
9-3.asm
eakirjyanova@user:~$ █
```

Рис. 4.20: Копирование

```
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
eakirjyanova@user:~/work/arch-pc/lab09$ gdb --args lab9-3 аргумент1 аргумент 2 'аргумент
3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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) █
```

Рис. 4.21: Запуск файла в отладчике

Ставлю метку на _start и запускаю файл (рис. 4.22).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/eakirjyanova/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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 lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в ecx количество
(gdb)

```

Рис. 4.22: Запуск файла lab9-3 через метку

Проверяю адрес вершины стека и убеждаюсь, что там хранится 5 элементов (рис. 4.23).

```

(gdb) x/x $esp
0xffffcfa0:      0x00000005
(gdb)

```

Рис. 4.23: Адрес вершины стека

Смотрю все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации (рис. 4.24).

```

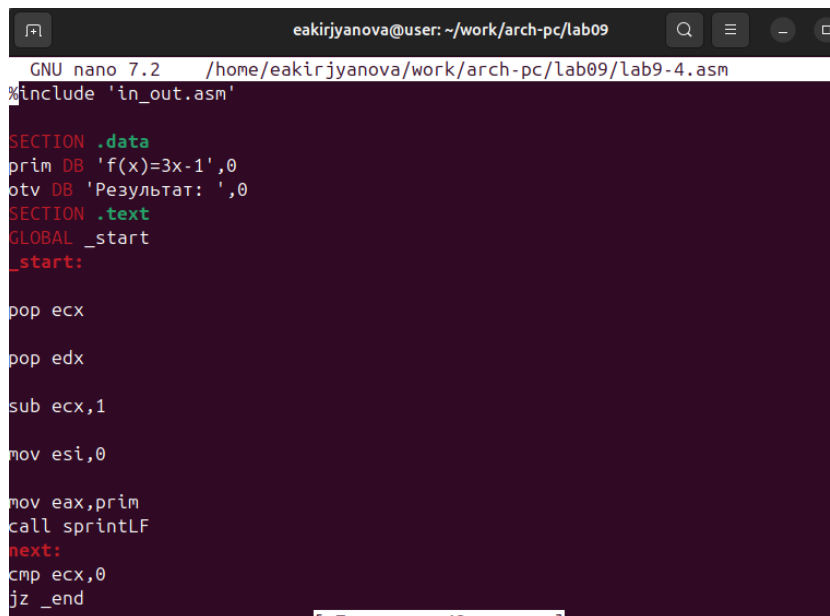
(gdb) x/s *(void**)($esp + 4)
0xfffffd174:      "/home/eakirjyanova/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xfffffd1a1:      "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xfffffd1b3:      "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xfffffd1c4:      "2"
(gdb) x/s *(void**)($esp + 20)
0xfffffd1c6:      "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:      <error: Cannot access memory at address 0x0>

```

Рис. 4.24: Все позиции стека

4.3 Задание для самостоятельной работы

Преобразовываю программу из лабораторной работы №8 (рис. 4.25) и реализую вычисления как подпрограмму (рис. 4.26).



```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab09/lab9-4.asm
%include 'in_out.asm'

SECTION .data
prim DB 'f(x)=3x-1',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

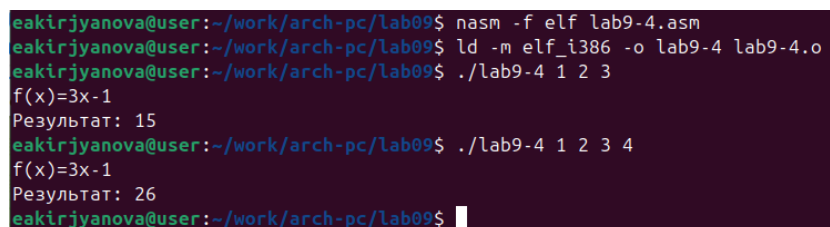
    pop ecx
    pop edx

    sub ecx,1

    mov esi,0

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

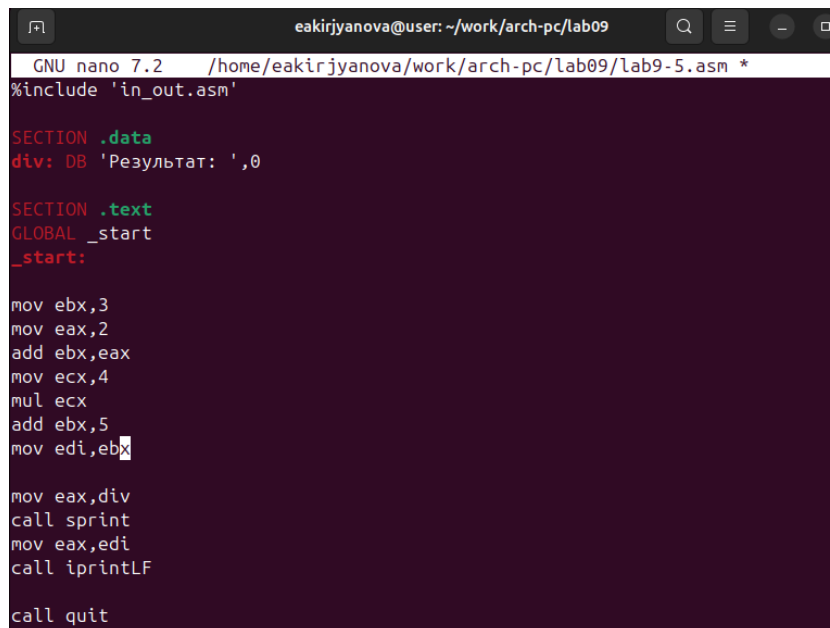
Рис. 4.25: Текст программы



```
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
eakirjyanova@user:~/work/arch-pc/lab09$ ./lab9-4 1 2 3
f(x)=3x-1
Результат: 15
eakirjyanova@user:~/work/arch-pc/lab09$ ./lab9-4 1 2 3 4
f(x)=3x-1
Результат: 26
eakirjyanova@user:~/work/arch-pc/lab09$
```

Рис. 4.26: Запуск программы

Переписываю программу (рис. 4.27) и пробую запустить ее чтобы увидеть ошибку. Ошибка арифметическая (рис. 4.28).



```
eakirjyanova@user: ~/work/arch-pc/lab09
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab09/lab9-5.asm *
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

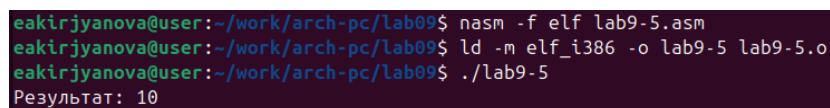
SECTION .text
GLOBAL _start
_start:

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. 4.27: Текст программы



```
eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
eakirjyanova@user:~/work/arch-pc/lab09$ ./lab9-5
Результат: 10
```

Рис. 4.28: Запуск программы

После появления ошибки, запускаю программу в отладчике (рис. 4.29).

```
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab9-5.o
eakirjyanova@user:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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 lab09-5...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-5.asm, line 8.
(gdb) r
Starting program: /home/eakirjyanova/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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 lab9-5.asm:8
8      mov     ebx,3
(gdb)
```

Рис. 4.29: Запуск программы в отладчике

```
(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.
```

Рис. 4.30: Запуск

Открываю регистры и анализирую их. Некоторые регистры стоят не на своих местах, исправляю это (рис. 4.31).

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd010 0xffffd010
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

B+>0x80490e8 <_start> mov ebx,0x3
0x80490ed <_start+5> mov eax,0x2
0x80490f2 <_start+10> add ebx,eax
0x80490f4 <_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>

```

Рис. 4.31: Анализ регистров

Меняю регистры и запускаю программу, программа работает корректно (рис. 4.32).

```

eakirjyanova@user:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-5.lst lab9-5.asm
eakirjyanova@user:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
eakirjyanova@user:~/work/arch-pc/lab09$ gdb lab9-5
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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) r
Starting program: /home/eakirjyanova/work/arch-pc/lab09/lab9-5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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: 25
[Inferior 1 (process 10297) exited normally]

```

Рис. 4.32: Повторный запуск программы

5 Вывод

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.