

Отчёт по лабораторной работе

Дисциплина: Архитектура компьютера

Кижваткина Анна Юрьевна

Содержание

1	Цель работы	6
2	Задание	7
3	Выполнение лабораторной работы	8
4	Выводы	19

Список иллюстраций

3.1	Создание каталога	8
3.2	Перемещение в каталог	8
3.3	Создание файла и проверка наличия	8
3.4	Ввод программы	8
3.5	Создание исполняемого файла и его запуск	9
3.6	Изменение программы	9
3.7	Создание исполняемого файла и его запуск	9
3.8	Создание файла и проверка наличия	9
3.9	Запись программы	10
3.10	Создание исполняемого файла	10
3.11	Запуск программы в отладчик	10
3.12	Команда run	10
3.13	Создание брейка	10
3.14	Команда disassemble _start	11
3.15	Команда disassembly-flavor intel	11
3.16	Режим псевдографики	11
3.17	Проверка меток	12
3.18	Добавление метки	12
3.19	Проверка меток	12
3.20	Команда si	12
3.21	Команда i r	13
3.22	Просмотр значения переменной	13
3.23	Просмотр значения переменной	13
3.24	Команда set	13
3.25	Изменение msg2	13
3.26	Вывод значений	14
3.27	Изменение значений регистра	14
3.28	Команда sr	14
3.29	Создание исполняемого файла	14
3.30	Запуск файла	14
3.31	Создание метки	15
3.32	Проверка количества элементов	15
3.33	Проверка позиций стека	15
3.34	Изменение программы	16
3.35	Создание исполняемого файла	16
3.36	Запуск файла	16
3.37	Создание файла	16

3.38 Программа в файле	17
3.39 Создание исполняемого файла и запуск	17
3.40 Запуск программы в отладчике	17
3.41 Анализ регистров	18
3.42 Создание исполняемого файла и запуск	18

Список таблиц

1 Цель работы

Целью данной лабораторной работы является 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. 2.1. Добавление точек останова. 2.2. Работа с данными программы в GDB. 2.3. Обработка аргументов командной строки в GDB.
3. Выполнение самостоятельной работы.

3 Выполнение лабораторной работы

Создаем каталог. (рис. 3.1).

```
aykizhvatkina@dk8n75 ~ $ mkdir ~/work/arch-pc/lab09
```

Рис. 3.1: Создание каталога

Перемещаемся в созданный каталог. (рис. 3.2).

```
aykizhvatkina@dk8n75 ~ $ cd ~/work/arch-pc/lab09
```

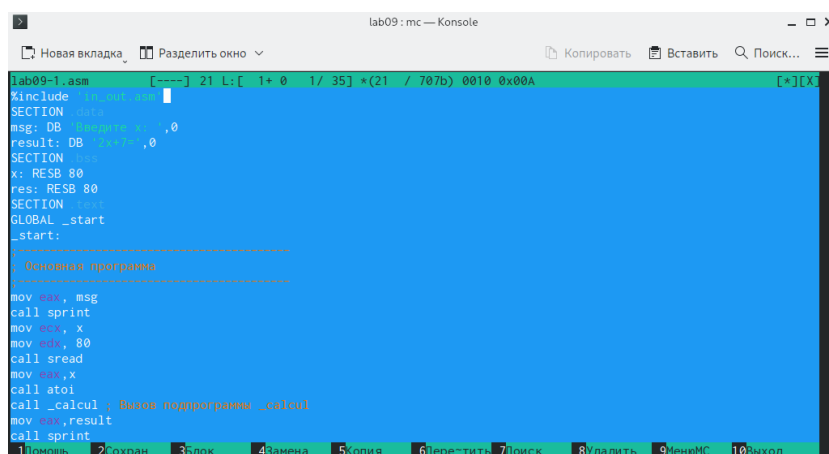
Рис. 3.2: Перемещение в каталог

Создаем файл и проверяем его наличие. (рис. 3.3).

```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ touch lab09-1.asm
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ls
lab09-1.asm
```

Рис. 3.3: Создание файла и проверка наличия

Вводим текст программы листинга. (рис. 3.4).



```
lab09-1.asm [---] 21 L: [ 1+ 0 1/ 35] *(21 / 707b) 0010 0x00A [*][X]
%include "lab09-1.asm"
SECTION .text
msg: DB "lab09-1.asm",0
result: DB "result",0
SECTION .data
x: RESB 80
res: RESB 80
SECTION .global
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
```

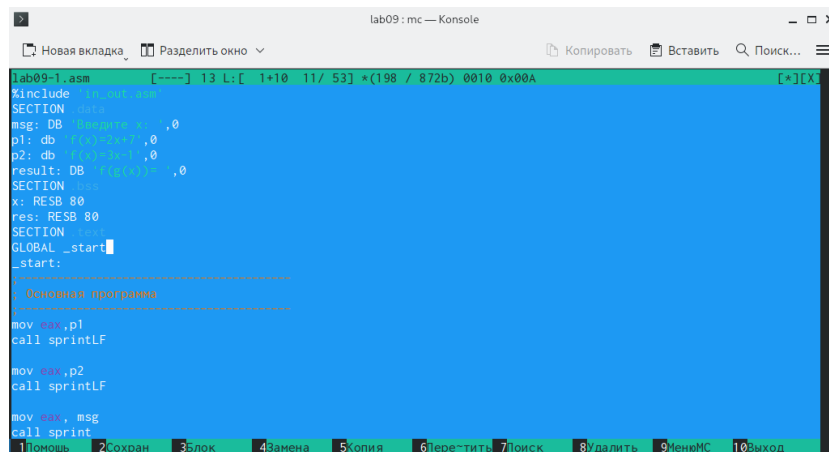
Рис. 3.4: Ввод программы

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

```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 5
2x+7=17
```

Рис. 3.5: Создание исполняемого файла и его запуск

Изменяем текст программы так, чтобы она решала $f(g(x))$. (рис. 3.6).



```
lab09-1.asm  [---] 13 L: [ 1+10 11/ 53] *(198 / 872b) 0010 0x00A [*][X]
%include "lab09-1.inc"
SECTION .data
msg: DB "Введите x: ",0
p1: db "f(x)=2x+7",0
p2: db "f(x)=3x-1",0
result: DB "f(g(x))=",0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
; =====
; Основная программа
; =====
mov eax, p1
call printf

mov eax, p2
call printf

mov eax, msg
call printf
```

Рис. 3.6: Изменение программы

Создаем исполняемый файл и запускаем его. (рис. 3.7).

```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ./lab09-1
f(x)=2x+7
f(x)=3x-1
Введите x: 5
f(g(x))= 35
```

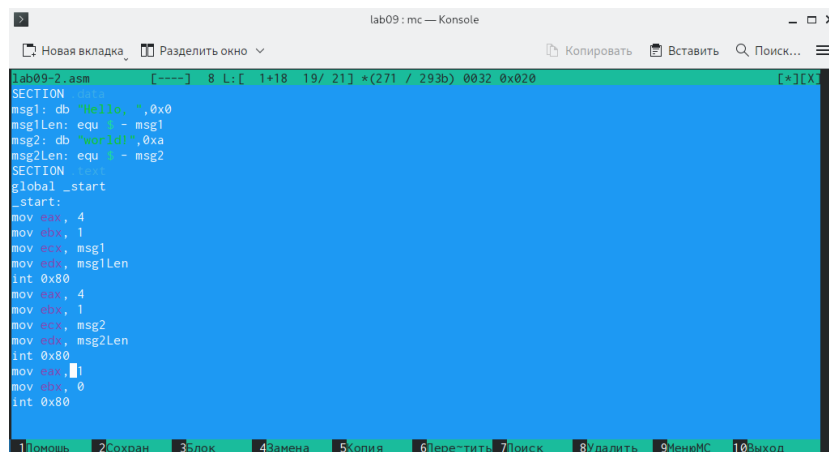
Рис. 3.7: Создание исполняемого файла и его запуск

Создаем файл и проверяем его наличие. (рис. 3.8).

```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ touch lab09-2.asm
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ls
in_out.asm  lab09-1  lab09-1.asm  lab09-1.o  lab09-2.asm
```

Рис. 3.8: Создание файла и проверка наличия

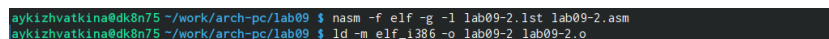
Вписываем в файл программу. (рис. 3.9).



```
lab09-2.asm [----] 8 L: [ 1+18 19/ 21] *(271 / 293b) 0032 0x020 [*][X]
SECTION
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Перетянуть 7Поиск 8Удалить 9Меню 10Выход
```

Рис. 3.9: Запись программы

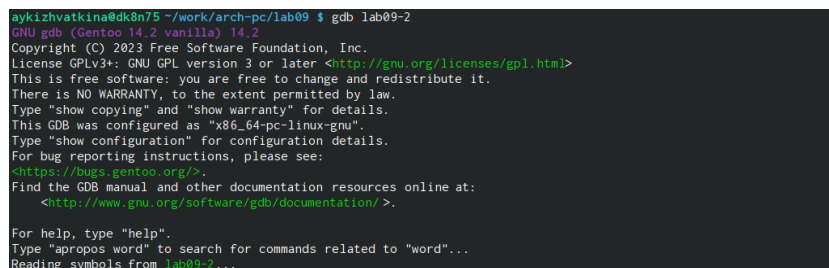
Создаем исполняемый файл. (рис. 3.10).



```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
```

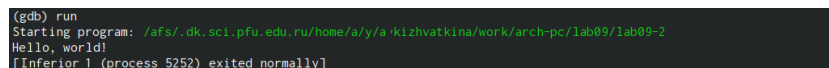
Рис. 3.10: Создание исполняемого файла

Запуск программы в отладчик gdb. (рис. 3.11 и рис. 3.12).



```
aykizhvatkina@dk8n75 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 14.2 vanilla) 14.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-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...
```

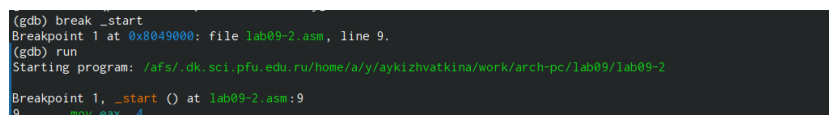
Рис. 3.11: Запуск программы в отладчик



```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 5252) exited normally]
```

Рис. 3.12: Команда run

Ставим брейк на _start. (рис. 3.13).



```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
```

Рис. 3.13: Создание брейка

Смотрим код программы начиная с `_start`. (рис. 3.14).

```
(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     $0x804a000,%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. 3.14: Команда `disassemble _start`

Переключаемся на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. В режиме АТТ отображаются знаки `%` и `$`, а в `intel` нет. (рис. 3.15).

```
(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,0x804a000
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. 3.15: Команда `disassembly-flavor intel`

Включаем режим псевдографики. (рис. 3.16).

```
lab09: gdb — Konsole
[ Register Values Unavailable ]

0x80490ae add BYTE PTR [eax],al
0x80490b0 add BYTE PTR [eax],al
0x80490b2 add BYTE PTR [eax],al
0x80490b4 add BYTE PTR [eax],al
0x80490b6 add BYTE PTR [eax],al
0x80490b8 add BYTE PTR [eax],al
0x80490ba add BYTE PTR [eax],al

native process 5259 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 3.16: Режим псевдографики

Проверяем наличие меток. (рис. 3.17).

```
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1      breakpoint       keep y   0x08049000 lab09-2.asm:9
       breakpoint already hit 1 time
(gdb)
```

Рис. 3.17: Проверка меток

Добавляем метку. (рис. 3.18).

```
(gdb) break *0x08049031
Breakpoint 2 at 0x08049031: file lab09-2.asm, line 20.
```

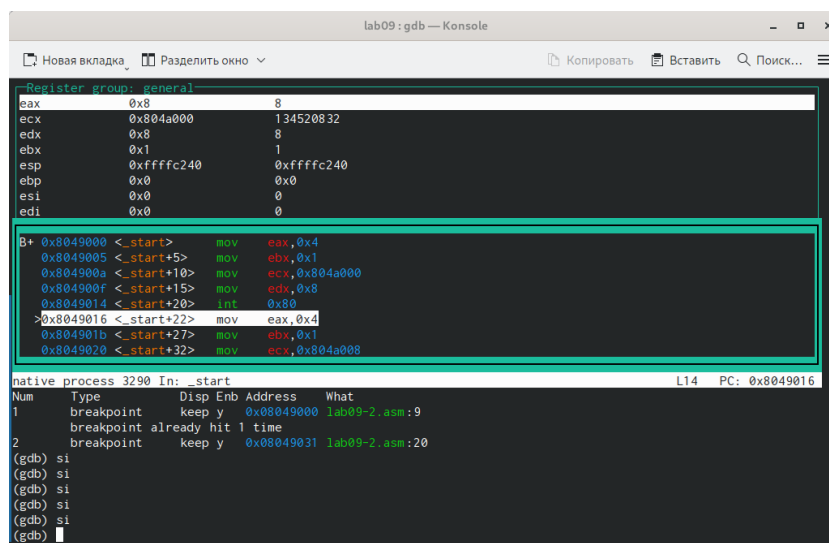
Рис. 3.18: Добавление метки

Проверяем наличие метки. (рис. 3.19).

```
(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
```

Рис. 3.19: Проверка меток

С помощью команды si смотрим регистры и изменяем их. (рис. 3.20).



```
lab09: gdb — Konsole
Новая вкладка Разделить окно Копировать Вставить Поиск...
Register group: general
eax 0x8 8
ecx 0x804a000 134520832
edx 0x8 8
ebx 0x1 1
esp 0xffffc240 0xffffc240
ebp 0x0 0x0
esi 0x0 0x0
edi 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
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
native process 3290 In: _start L14 PC: 0x8049016
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) si
(gdb)
```

Рис. 3.20: Команда si

Вводим команду i r.

The screenshot shows a GDB console window with the title 'lab09: gdb — Konsole'. The top menu bar includes 'Новая вкладка', 'Разделить окно', 'Копировать', 'Вставить', and 'Поиск...'. The main window is divided into two panes. The top pane, titled 'Register group: general', displays the values of general-purpose registers: eax (0x8), ecx (0x804a000), edx (0x8), ebx (0x1), esp (0xffffc240), ebp (0x0), esi (0x0), and edi (0x0). The bottom pane shows assembly code with addresses and instructions: 0x8049000 <_start> mov eax, 0x8; 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. Below the assembly code, the 'native process 3290 In: _start' section shows the same register values as the top pane, with the instruction pointer (eip) at 0x8049016. The status bar at the bottom indicates '--Type <RET> for more, q to quit, c to continue without paging--'.

Рис. 3.21: Команда i r

С помощью команды смотрим значение переменной msg1. (рис. 3.22).

The screenshot shows a GDB console with the command '(gdb) x/1sb &msg1' entered. The output is '0x804a000 <msg1>: "Hello, "'.

Рис. 3.22: Просмотр значения переменной

Смотрим значение переменной msg2. (рис. 3.23).

The screenshot shows a GDB console with the command '(gdb) x/1sb 0x804a008' entered. The output is '0x804a008 <msg2>: "world!\n\034"'. The '\034' represents a null terminator.

Рис. 3.23: Просмотр значения переменной

С помощью команды set меняем значение переменной msg1. (рис. 3.24).

The screenshot shows a GDB console with two commands: '(gdb) set {char}&msg1='h'' and '(gdb) x/1sb &msg1'. The output is '0x804a000 <msg1>: "hello, "'.

Рис. 3.24: Команда set

С помощью команды set меняем значение переменной msg2. (рис. 3.25).

The screenshot shows a GDB console with two commands: '(gdb) set {char}0x804a008='G'' and '(gdb) x/1sb 0x804a008'. The output is '0x804a008 <msg2>: "Gorld!\n\034"'. The '\034' represents a null terminator.

Рис. 3.25: Изменение msg2

Выводим значение регистров ecx и eax. (рис. 3.26).

```
(gdb) p/f $msg1
$10 = void
(gdb) p/s $eax
$11 = 4
(gdb) p/t $eax
$12 = 100
(gdb) p/s $ecx
$13 = 134520832
(gdb) p/x $ecx
$14 = 0x804a000
```

Рис. 3.26: Вывод значений

Изменяем значение регистра `ebx`. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум. (рис. 3.27).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$7 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2
```

Рис. 3.27: Изменение значений регистра

Копируем файл `lab8-2.asm` в папку. (рис. 3.28).

```
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис. 3.28: Команда `cp`

Создаем исполняемый файл. (рис. 3.29).

```
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 3.29: Создание исполняемого файла

Запускаем файл в отладчике с указанием аргументов. (рис. 3.30).

```
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент2 'аргумент 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...
```

Рис. 3.30: Запуск файла

Ставим метку на `_start`. (рис. 3.31).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab09/lab09-3 аргумент1 аргумент
2 аргумент\ 3
Breakpoint 1, _start () at lab09-3.asm:8
8      pop ecx ; Извлекаем из стека в 'ecx' количество
```

Рис. 3.31: Создание метки

Проверяем количество элементов в вершине стека. (рис. 3.32).

```
(gdb) x/x $esp
0xffffc1f0: 0x00000005
```

Рис. 3.32: Проверка количества элементов

Просматриваем позиции стека. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт.

```
(gdb) x/s *(void**)( $esp + 4)
0xffffc48d: "/afs/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8)
0xffffc4d7: "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xffffc4e9: "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xffffc4fa: "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffc4fc: "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
```

Рис. 3.33: Проверка позиций стека

Переделываем программу из лабораторной работы 8 так, чтобы вычисления были подпрограммой. (рис. 3.34).

```

lab8-4.asm [----] 3 L: [ 1+29 30/ 32] *(336 / 354b) 0032 0x020 [*)(X]
%include "lab8-4.asm"
section .data
P1 db "f(x)=4x+3",0
0 db "f(x)=4x+3",0
section .text
global _start
_start:
pop ecx
pop ecx
sub ecx,1
mov esi,0
mov ecx,P1
call sprintf
next:
cmp ecx,0
jz _end
pop ecx
call atoi
call P2
add esi,ecx
loop next
_end:
mov ecx,0
call sprintf
mov ecx,esi
call iprintLF
call quit
P2:
mov ecx,4
mul ecx
add ecx,3
ret

```

Рис. 3.34: Изменение программы

Создаем исполняемый файл. (рис. 3.35).

```

aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab8-4.asm
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab8-4 lab8-4.o

```

Рис. 3.35: Создание исполняемого файла

Запускаем файл и проверяем правильность выполнения. (рис. 3.36).

```

aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ./lab8-4 1 2 3
f(x)=4x+3
Результат: 33
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ./lab8-4 1 2 3 4
f(x)=4x+3
Результат: 52
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ./lab8-4 1 2 3 4 5
f(x)=4x+3
Результат: 75

```

Рис. 3.36: Запуск файла

Создаем файл. (рис. 3.37).

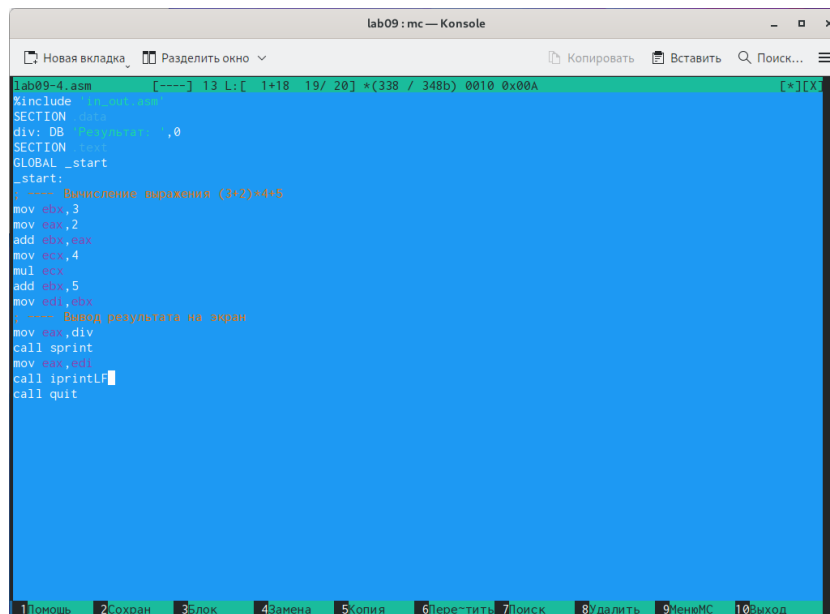
```

aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ touch lab09-4.asm
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ls
in_out.asm lab09-1.asm lab09-2 lab09-2.lst lab09-3 lab09-3.lst lab09-4.asm lab8-4.asm
lab09-1 lab09-1.o lab09-2.asm lab09-2.o lab09-3.asm lab09-3.o lab8-4 lab8-4.o

```

Рис. 3.37: Создание файла

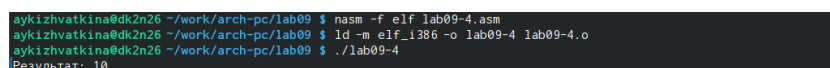
Переписываем программу в файл. (рис. 3.38).



```
lab09-4.asm [----] 13 L: [ 1+18 19/ 20] *(338 / 348b) 0010 0x00A [*][X]
%include "lab09-4.inc"
SECTION .data
div: DB 0x00000000,0
SECTION .text
GLOBAL _start
_start:
; ---- Вывод значения (3+2)*4*5
mov ebx,3
mov ecx,2
add ebx,ecx
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov ecx,edi
call iprintlf
call quit
```

Рис. 3.38: Программа в файле

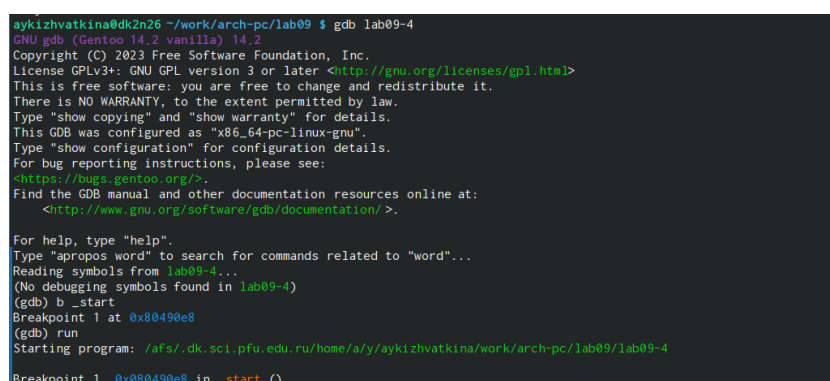
Создаем исполняемый файл и запускаем его. (рис. 3.39). Программа выводит неверный ответ.



```
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ ./lab09-4
Результат: 10
```

Рис. 3.39: Создание исполняемого файла и запуск

Запускаем программу в отладчике. (рис. 3.40).



```
aykizhvatkina@dk2n26 ~/work/arch-pc/lab09 $ gdb lab09-4
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 lab09-4...
(No debugging symbols found in lab09-4)
(gdb) b _start
Breakpoint 1 at 0x00490e8
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab09/lab09-4
Breakpoint 1, 0x00490e8 in _start ()
```

Рис. 3.40: Запуск программы в отладчике

Открываем регистры и анализируем их. Я поняла, что некоторые регистры стоят не на своих местах. Исправляем это. (рис. 3.41).

```

lab09:gdb — Konsole
[+] Новая вкладка [x] Разделить окно
[+] Копировать [x] Вставить [x] Поиск... [x]

Register group: general
eax 0x8 8
ecx 0x4 4
edx 0x0 0
ebx 0x5 5
esp 0xfffffc230 0xfffffc230
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x80490fb 0x80490fb <_start+19>

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>
0x804910a <_start+34> mov eax,edi

native process 3226 In: _start L?? PC: 0x80490fb
(gdb) layout regs
(gdb) si
0x080490ed in _start ()
(gdb) si
0x080490f2 in _start ()
(gdb) si
0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) si
0x080490fb in _start ()
(gdb)

```

Рис. 3.41: Анализ регистров

Создаем исполняемый файл и запускаем его. Проверяем правильность выполнения. (рис. 3.42).

```

aykizhvatkina@dk3n51 ~/work/arch-pc/lab09 $ nasm -f elf lab09-4.asm
aykizhvatkina@dk3n51 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
aykizhvatkina@dk3n51 ~/work/arch-pc/lab09 $ ./lab09-4
Результат: 25

```

Рис. 3.42: Создание исполняемого файла и запуск

4 Выводы

С помощью данной лабораторной работы мы приобрели навыки написания программ с использованием подпрограмм. Ознакомились с методами отладки при помощи GDB и его основными возможностями.