

Отчет по лабораторной работе № 9

Дисциплина: архитектура компьютера

Выслоух Алиса Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация подпрограмм в NASM	7
3.2	Самостоятельная работа.	18
4	Выводы	25

Список иллюстраций

3.1	Создание папки и файла.	7
3.2	Ввод текста.	8
3.3	Запуск программы.	8
3.4	Изменение текста программы.	9
3.5	Запуск программы.	10
3.6	Запуск программы в gdb.	10
3.7	Брейкпоинт.	10
3.8	Получение файла листинга.	11
3.9	Первая выбранная строка.	12
3.10	Вторая выбранная строка.	12
3.11	Третья выбранная строка.	13
3.12	Ввод программы.	13
3.13	Ввод программы.	13
3.14	Ввод программы.	14
3.15	Ввод программы.	14
3.16	Ввод программы.	15
3.17	Ввод программы.	15
3.18	Ввод программы.	16
3.19	Ввод программы.	17
3.20	Ввод программы.	17
3.21	Ввод программы.	17
3.22	Ввод программы.	18
3.23	Ввод программы.	18
3.24	Ввод программы.	19
3.25	Ввод программы.	20
3.26	Ввод программы.	21
3.27	Ввод программы.	21
3.28	Ввод программы.	22
3.29	Ввод программы.	23
3.30	Ввод программы.	24

Список таблиц

1 Цель работы

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

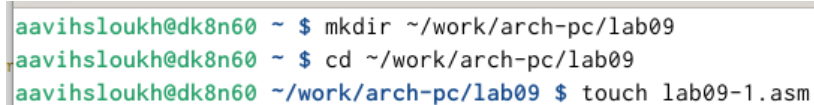
2 Задание

1. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задание для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM

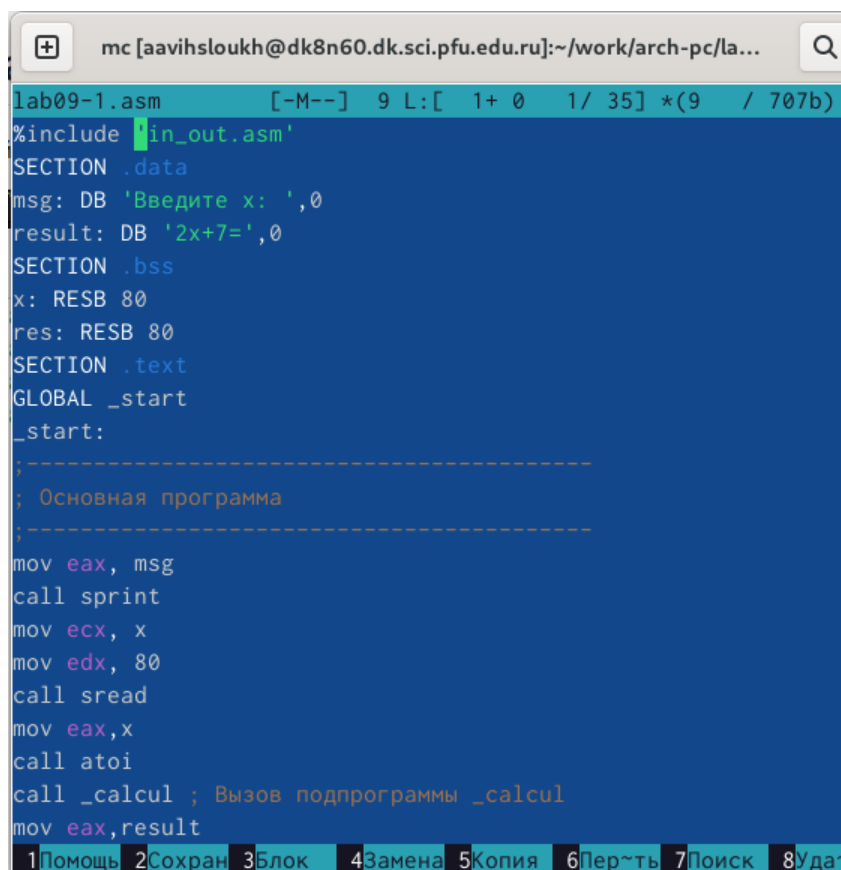
Я создала каталог lab09 и создала файл lab09-1.asm (рис. 3.1).



```
aavihsloukh@dk8n60 ~ $ mkdir ~/work/arch-pc/lab09
aavihsloukh@dk8n60 ~ $ cd ~/work/arch-pc/lab09
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ touch lab09-1.asm
```

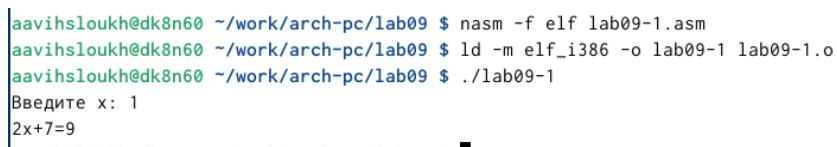
Рис. 3.1: Создание папки и файла.

Я ввела текст листинга в файл и запустила программу (рис. 3.2) (рис. 3.3).



```
lab09-1.asm [-M--] 9 L:[ 1+ 0 1/ 35] *(9 / 707b)
#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
```

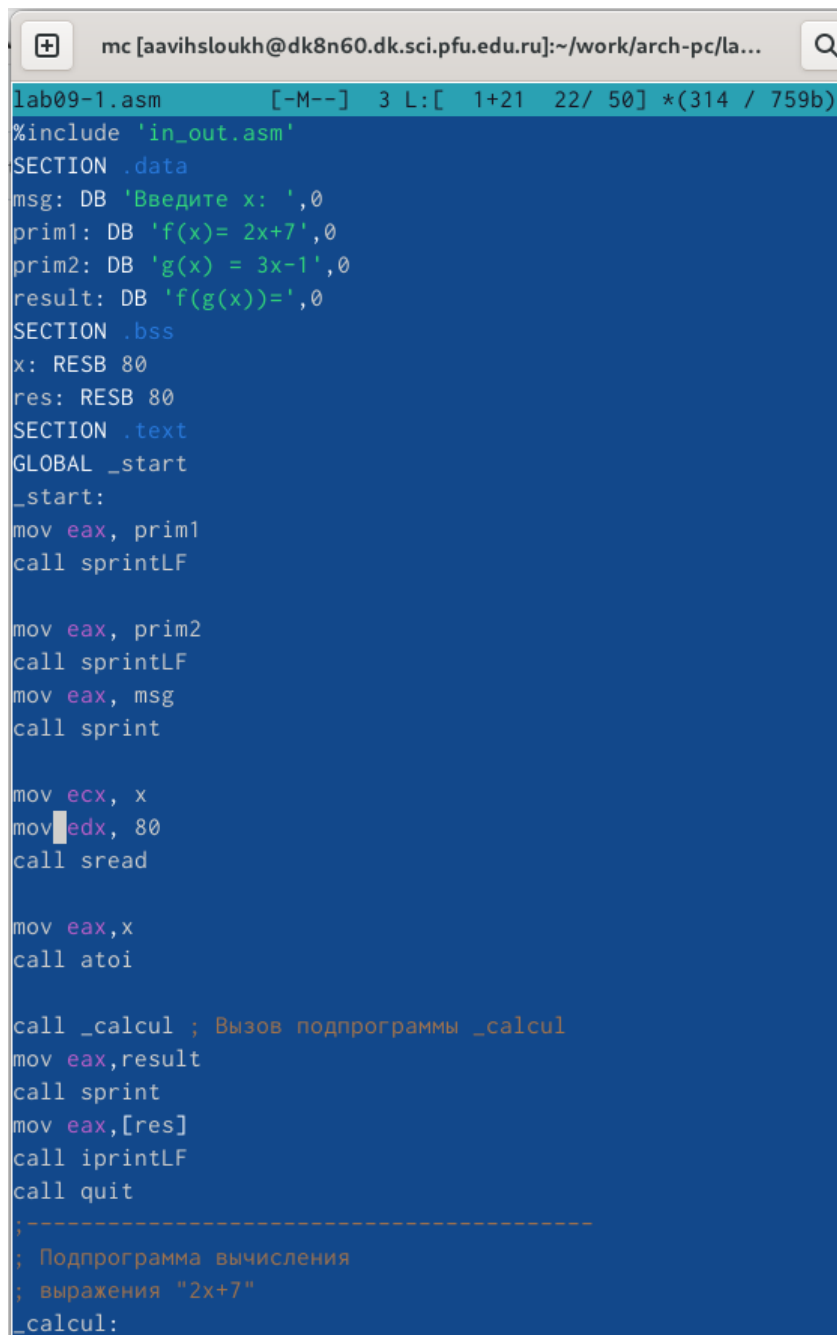
Рис. 3.2: Ввод текста.



```
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 1
2x+7=9
```

Рис. 3.3: Запуск программы.

Я изменила текст программы, чтобы она решала выражение $f(g(x))$. (рис. 3.4)
(рис. 3.5).



```
lab09-1.asm      [-M--]  3 L:[ 1+21  22/ 50] *(314 / 759b)
#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
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:
```

Рис. 3.4: Изменение текста программы.

```

aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ./lab09-1
f(x)= 2x+7
g(x) = 3x-1
Введите x: 1
f(g(x))=11

```

Рис. 3.5: Запуск программы.

Я создала файл lab09-2.asm и вписал туда программу. Я загрузила и запустила файл второй программы в отладчик gdb. (рис. 3.6).

```

aavihsloukh@dk8n60 - lab09
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
aavihsloukh@dk8n60 ~/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/a/aavihsloukh/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 10256) exited normally]
(gdb)

```

Рис. 3.6: Запуск программы в gdb.

Я поставил брейкпоинт на метку _start и запустил программу. (рис. 3.7).

```

(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/a/aavihsloukh/work/arch-pc/lab09/lab09-2

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

```

Рис. 3.7: Брейкпоинт.

Я просмотрела дисассимплированный код программы начиная с метки. (рис. 3.8).

```

(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. 3.8: Получение файла листинга.

С помощью команды я переключилась на intel'овское отображение синтаксиса. Отличие заключается в командах, в диссасилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть (рис. 3.9).

```

(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. 3.9: Первая выбранная строка.

Для удобства я включила режим псевдографики (рис. 3.10).

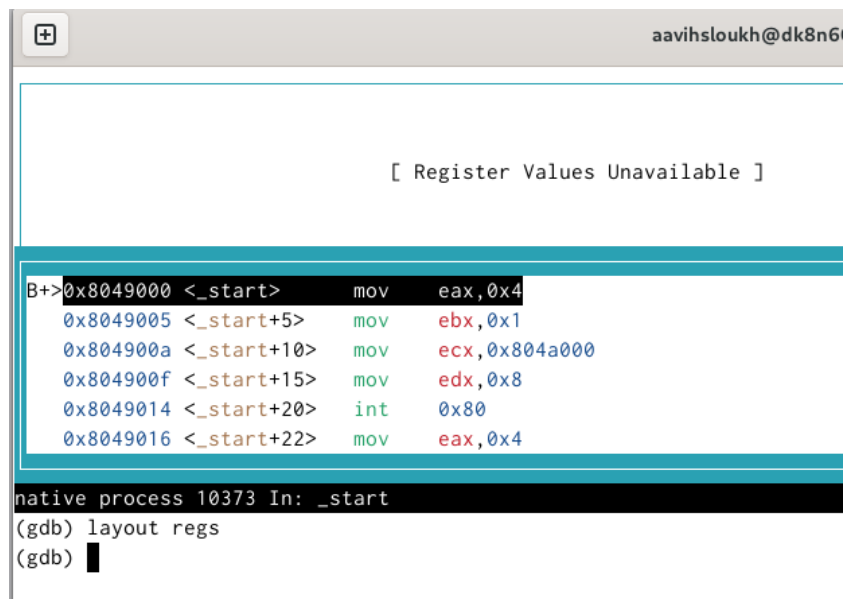


Рис. 3.10: Вторая выбранная строка.

Я добавила еще одну метку на предпоследнюю инструкцию (рис. 3.11) (рис. 3.11).

```
(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
(gdb) b *0x8049031
```

Рис. 3.11: Третья выбранная строка.

```
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) █
```

Рис. 3.12: Ввод программы.

С помощью команды `si` я посмотрела регистры и изменила их (рис. 3.13).

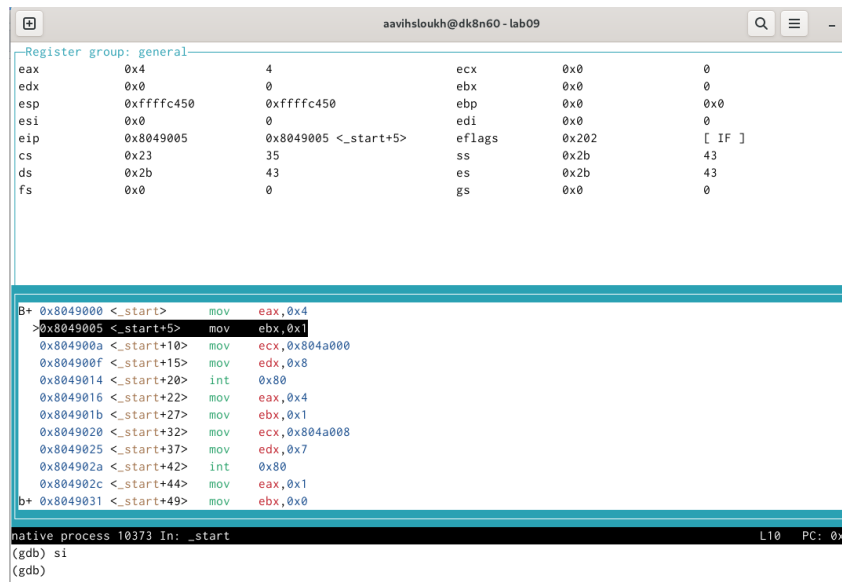


Рис. 3.13: Ввод программы.

С помощью команды `si` я посмотрела значение переменной `msg1`, следом я посмотрела значение второй переменной `msg2` (рис. 3.14).

```

aavihsloukh@dk8n60 - lab09

Register group: general
eax      0x4      4      ecx      0x0
edx      0x0      0      ebx      0x0
esp      0xffffc450 0xffffc450  ebp      0x0
esi      0x0      0      edi      0x0
eip      0x8049005 0x8049005 <_start+5>  eflags   0x202
cs       0x23     35     ss       0x2b
ds       0x2b     43     es       0x2b
fs       0x0      0      gs       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
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 10373 In: _start
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 3.14: Ввод программы.

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

```

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

```

Рис. 3.15: Ввод программы.

Я изменила 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: Ввод программы.

Я вывела значение регистров есх и еах.(рис. 3.17)

```
(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. 3.17: Ввод программы.

Я изменила значение регистра ебх. 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. 3.18)

```
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb) █
```

Рис. 3.18: Ввод программы.

Я завершила работу с файлов вышел.(рис. 3.19)


```

$7 = 2
(gdb) c
Continuing.
hhllo, Lor d!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) si
(gdb) q
A debugging session is active.

    Inferior 1 [process 10373] will be killed.

Quit anyway? (y or n)

```

Рис. 3.19: Ввод программы.

Я скопировала файл lab8-2.asm и переименовала его. Запустила файл в отладчике и указала аргументы.(рис. 3.20)

```

aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 '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...
(gdb) █

```

Рис. 3.20: Ввод программы.

Поставила метку на _start и запустила файл.(рис. 3.21)

```

Reading symbols from lab09-3...
(gdb) b _start
Breakpoint 1 at 0x00490e08: file lab09-3.asm, line 8.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aavihsloukh/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:8
8      pop    еск ; Извлекаем из стека в 'еск' количество
(gdb) █

```

Рис. 3.21: Ввод программы.

Я проверила адрес вершины стека и убедилась что там хранится 5 элементов.(рис. 3.22)

```
(gdb) x/x $esp
0xfffffc410:      0x00000005
(gdb) █
```

Рис. 3.22: Ввод программы.

Я посмотрела 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. 3.23)

```
(gdb) x/x $esp
0xfffffc410:      0x00000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc665:      "/afs/.dk.sci.pfu.edu.ru/home/a/a/aavihsloikh/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc6ad:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc6bf:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc6d0:      "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc6d2:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb) █
```

Рис. 3.23: Ввод программы.

3.2 Самостоятельная работа.

Я преобразовала программу из лабораторной работы №8 и реализовала вычисления как подпрограмму (рис. 3.24) (рис. 3.25)



```
lab09-4.asm [-M--] 0 L:[ 1+17 18/ 43] *(1
#include 'in_out.asm'

SECTION .data
prim DB "f(x)=8x-3",0
msg 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.

_end:
mov eax,msg
call sprintf
mov eax,esi
```

Рис. 3.24: Ввод программы.



```
aavihsloukh@dk8n60 - lab09
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-4.lst lab09-4.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-4 lab09-4.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ./lab09-4 1 2
f(x)=8x-3
Результат: 18
```

Рис. 3.25: Ввод программы.

Я переписала программу и попробовала запустить ее чтобы увидеть ошибку. Ошибка была арифметическая, так как вместо 25, программа выводит 10.(рис. 3.26) (рис. 3.27)

```

lab9-5.asm      [----] 11 L:[ 1+13
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 3.26: Ввод программы.

```

aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10

```

Рис. 3.27: Ввод программы.

После появления ошибки, я запустила программу в отладчике.(рис. 3.28)

```

aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(No debugging symbols found in lab9-5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/aavihsloukh/work/arch-pc/lab09/lab9-5

Breakpoint 1, 0x80490e8 in _start ()
(gdb) set disassemble _start
Ambiguous set command "disassemble _start": disassemble-next-line, disassembler-options.
(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 <iprintLF>
      0x08049111 <+41>:   call    0x80490db <quit>
End of assembler dump.
(gdb)

```

Рис. 3.28: Ввод программы.

Я открыла регистры и проанализировала их, поняла что некоторые регистры стоят не на своих местах и исправила это.(рис. 3.29)

+

aavihsloukh@dk8n60 - lab09

Register group: general

eax	0x8	8
ecx	0x4	4
edx	0x0	0
ebx	0x5	5
esp	0xffffc3e0	0xffffc3e0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x80490fb	0x80490fb <_start+19>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43

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

0x804910c <_start+36> call 0x8049086 <iprintLF>

0x8049111 <_start+41> call 0x80490db <quit>

0x8049116 add BYTE PTR [eax],al

native process 15845 In: _start

(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. 3.29: Ввод программы.

Я изменила регистры и запустила программу, программа вывела ответ 25, то есть все работает правильно (рис. 3.30)

```

aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ nasm -f elf lab9-5.asm
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
aavihsloukh@dk8n60 ~/work/arch-pc/lab09 $ gdb lab9-5
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 lab9-5...
(No debugging symbols found in lab9-5)
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/aavihsloukh/work/arch-pc/lab09/lab9-5
Результат: 25
[Inferior 1 (process 16645) exited normally]
(gdb)

```

Рис. 3.30: Ввод программы.

4 Выводы

Я приобрел навыки написания программ использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.