

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Бражко Александра Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выполнение самостоятельной работы	18
6	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Ввод листинга	8
4.3	Создание и проверка	9
4.4	Изменение программы	9
4.5	Создание и проверка	9
4.6	Создание файла	9
4.7	Ввод листинга	10
4.8	Запуск и загрузка	10
4.9	Запуск	11
4.10	Просмотр кода	11
4.11	Переключение	11
4.12	Режим псевдографики	12
4.13	Просмотр и добавление метки	12
4.14	Просмотр и замена	13
4.15	Просмотр и замена	13
4.16	Просмотр	13
4.17	Просмотр	14
4.18	Изменение значения	14
4.19	Изменение значения	14
4.20	Просмотр значений	15
4.21	Изменение значений	15
4.22	Завершение и выход	15
4.23	Копирование	16
4.24	Запуск	16
4.25	Установка и запуск	16
4.26	Проверка	16
4.27	Просмотр позиций стека	17
5.1	Программа	18
5.2	Создание и проверка	19
5.3	Программа	19
5.4	Создание и проверка	19

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

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

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

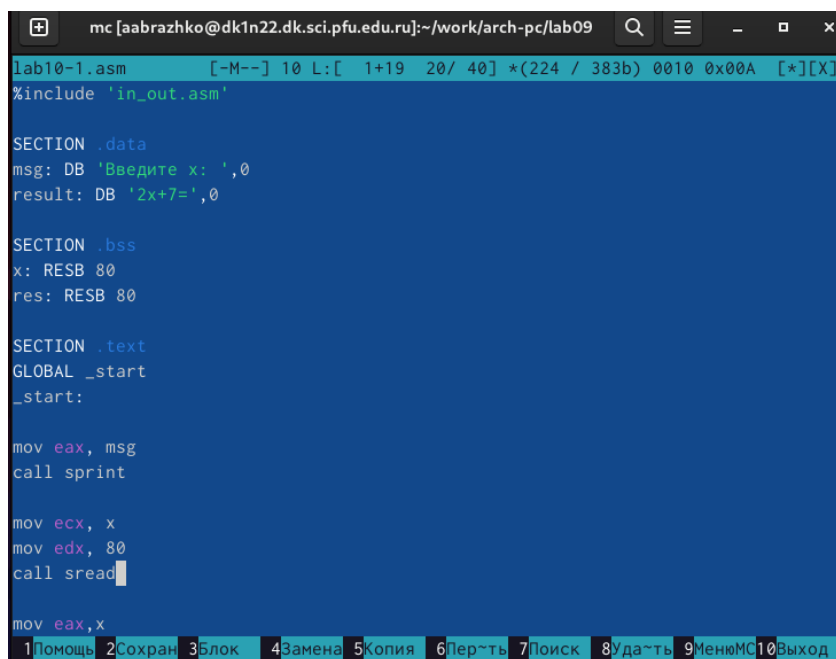
4 Выполнение лабораторной работы

Создаём каталог для выполнения лабораторной работы № 9, переходим в него и создаём файл lab9-1.asm (рис. 4.1).

```
aabrazhko@dk1n22 ~ $ mkdir ~/work/arch-pc/lab09
aabrazhko@dk1n22 ~ $ cd ~/work/arch-pc/lab09
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ touch lab10-1.asm
```

Рис. 4.1: Создание каталога и файла

Вводим в файл lab9-1.asm текст программы из листинга 9.1 (рис. ??).



```
mc [aabrazhko@dk1n22.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab09
lab10-1.asm [-M--] 10 L: [ 1+19 20/ 40] *(224 / 383b) 0010 0x00A [*][X]
#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:

mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
```

Рис. 4.2: Ввод листинга

Создаём исполняемый файл и проверяем его работу (рис. 4.3).


```

aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ nasm -f elf lab10-1.asm
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab10-1 lab10-1.o
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ./lab10-1
Введите x: 5
2x+7=17

```

Рис. 4.3: Создание и проверка

Вносим изменения в программу, чтобы решалось выражение $f(g(x))$ (рис. 4.4).

```

lab10-1.asm  [----]  8 L: [ 35+ 7 42/ 56] *(447 / 556b) 0010 0x00A  [*][X]
mov  eax,result
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. 4.4: Изменение программы

Создаём исполняемый файл и проверяем его работу (рис. 4.5).

```

aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ nasm -f elf lab10-1.asm
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab10-1 lab10-1.o
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ./lab10-1
f(x)=2x+7
g(x)=3x-1
Введите x: 1
f(g(x))=11

```

Рис. 4.5: Создание и проверка

Создаём файл lab9-2.asm (рис. 4.6).

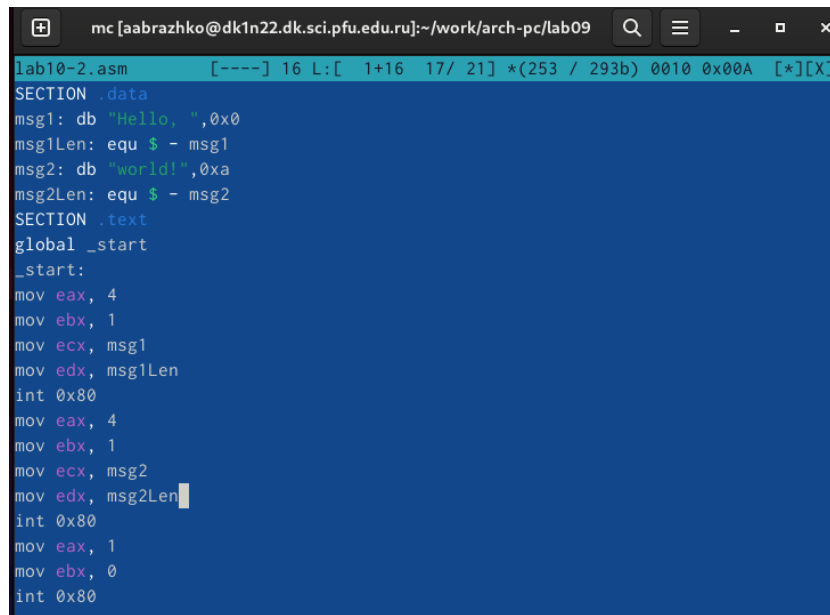
```

aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ touch lab10-2.asm

```

Рис. 4.6: Создание файла

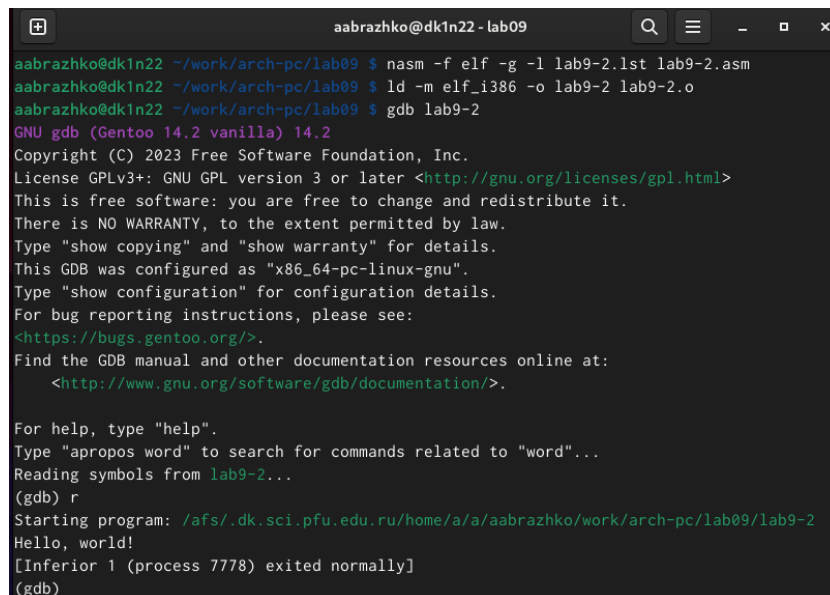
Вводим в файл lab9-2.asm текст программы из листинга 9.2 (рис. 4.7).



```
lab10-2.asm [----] 16 L: [ 1+16 17/ 21] *(253 / 293b) 0010 0x00A [*][X]
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. 4.7: Ввод листинга

Загружаем и запускаем файл в отладчике gdb (рис. 4.8).



```
aabrazhko@dk1n22 - lab09
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ gdb lab9-2
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 lab9-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aabrazhko/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 7778) exited normally]
(gdb)
```

Рис. 4.8: Запуск и загрузка

Ставим брекпоинт на метку _start и запускаем программу (рис. 4.9).

```
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aabrazhko/work/arch-pc/lab09/lab9-2

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

Рис. 4.9: Запуск

Просматриваем дисассимплированный код программы, начиная с метки (рис. 4.10).

```
(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. 4.10: Просмотр кода

Переключаемся на отображение команд с Intel'овским синтаксисом (рис. 4.11).

```
(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. 4.11: Переключение

Включим режим псевдографики для более удобного анализа программы (рис. 4.12).



Рис. 4.12: Режим псевдографики

Посмотрим наличие меток и добавим еще одну метку на предпоследнюю инструкцию (рис. 4.13).

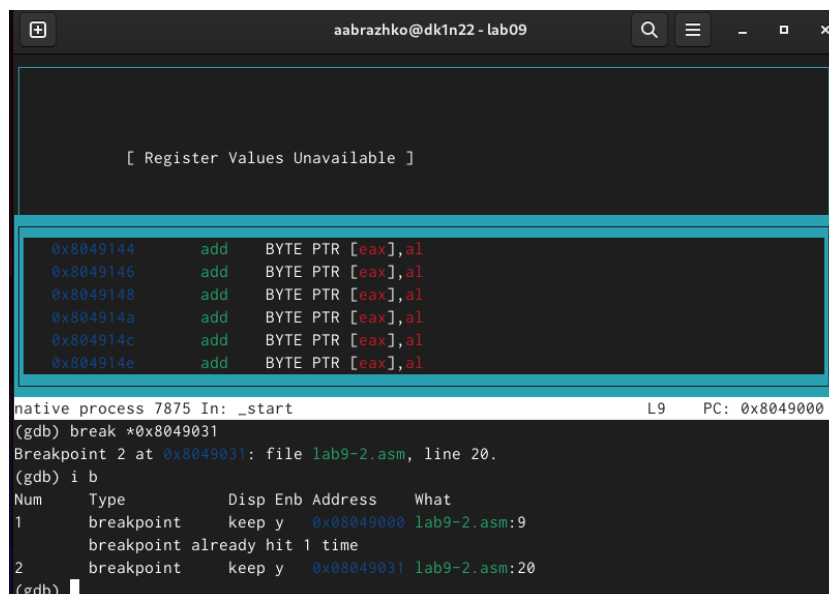


Рис. 4.13: Просмотр и добавление метки

С помощью команды `si` мы посмотрим регистры и изменим их (рис. 4.14, рис. 4.15).

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

native No process In:                                L??  PC: ??
(gdb) layout regs
(gdb) i r
The program has no registers now.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aabrazhko/work/arch-pc/lab09/
lab9-2
Hello, world!
(gdb) ior 1 (process 17583) exited normally]
```

Рис. 4.14: Просмотр и замена

```
aabrazhko@dk1n22 - lab09
[ Register Values Unavailable ]

0x804914c    add     BYTE PTR [eax],a1
0x804914e    add     BYTE PTR [eax],a1
0x8049150    add     BYTE PTR [eax],a1
0x8049152    add     BYTE PTR [eax],a1
0x8049154    add     BYTE PTR [eax],a1
0x8049156    add     BYTE PTR [eax],a1

native process 7875 In: _start                        L9  PC: 0x8049000
eip          0x8049000    0x8049000 <_start>
eflags       0x202       [ IF ]
cs           0x23        35
ss           0x2b        43
ds           0x2b        43
es           0x2b        43
fs           0x0         0
gs           0x0         0
```

Рис. 4.15: Просмотр и замена

С помощью команды посмотрим значение переменной msg1 (рис. 4.16).

```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) █
```

Рис. 4.16: Просмотр

Посмотрим значение переменной msg2 (рис. 4.17).

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

Рис. 4.17: Просмотр

С помощью команды set изменим значение переменной msg1 (рис. 4.18).

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

Рис. 4.18: Изменение значения

С помощью команды set изменим значение переменной msg2 (рис. 4.19).

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

Рис. 4.19: Изменение значения

Просмотрим значения регистра edx (рис. 4.20).

```
(gdb) p/s $edx
$2 = 0
(gdb) p/t $edx
$3 = 0
(gdb) p/x $edx
$4 = 0x0
(gdb) 
```

Рис. 4.20: Просмотр значений

С помощью команды `set` изменим значения 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. 4.21).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb) 
```

Рис. 4.21: Изменение значений

Завершаем работу файлов и выходим (рис. 4.22).

```
[Inferior 1 (process 3985) exited normally]
```

Рис. 4.22: Завершение и выход

Скопируем файл `lab8-2.asm`, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем `lab9-3.asm` (рис. 4.23).

```
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
```

Рис. 4.23: Копирование

Запускаем файл в отладчике и указываем аргументы (рис. 4.24).

```
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ gdb --args lab9-3 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 lab9-3...
(gdb)
```

Рис. 4.24: Запуск

Установим точку останова перед первой инструкцией в программе и запустим её (рис. 4.25).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 10.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/a/aabrazhko/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:10
10      pop ecx      ; Извлекаем из стека в 'ecx' количество
(gdb)
```

Рис. 4.25: Установка и запуск

Проверим адрес вершины стека (рис. 4.26).

```
(gdb) x/x $esp
0xfffffc430:      0x00000005
(gdb)
```

Рис. 4.26: Проверка

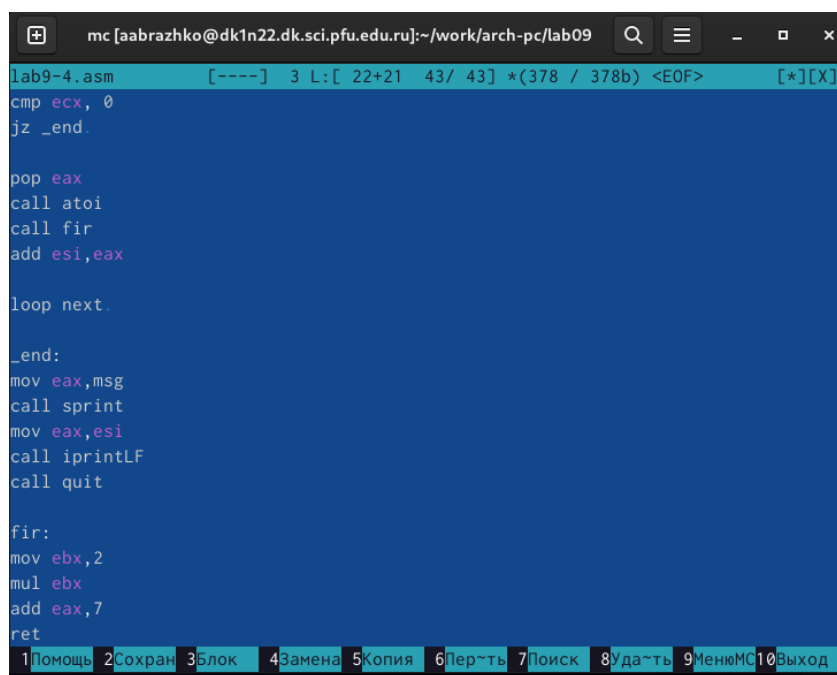
Просмотрим все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации. (рис. 4.27).

```
(gdb) x/x $esp
0xfffffc430: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc686: "/afs/.dk.sci.pfu.edu.ru/home/a/a/aabrazhko/work/arch-pc/lab09/lab
9-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc6cb: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc6dd: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc6ee: "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc6f0: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 4.27: Просмотр позиций стека

5 Выполнение самостоятельной работы

Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $\Pi(\Pi)$ как подпрограмму. (рис. 5.1).



```
lab9-4.asm      [----]  3 L:[ 22+21  43/ 43] *(378 / 378b) <EOF>  [*][X]
cmp ecx, 0
jz _end.

pop eax
call atoi
call fir
add esi, eax

loop next.

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

fir:
mov ebx, 2
mul ebx
add eax, 7
ret

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС 10Выход
```

Рис. 5.1: Программа

Создаём исполняемый файл и проверяем его работу (рис. 5.2).

```

aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-4.lst lab9-4.asm
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ./lab9-4 1 2
f(x)=7+2x
Результат: 20
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ./lab9-4 1 2 3
f(x)=7+2x
Результат: 33

```

Рис. 5.2: Создание и проверка

Переписываем программу из листинга 9.3 (рис. 5.3).

```

lab9-5.asm      [----] 13 L: [ 1+19 20/ 21] *(229 / 239b) 0010 0x00A [*][X]
#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. 5.3: Программа

Создаём исполняемый файл и проверяем его работу. Программа должна вывести 10, что является ошибкой, так как должно быть 25 (рис. 5.4).

```

aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab9-5.lst lab9-5.asm
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-5 lab9-5.o
aabrazhko@dk1n22 ~/work/arch-pc/lab09 $ ./lab9-5
Результат: 10

```

Рис. 5.4: Создание и проверка

6 Выводы

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

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.