

Отчёт по лабораторной работе №10

Дисциплина: архитектура компьютера.

Матюхин Павел Андреевич НММбд-02-22

Содержание

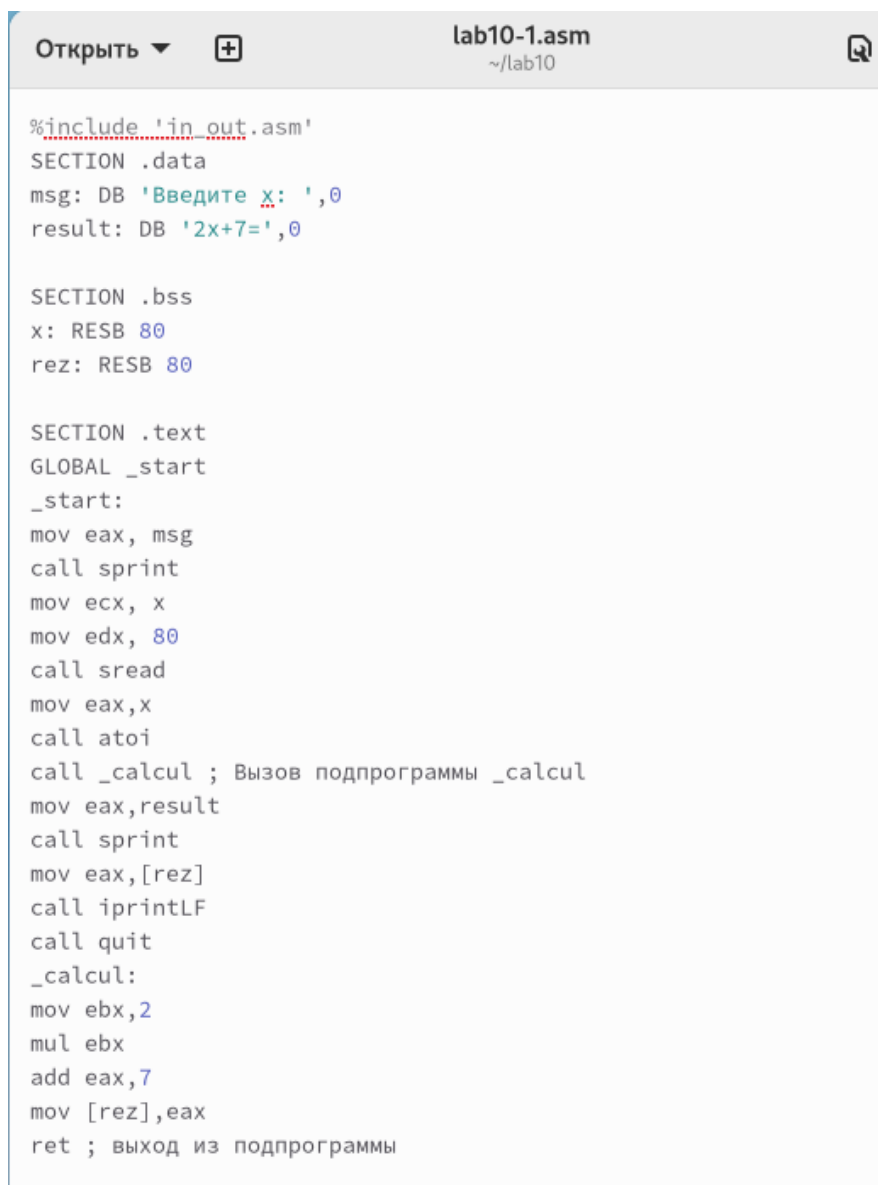
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Самостоятельная работа	20
4	Выводы	27

1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы № 10, перешел в него и создал файл lab10-1.asm:



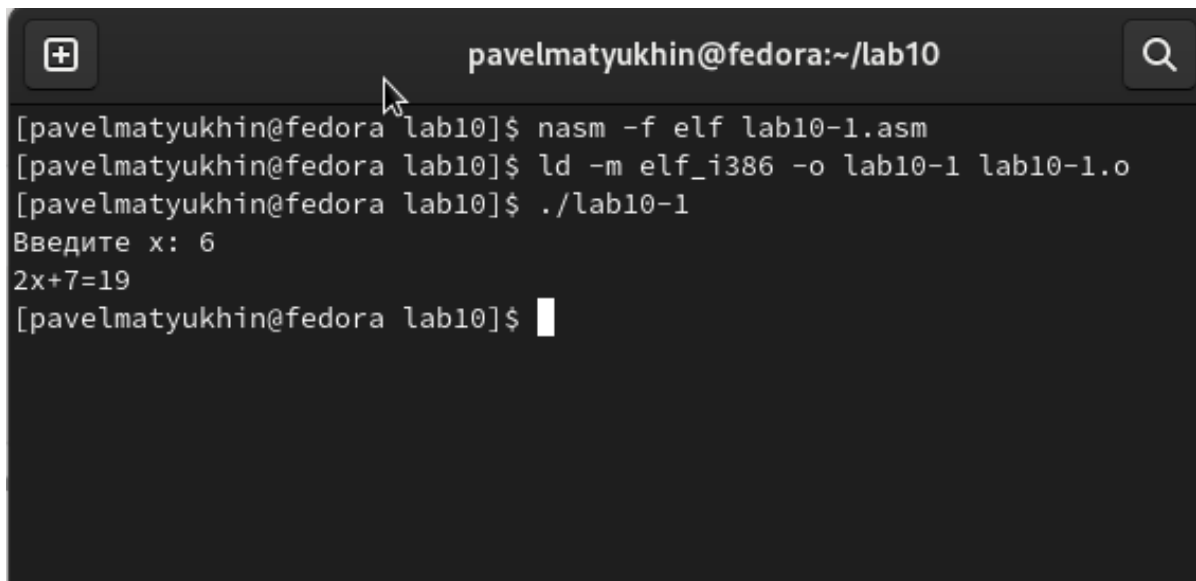
```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0

SECTION .bss
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Рис. 2.1: Файл lab10-1.asm

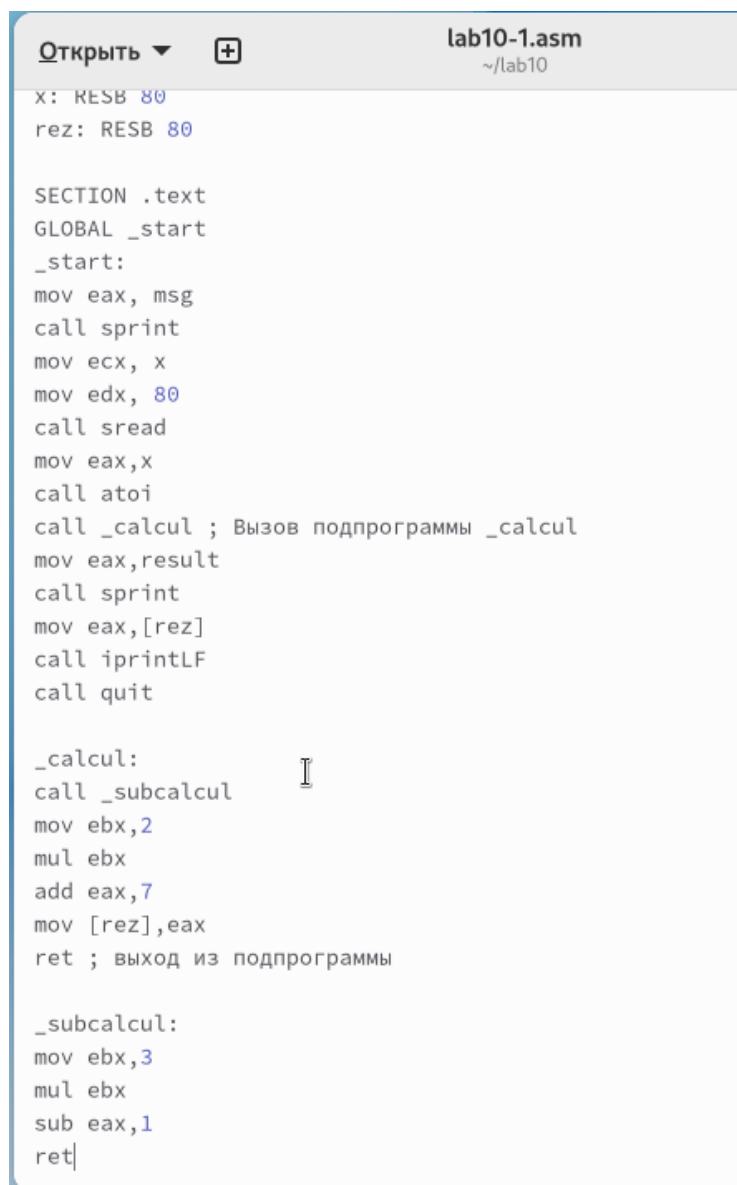
2. Изучил текст программы из листинга и проверил работу.

A terminal window with a dark background. The title bar shows 'pavelmatyukhin@fedora:~/lab10'. The terminal content shows the following sequence of commands and output:

```
[pavelmatyukhin@fedora lab10]$ nasm -f elf lab10-1.asm
[pavelmatyukhin@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[pavelmatyukhin@fedora lab10]$ ./lab10-1
Введите x: 6
2x+7=19
[pavelmatyukhin@fedora lab10]$
```

Рис. 2.2: Работа программы lab10-1.asm

3. Изменил текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$



```
Открыть ▾ + lab10-1.asm
~/lab10

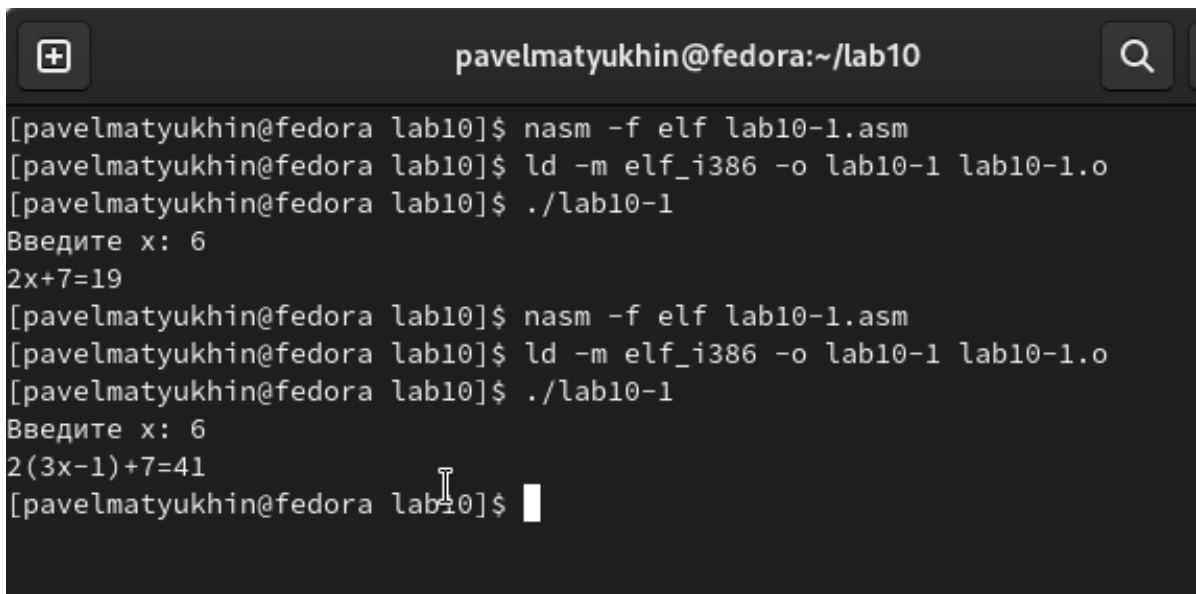
x: RESB 80
rez: RESB 80

SECTION .text
GLOBAL _start
_start:
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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

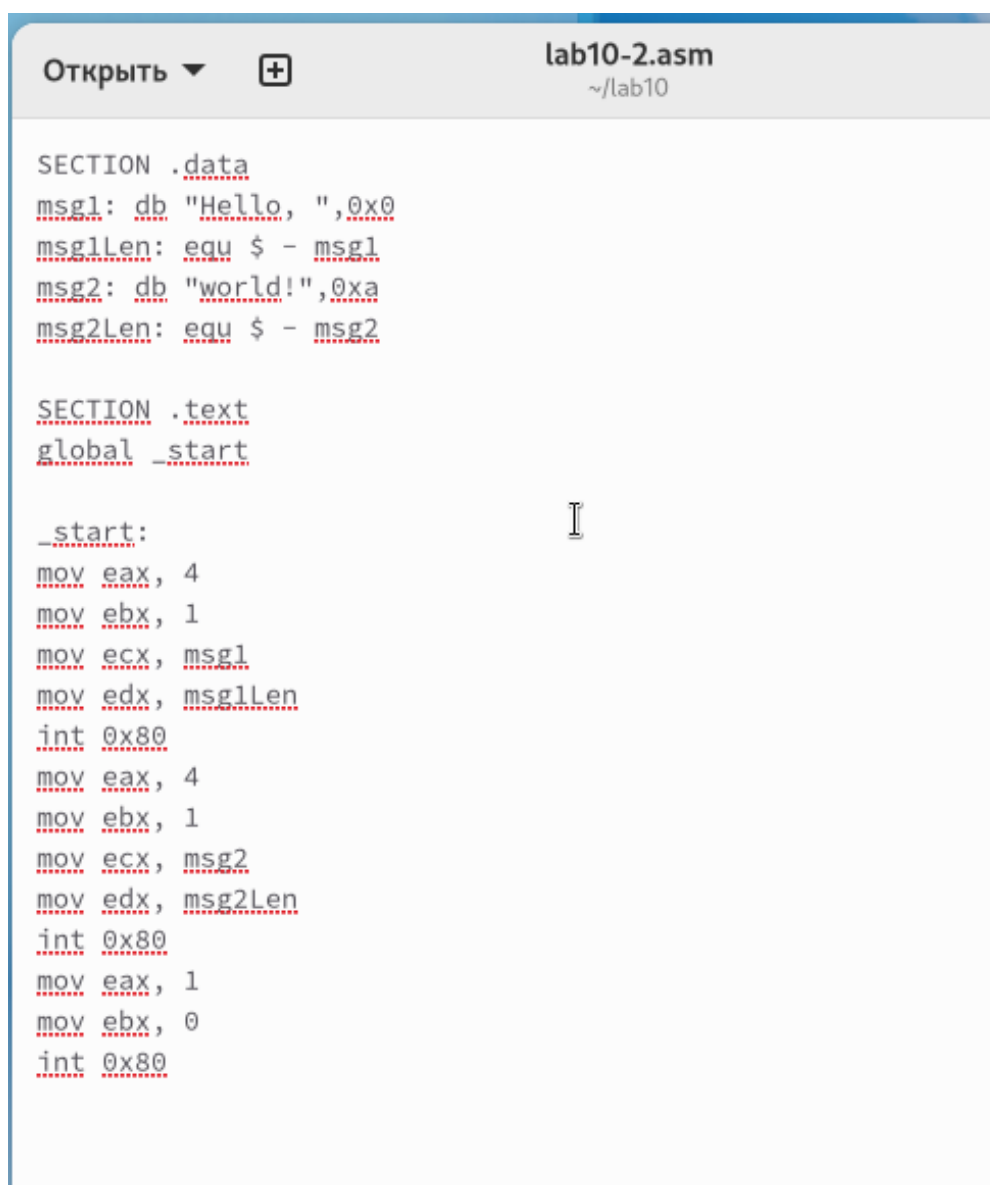
Рис. 2.3: Файл lab10-1.asm



```
pavelmatyukhin@fedora:~/lab10
[pavelmatyukhin@fedora lab10]$ nasm -f elf lab10-1.asm
[pavelmatyukhin@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[pavelmatyukhin@fedora lab10]$ ./lab10-1
Введите x: 6
2x+7=19
[pavelmatyukhin@fedora lab10]$ nasm -f elf lab10-1.asm
[pavelmatyukhin@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[pavelmatyukhin@fedora lab10]$ ./lab10-1
Введите x: 6
2(3x-1)+7=41
[pavelmatyukhin@fedora lab10]$
```

Рис. 2.4: Работа программы lab10-1.asm

4. Создал файл lab10-2.asm с текстом программы из Листинга 10.2 (Программа печати сообщения Hello world!):



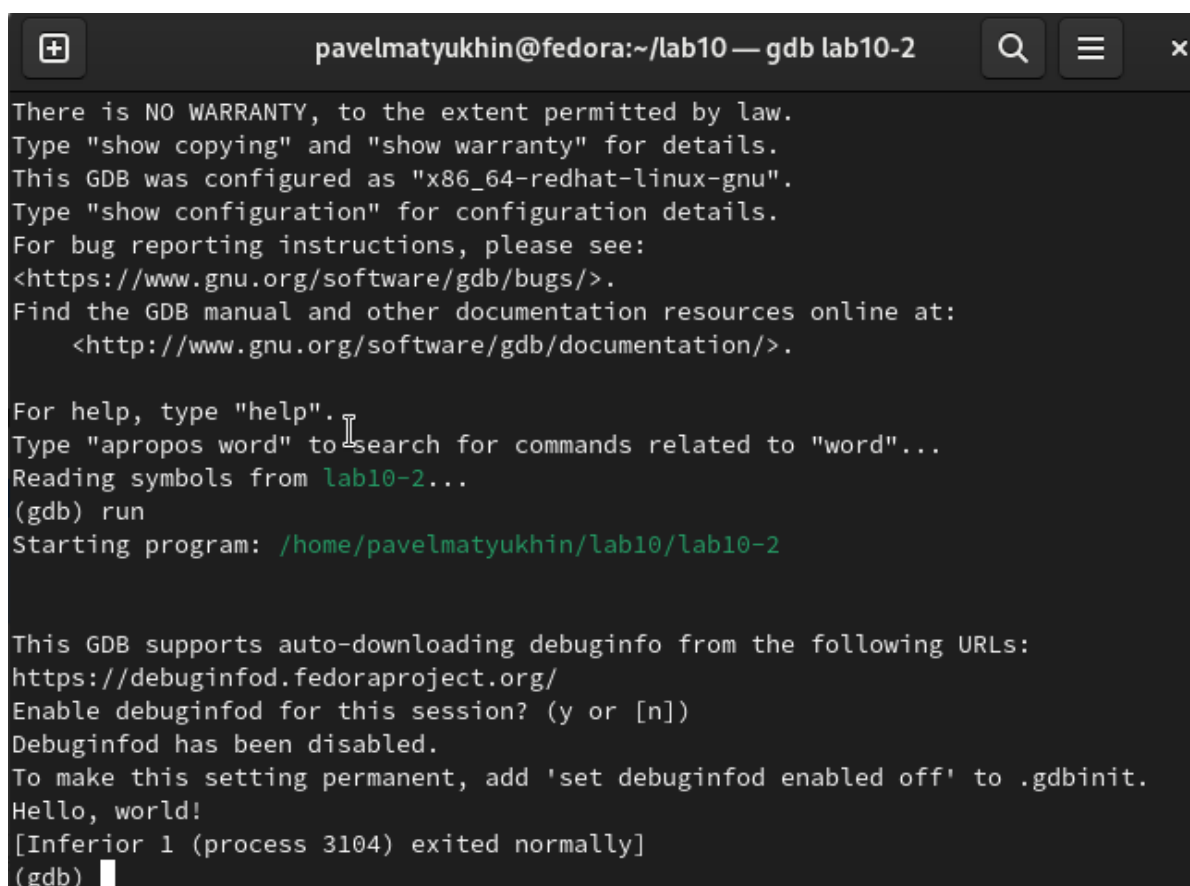
```
SECTION .data
msg1: db "Hello, ",0x0
msg1len: equ $ - msg1
msg2: db "world!",0xa
msg2len: equ $ - msg2

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. 2.5: Файл lab10-2.asm

Загрузил исполняемый файл в отладчик gdb: Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r)

A screenshot of a terminal window titled 'pavelmatyukhin@fedora:~/lab10 — gdb lab10-2'. The window shows the GDB startup sequence. It starts with a disclaimer about warranty and configuration, followed by a prompt for help. The user enters 'run', and the program starts, displaying 'Hello, world!'. The program then exits normally, and the user is back at the GDB prompt. The terminal text is as follows:

```
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-redhat-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 lab10-2...
(gdb) run
Starting program: /home/pavelmatyukhin/lab10/lab10-2

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 3104) exited normally]
(gdb)
```

Рис. 2.6: Работа программы lab10-2.asm в отладчике

Для более подробного анализа программы установил брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустил её. Посмотрел дисассимилированный код программы

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-2

Breakpoint 1 at 0x8049000: file lab10-2.asm, line 11.
(gdb) run
Starting program: /home/pavelmatyukhin/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:11
11      mov eax, 4
(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. 2.7: дисассимилированный код

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-2

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) 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. 2.8: дисассимилированный код в режиме интел

Проверил (`_start`) с помощью команды `info breakpoints` (кратко `i b`) Установил еще одну точку останова по адресу инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установите точку.

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-2
[ 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 3108 In: _start L11 PC: 0x8049000
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--cs 0x23 3
5
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0
(gdb)
```

Рис. 2.9: точка остановки

Выполнил 5 инструкций с помощью команды stepi (или si) и проследил за изменением значений регистров.

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd210 0xffffd210
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 3108 In: _start L12 PC: 0x8049005
eip      0x8049000 0x8049000 <_start>
eflags   0x202 [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--cs 0x23 3
5
ss      0x2b 43
ds      0x2b 43
es      0x2b 43
fs      0x0 0
gs      0x0 0
(gdb) si
(gdb) 
```

Рис. 2.10: изменение регистров

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-2
Register group: general
eax      0x4      4
ecx      0x804a008 134520840
edx      0x7      7
ebx      0x1      1
esp      0xffffd210 0xffffd210
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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 3108 In: _start L20 PC: 0x80490:
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.11: изменение регистров

Посмотрите значение переменной msg1 по имени Посмотрите значение переменной msg2 по адресу Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Измените первый символ переменной msg1 Замените любой символ во второй переменной msg2.

```
native process 3108 In: _start
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lor!d!\n\034"
(gdb) 
```

Рис. 2.12: изменение значения переменной

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра `edx`. С помощью команды `set` изменил значение регистра `ebx`


```
native process 3108 In: _start
```

```
(gdb) p/s $ecx
```

```
$3 = 134520840
```

```
(gdb) p/x $ecx
```

```
$4 = 0x804a008
```

```
(gdb) p/s $edx
```

```
$5 = 7
```

```
(gdb) p/t $edx
```

```
$6 = 111
```

```
(gdb) p/x $edx
```

```
$7 = 0x7
```

```
(gdb) 
```

Рис. 2.13: вывод значения регистра

С помощью команды set изменил значение регистра ebx

```
native process 3108 In: _start
(gdb) p/t $edx
$6 = 111
(gdb) p/x $edx
$7 = 0x7
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb) 
```

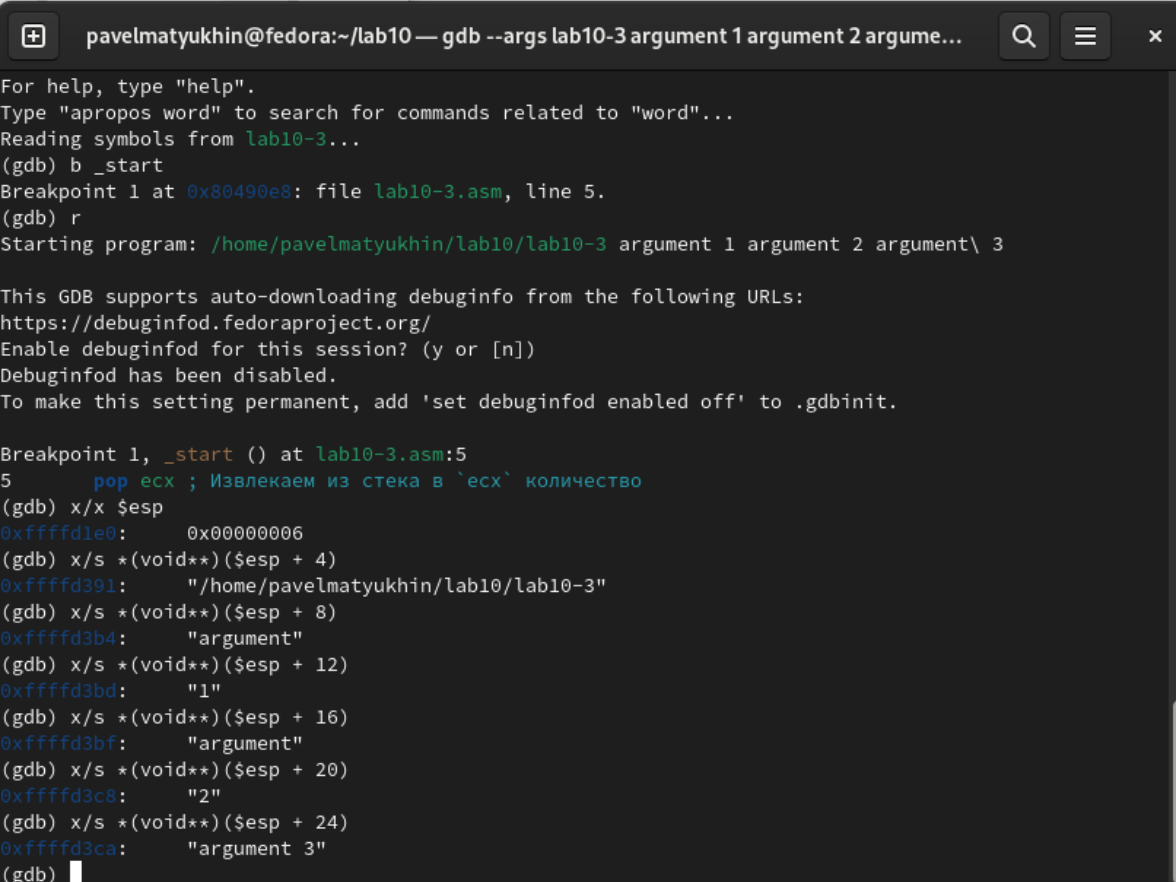
Рис. 2.14: вывод значения регистра

5. Скопировал файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы

Установил точку останова перед первой инструкцией в программе и запустил ее.

Посмотрел 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.



```
pavelmatyukhin@fedora:~/lab10 — gdb --args lab10-3 argument 1 argument 2 argume...
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab10-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) r
Starting program: /home/pavelmatyukhin/lab10/lab10-3 argument 1 argument 2 argument\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1e0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd391: "/home/pavelmatyukhin/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3b4: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3bd: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3bf: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3c8: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3ca: "argument 3"
(gdb) 
```

Рис. 2.15: вывод значения регистра

3 Самостоятельная работа

1. Преобразовал программу из лабораторной работы №9 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.



```
SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
call calc
add esi,eax

loop next

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


calc:
mov ebx,2
mul ebx
add eax,7
ret
```

Рис. 3.1: Файл lab10-4.asm

```
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$ nasm -f elf lab10-4.asm  
[pavelmatyukhin@fedora lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o  
[pavelmatyukhin@fedora lab10]$ ./lab10-4  
f(x)=7+2x  
Результат: 0  
[pavelmatyukhin@fedora lab10]$ ./lab10-4 1 2 3 4 5 6 7 8  
f(x)=7+2x  
Результат: 128  
[pavelmatyukhin@fedora lab10]$
```

Рис. 3.2: Работа программы lab10-4.asm

2. С помощью отладчика GDB, анализируя изменения значений регистров, определил ошибку и исправил ее.

Открыть ▾ 

lab10-5.asm
~/lab10

```
%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. 3.3: код с ошибкой

The screenshot shows a GDB window titled "pavelmatyukhin@fedora:~/lab10 — gdb lab10-5". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers: eax (0x8), ecx (0x4), edx (0x0), ebx (0xa), esp (0xffffd210), ebp (0x0), esi (0x0), and edi (0xa). The middle section shows a list of assembly instructions with their addresses and disassembled forms. The instruction at address 0x8049100 is highlighted: "> 0x8049100 <_start+24> mov eax,0x804a000". The bottom section shows the status of the native process 3392, indicating it is in the "_start" function at address L16, with the PC (Program Counter) set to 0x8049100. Below this, a breakpoint is listed: "Breakpoint 1, _start () at lab10-5.asm:8". The GDB prompt "(gdb)" is visible at the bottom of the window.

```
pavelmatyukhin@fedora:~/lab10 — gdb lab10-5

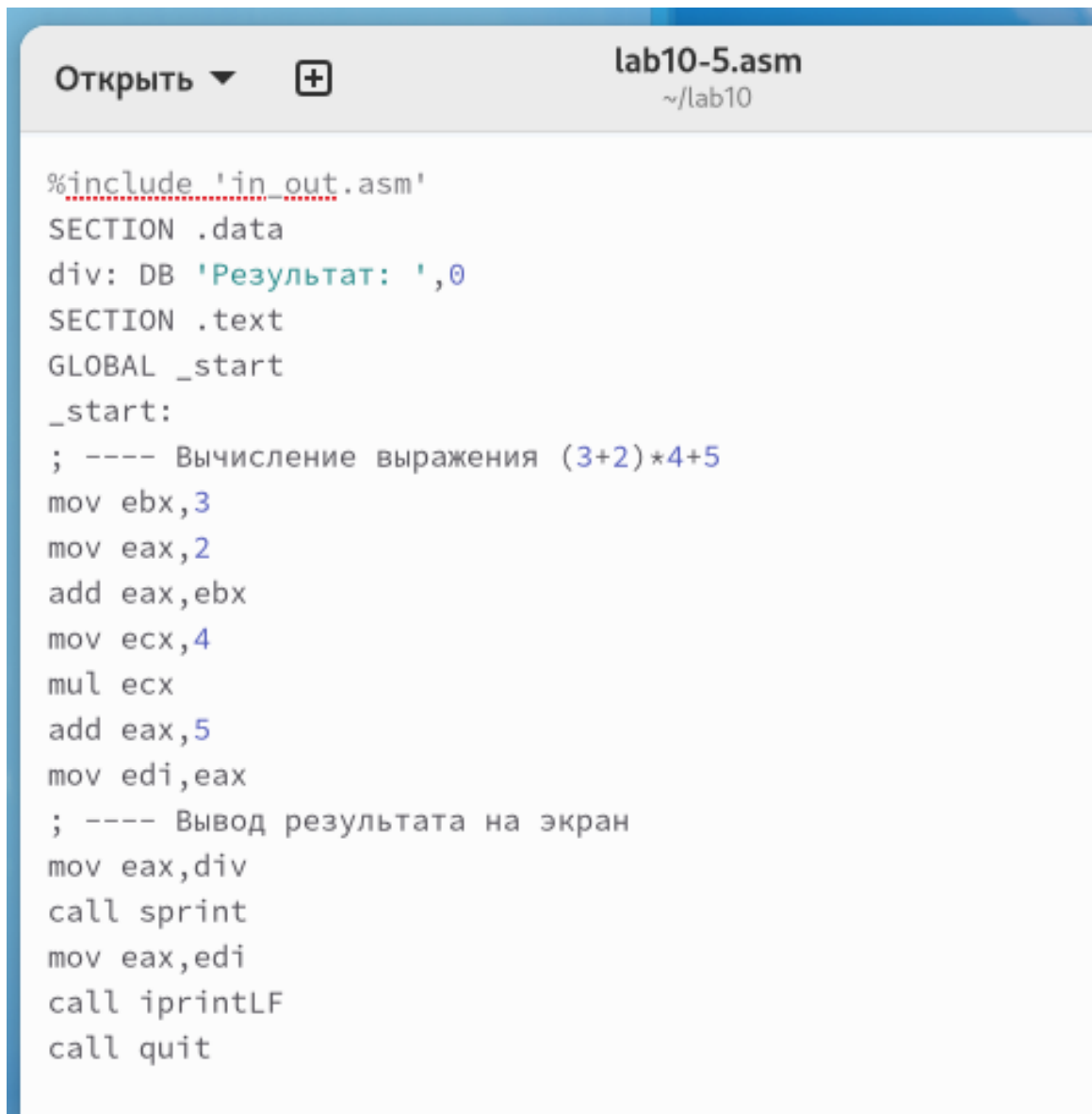
Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa     10
esp      0xffffd210 0xffffd210
ebp      0x0      0x0
esi      0x0      0
edi      0xa     10

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>

native process 3392 In: _start L16 PC: 0x8049100

Breakpoint 1, _start () at lab10-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 3.4: отладка




```
Открыть ▾  lab10-5.asm  
~/lab10  
  
%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 eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,eax  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рис. 3.5: код исправлен

```
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$ ./lab10-5  
Результат: 10  
[pavelmatyukhin@fedora lab10]$ nasm -g -f elf lab10-5.asm  
[pavelmatyukhin@fedora lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o  
[pavelmatyukhin@fedora lab10]$ ./lab10-5  
Результат: 25  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$  
[pavelmatyukhin@fedora lab10]$
```

Рис. 3.6: проверка работы

4 Выводы

Освоил работу с подпрограммами и отладчиком.