

Отчёта по лабораторной работе 10

Понятие подпрограммы. Отладчик GDB.

Чесноков Артемий Павлович НПИбд-02-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работы	22
4	Выводы	29

Список иллюстраций

2.1	Файл lab10-1.asm	7
2.2	Работа программы lab10-1.asm	8
2.3	Файл lab10-1.asm	9
2.4	Работа программы lab10-1.asm	10
2.5	Файл lab10-2.asm	11
2.6	Работа программы lab10-2.asm в отладчике	12
2.7	дисассимилированный код	13
2.8	дисассимилированный код в режиме интел	14
2.9	точка остановки	15
2.10	изменение регистров	16
2.11	изменение регистров	17
2.12	изменение значения переменной	18
2.13	вывод значения регистра	19
2.14	вывод значения регистра	20
2.15	вывод значения регистра	21
3.1	Файл lab10-4.asm	23
3.2	Работа программы lab10-4.asm	24
3.3	код с ошибкой	25
3.4	отладка	26
3.5	код исправлен	27
3.6	проверка работы	28

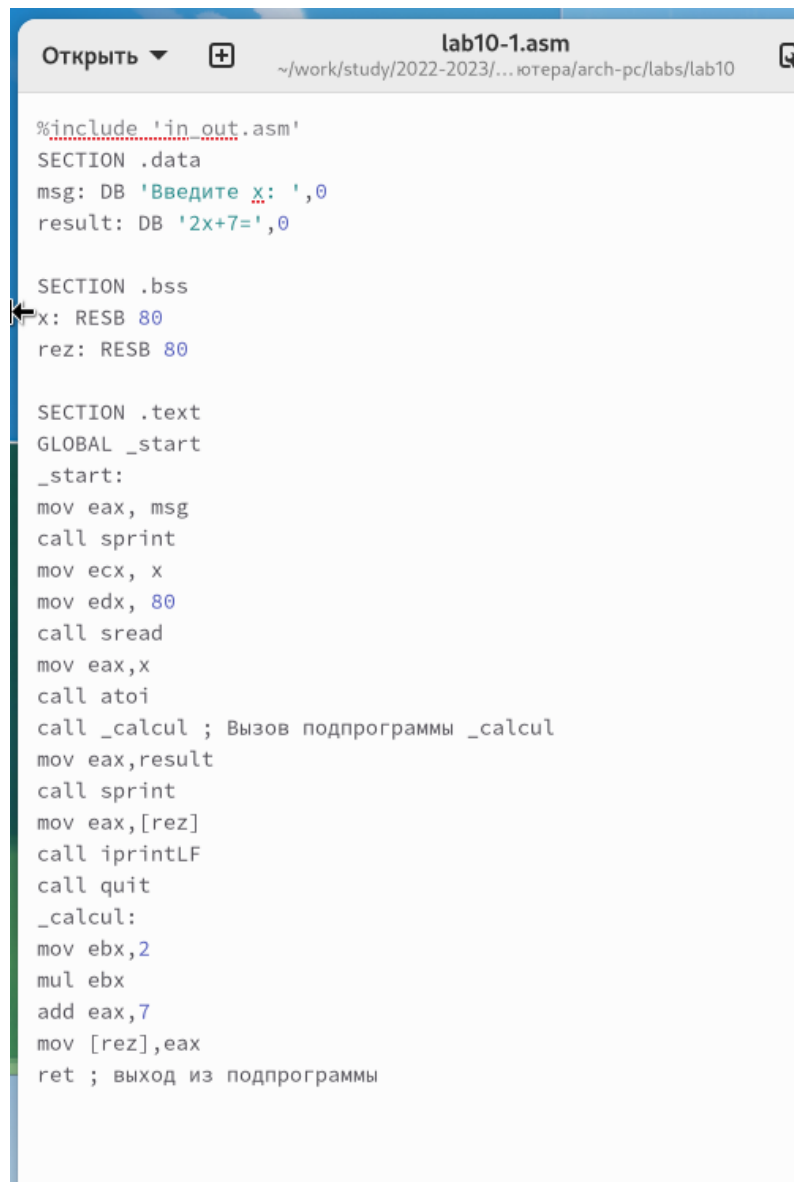
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создаим файл lab10-1.asm:
2. Введём в этот файл программу из лабораторной(Листинг 10.1). (рис. 2.1, 2.2)



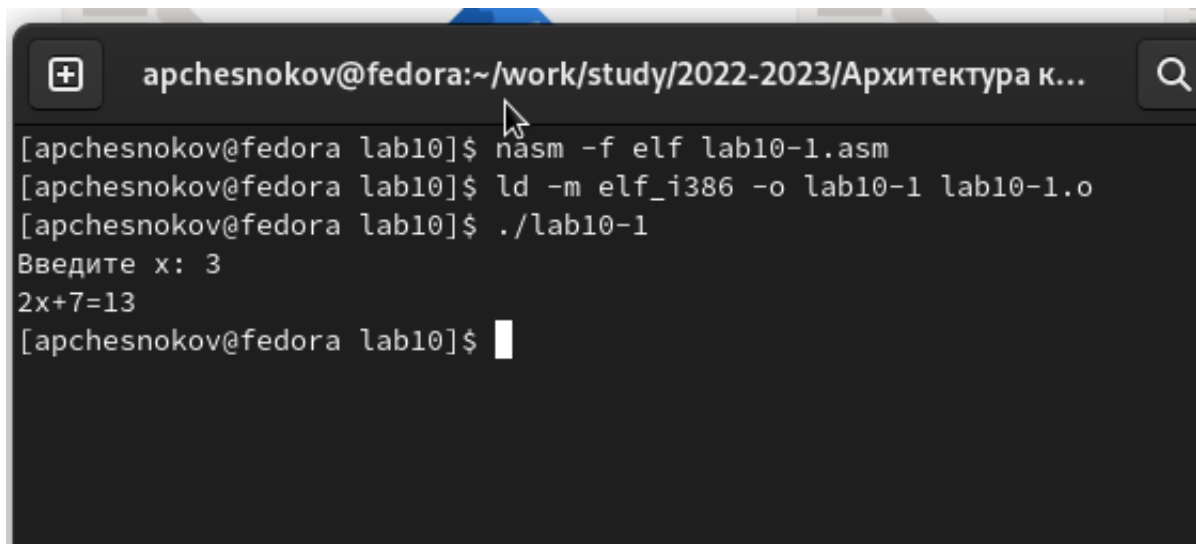
```
lab10-1.asm
~/work/study/2022-2023/... ютеpa/arch-pc/labs/lab10

%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

A terminal window with a dark background. The title bar shows the user 'archesnokov@fedora' and the current directory '~/work/study/2022-2023/Архитектура к...'. The terminal content shows the following commands and output:

```
[archesnokov@fedora lab10]$ nasm -f elf lab10-1.asm
[archesnokov@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[archesnokov@fedora lab10]$ ./lab10-1
Введите x: 3
2x+7=13
[archesnokov@fedora lab10]$
```

Рис. 2.2: Работа программы lab10-1.asm

3. Изменим текст программы, по задаче из лабораторной, чтобы вычислялось $f(g(x))$, где $f(x) = 2x + 7$, $g(x) = 3x - 1$ (рис. 2.3, 2.4)


```
Открыть + lab10-1.asm
~/work/study/2022-2023/...ютеpa/arch-pc/labs/lab10

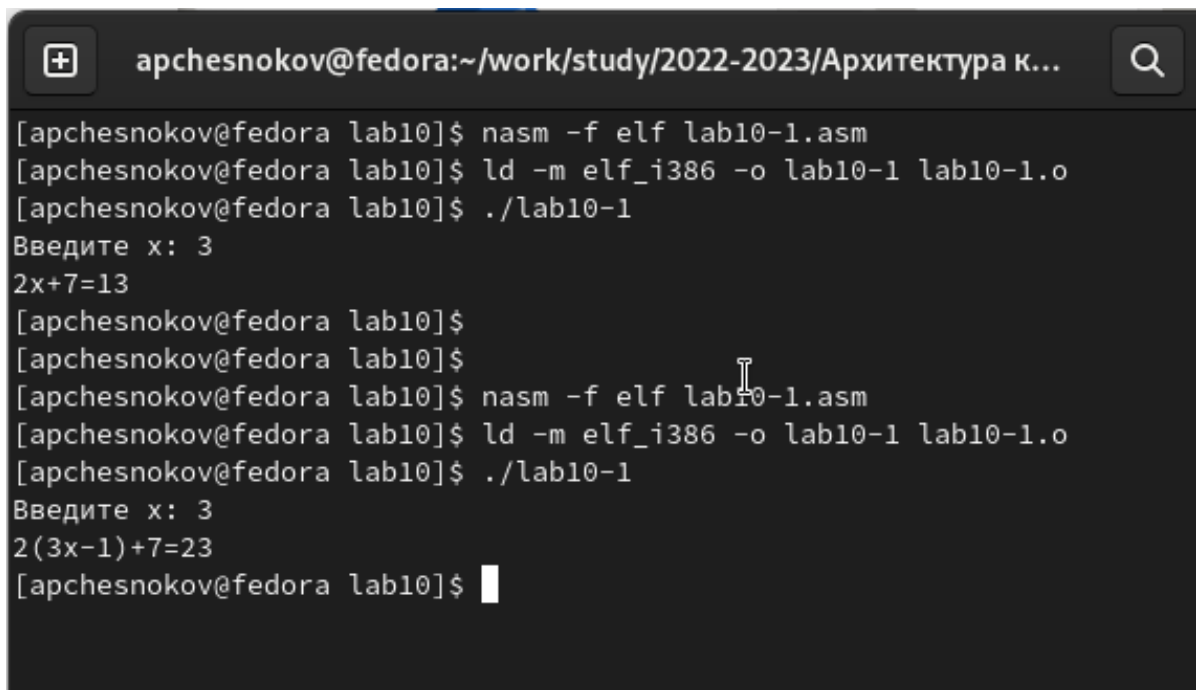
SECTION .bss
x: RESB 80
rez: RESB 80

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:
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
```

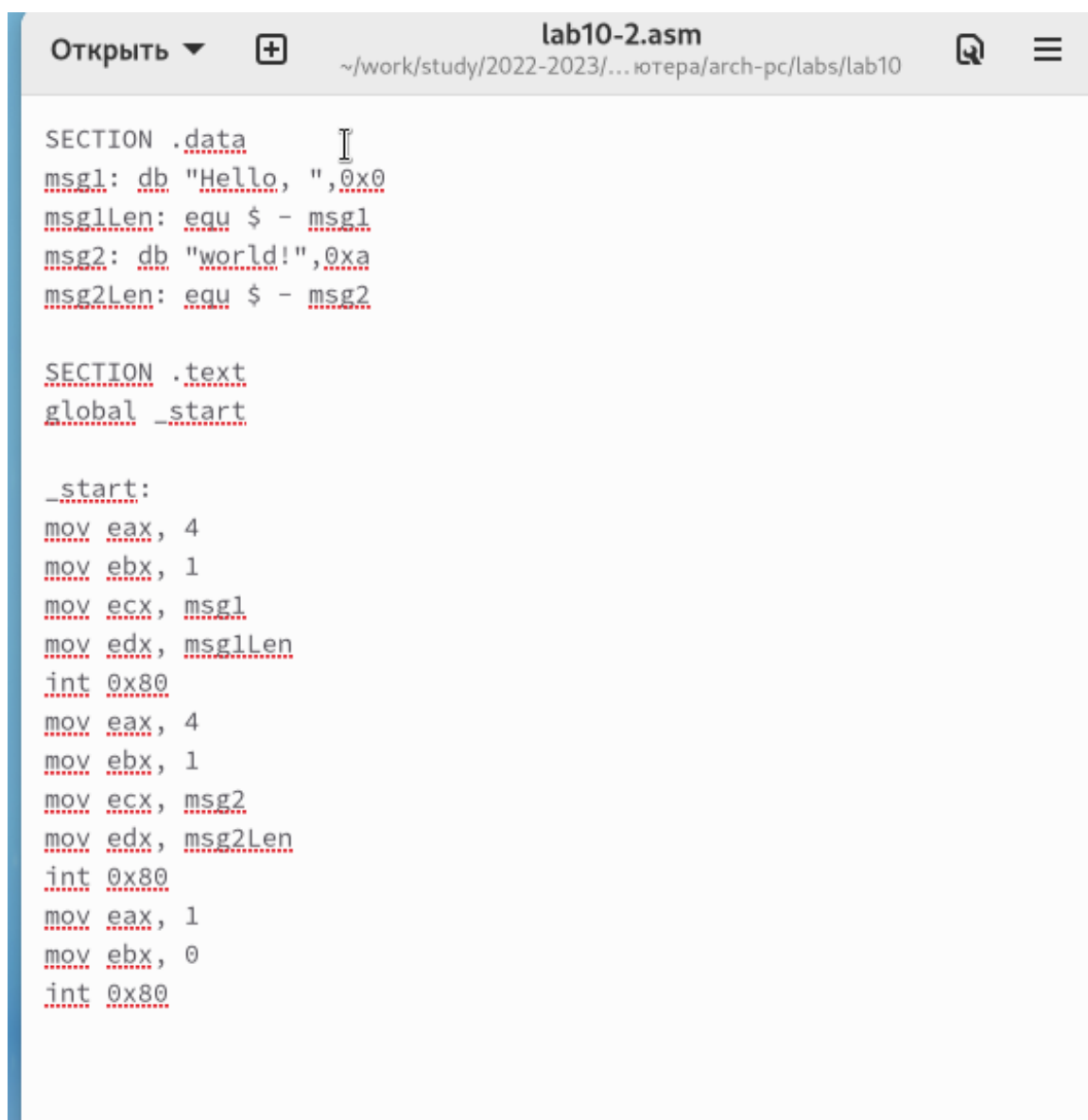
Рис. 2.3: Файл lab10-1.asm



```
archesnokov@fedora:~/work/study/2022-2023/Архитектура к...  
[archesnokov@fedora lab10]$ nasm -f elf lab10-1.asm  
[archesnokov@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o  
[archesnokov@fedora lab10]$ ./lab10-1  
Введите x: 3  
2x+7=13  
[archesnokov@fedora lab10]$  
[archesnokov@fedora lab10]$  
[archesnokov@fedora lab10]$ nasm -f elf lab10-1.asm  
[archesnokov@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o  
[archesnokov@fedora lab10]$ ./lab10-1  
Введите x: 3  
2(3x-1)+7=23  
[archesnokov@fedora lab10]$
```

Рис. 2.4: Работа программы lab10-1.asm

4. Создадим lab10-2.asm с текстом программы из Листинга 10.2: (рис. 2.5)



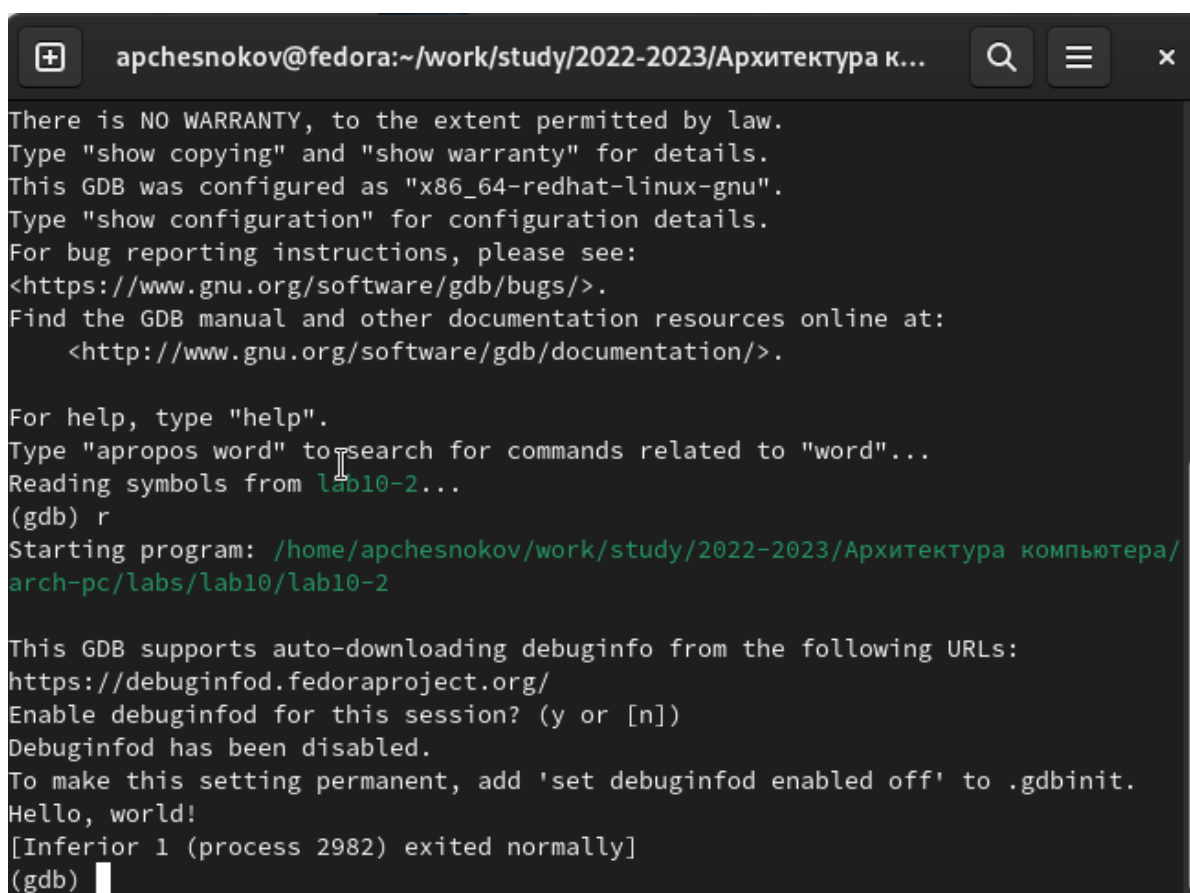
```
SECTION .data
msg1: db "Hello, ",0x0
msg1len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2

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

Проверьте работу программы, запустив ее в оболочке GDB:(рис. 2.6)

A screenshot of a terminal window with a dark background. The window title bar shows the user 'apchesnokov@fedora' and the path '~/work/study/2022-2023/Архитектура к...'. The terminal text shows a GDB session starting with a warning about no warranty, followed by configuration details and help instructions. The user enters 'r' to run the program, which is 'lab10-2' located at '/home/apchesnokov/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10/lab10-2'. The program outputs 'Hello, world!' and then '[Inferior 1 (process 2982) exited normally]'. The prompt '(gdb)' is visible at the bottom.

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура к...
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) r
Starting program: /home/apchesnokov/work/study/2022-2023/Архитектура компьютера/
arch-pc/labs/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 2982) exited normally]
(gdb) 
```

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

Для более подробного анализа программы установите брейкпоинт на метку start и запустим её. Так же посмотрим диасемблированный код (рис. 2.7, 2.8)

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура к...
No thread selected
(gdb) r
Starting program: /home/apchesnokov/work/study/2022-2023/Архитектура компьютера/
arch-pc/labs/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

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура к...
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

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверьте это с помощью команды `info breakpoints` (кратко `i b`) Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции (`mov ebx,0x0`) и установите точку.(рис. 2.9)

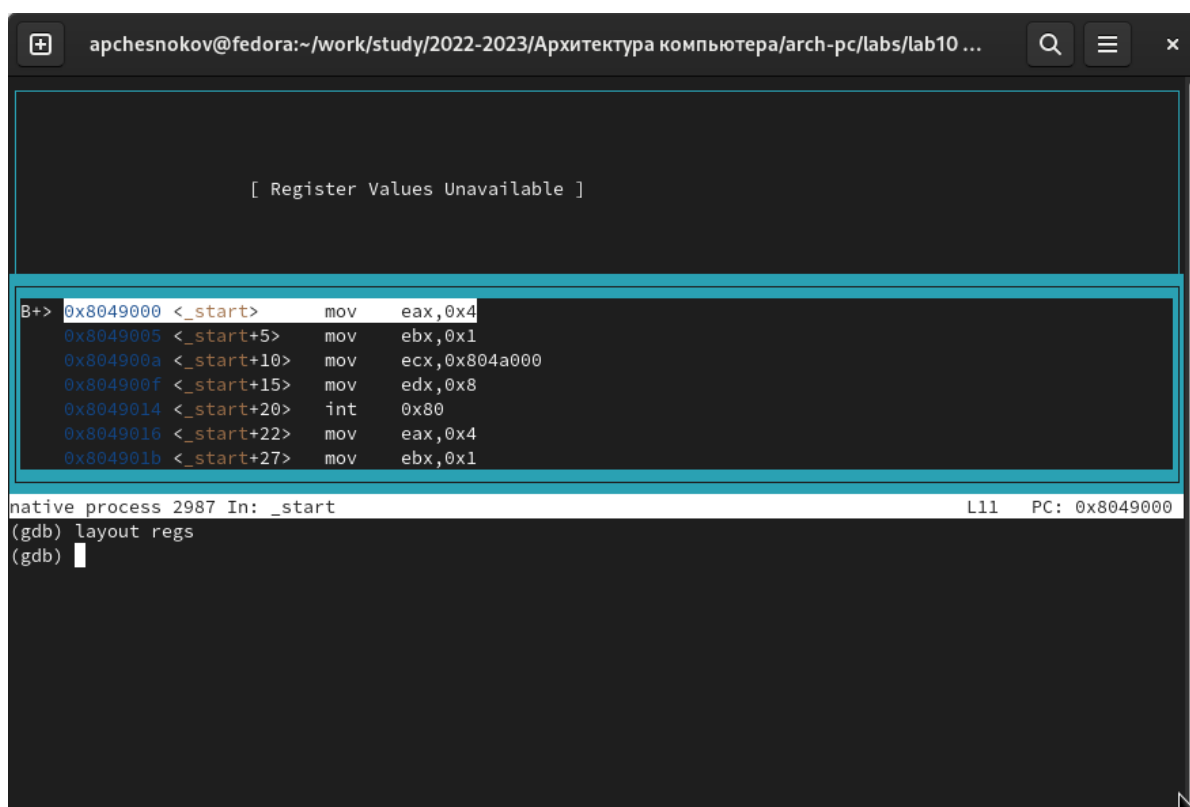


Рис. 2.9: точка остановки

Выполним инструкций с помощью команды `si` и проследим за изменением значений регистров. (рис. 2.11 2.12)

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd140 0xffffd140
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
0x804901b <_start+27>     mov     ebx,0x1

native process 2987 In: _start L12 PC: 0x8049005
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--siss 0x2b 43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) █
```

Рис. 2.10: изменение регистров


```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd140 0xffffd140
ebp      0x0      0
esi      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

native process 2987 In: _start L18 PC: 0x8049020
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.11: изменение регистров

Посмотрим значение переменной msg1 Посмотрим значение переменной msg2 Измените первый символ переменной msg1 Замените любой символ во второй переменной msg2. (рис. 2.12)

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd140 0xffffd140
ebp      0x0      0x0
esi      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

native process 2987 In: _start L18 PC: 0x8049020
(gdb) si
(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>: "Lorld!\n\034"
(gdb)
```

Рис. 2.12: изменение значения переменной

Выведете в различных форматах значение регистра edx. С помощью команды set изменим значение регистра ebx:(рис. 2.13)

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd140 0xffffd140
ebp      0x0      0
esi      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

native process 2987 In: _start L18 PC: 0x8049020
$2 = 100
(gdb) p/s $ecx
$3 = 134520832
(gdb) p/x $ecx
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) 
```

Рис. 2.13: вывод значения регистра

С помощью команды set изменим значение регистра ebx:(рис. 2.14)

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

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd140 0xffffd140
ebp      0x0      0x0
esi      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

native process 2987 In: _start L18 PC: 0x8049020
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(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. Скопируем файл lab9-2.asm. Создадим исполняемый файл. Загрузим исполняемый файл в отладчик, указав аргументы.

Для начала установим точку остановки перед первой инструкцией в программе и запустим ее.

Как видно, число аргументов равно 5 – это имя программы lab10-3 и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрим остальные позиции стека (рис. 2.15)

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
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 lab10-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) run
Starting program: /home/apchesnokov/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10/lab10-3

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
0xffffd140: 0x00000001
(gdb) x/s *(void**)(esp + 4)
0xffffd2e9: "/home/apchesnokov/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10/lab10-3"
(gdb)
```

Рис. 2.15: вывод значения регистра

шаг равен размеру переменной - 4 байтам., поэтому шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12])

3 Самостоятельная работы

1. Преобразуем программу из самостоятельного задания 1, лабораторной работы №9, реализовав вычисление значения функции $f(x)$ как подпрограмму.
(рис. 3.1 3.2)



```
Открыть ▾ + lab10-4.asm
~/work/study/2022-2023/... ютеpa/arch-пс/labs/lab10

msg db "Результат: ",0
fx: db 'f(x)=12x-7 ',0

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

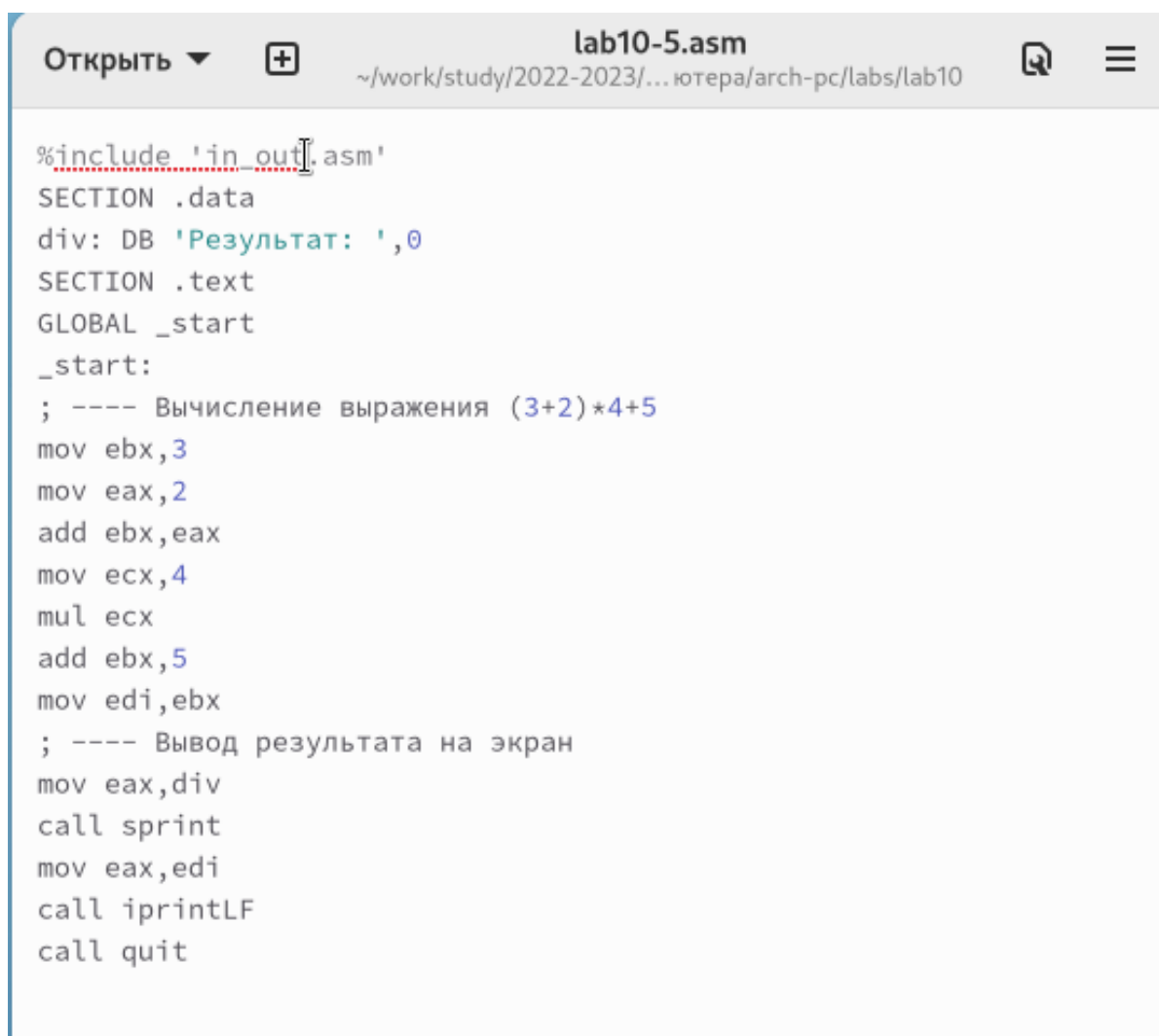
_calc:
mov ebx,12
mul ebx
sub eax,7
ret
```

Рис. 3.1: Файл lab10-4.asm

```
[archesnikov@fedora lab10]$  
[archesnikov@fedora lab10]$ nasm -f elf lab10-4.asm  
[archesnikov@fedora lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o  
[archesnikov@fedora lab10]$ ./lab10-4  
f(x)=12x-7  
Результат: 0  
[archesnikov@fedora lab10]$ ./lab10-4 1  
f(x)=12x-7  
Результат: 5  
[archesnikov@fedora lab10]$ ./lab10-4 1 2 3 4 5 6  
f(x)=12x-7  
Результат: 210  
[archesnikov@fedora lab10]$
```

Рис. 3.2: Работа программы lab10-4.asm

2. Скопируем из листинга программу, неверно рабочую. Проверим её с помощью отладчика GDB, определим ошибку и исправим ее.(рис. 3.3 3.4 3.5 3.6)



The screenshot shows a code editor window titled "lab10-5.asm". The address bar indicates the file path: "~/work/study/2022-2023/... ютера/arch-pc/labs/lab10". The code is as follows:

```
%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
```

A red dotted line underlines the line `%include 'in_out.asm'`, and a cursor is positioned at the end of this line, indicating a syntax error.

Рис. 3.3: код с ошибкой



```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd140 0xffffd140
ebp      0x0      0x0
esi      0x0      0

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490fe <_start+22>    mov     edi,ebx
0x8049100 <_start+24>    add     ebx,ea804a000
0x8049105 <_start+29>    call    0x804900f <sprint>
0x804910a <_start+34>    mul     eax,edi
0x804910c <_start+36>    call    0x8049086 <iprintf>
> 0x8049111 <_start+41>    call    0x80490db <quit>
                                04a000
                                rint>

native process 3129 In: _start L14 PC: 0x80490fe
Breakpoint 1: No process In: 10-5.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 10
[Inferior 1 (process 3129) exited normally]
(gdb)
```

Рис. 3.4: отладка

Перепутан порядок аргументов у инструкции add. Так же по окончании работы в edi отправляется ebx вместо eax. Исправим програму и запустим её. Убедимся в работоспособности

Открыть ▾  lab10-5.asm 

~/work/study/2022-2023/...ютера/arch-pc/labs/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

```
apchesnokov@fedora:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab10 ...
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd140 0xffffd140
ebp      0x0       0
esi      0x0       0

B+ 0x80490e8 <_start> mov ebx,0x3
0x80490fe <_start+22> mov edi,eax
0x8049100 <_start+24> add eax,eb804a000
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mul eax,edi
0x804910c <_start+36> call 0x8049086 <iprintLF>
> 0x8049111 <_start+41> call 0x80490db <quit>
                                04a000
                                rint>

native process 3174 In: _start L14 PC: 0x80490fe
Breakpoint 1: No process In: 10-5.asm:8 L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 3174) exited normally]
(gdb)
```

Рис. 3.6: проверка работы

4 Выводы

Приобрели навыки написания программ с использованием подпрограмм. Ознакомились с методами отладки при помощи GDB и его основными возможностями. https://github.com/Sinabon2004/study_2022-2023_arh-pc