

Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий самостоятельной работы	20
4	Выводы	25

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Работа с файлом lab10-1.asm	8
2.3	Работа с измененным файлом lab10-1.asm	10
2.4	Работа с файлом lab10-2.asm	11
2.5	Запуск программы lab10-2.asm	11
2.6	Брейкпоинт и дисассимилированный код программы . . .	12
2.7	Intel'овский синтаксис	13
2.8	Информация о брейкпоинте	14
2.9	Установка второго брейкпоинта, вывод информации . . .	14
2.10	Выполнение команды si	15
2.11	Содержимое регистров	15
2.12	Содержимое переменных и инструкции	16
2.13	Замена символов в переменных msg1 и msg2	16
2.14	Просмотр значений нескольких регистров	16
2.15	Просмотр значений регистра edx	17
2.16	Замена значения регистра ebx	17
2.17	Завершение работы программы	17
2.18	Работа с файлом lab10-3.asm	18
2.19	Просмотр регистра esp	18
2.20	Просмотр остальных позиций стека	19
3.1	Тестирование программы lab10my.asm	22
3.2	Убеждаемся в неработоспособности программы	24
3.3	Поиск ошибки	24
3.4	Успешная отработка 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 Выполнение лабораторной работы

Для начала создадим каталог для программ 10-ой лабораторной работы, перейдем в него и создадим файл lab10-1.asm (рис. 2.1).

```
[aamishina@fedora ~]$ mkdir ~/work/arch-pc/lab10
[aamishina@fedora ~]$ cd ~/work/arch-pc/lab10
[aamishina@fedora lab10]$ touch lab10-1.asm
[aamishina@fedora lab10]$
```

Рис. 2.1: Создание каталога и файла

Вводим текст программы из листинга 10.1 в наш файл. Создадим и запустим исполняемый файл, удостоверимся в его работе (рис. 2.2).

Программа lab10-1.asm:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rezs: 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
call sprint
mov eax, [res]
call iprintLF
call quit
;-----

; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

```

```

[aamishina@fedora lab10]$ nasm -f elf lab10-1.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[aamishina@fedora lab10]$ ./lab10-1
Введите x: 5
2x+7=17
[aamishina@fedora lab10]$

```

Рис. 2.2: Работа с файлом lab10-1.asm

Меняем текст программы, добавляя подпрограмму `_subcalcul` для вычисления выражения $f(g(x))$. Создаем исполняемый файл, программа отработывает успешно (рис. 2.3).

Измененная программа lab10-1.asm:

```

#include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

SECTION .bss
x: RESB 80
rez: 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
call sprint
mov eax,[rez]
call iprintLF
call quit

;-----
; Подпрограмма вычисления
; выражения "2x+7"
_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

```

```
[aamishina@fedora lab10]$ nasm -f elf lab10-1.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[aamishina@fedora lab10]$ ./lab10-1
Введите x: 5
2(3x-1)+7=35
[aamishina@fedora lab10]$
```

Рис. 2.3: Работа с измененным файлом lab10-1.asm

Теперь создадим файл lab10-2.asm, вводим в него текст листинга 10.2. Создаем исполняемый файл с ключом '-g' для работы с GDB. Загружаем исполняемый файл в отладчик gdb (рис. 2.4). Запускаем программу с помощью команды run, она отработывает успешно (рис. 2.5).

Программа lab10-2.asm:

```
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
```

```
[aamishina@fedora lab10]$ touch lab10-2.asm
[aamishina@fedora lab10]$ nasm -f elf -g -l lab10-2.lst lab10-2.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-2 lab10-2.o
[aamishina@fedora lab10]$ gdb lab10-2
GNU gdb (GDB) Fedora 11.2-3.fc36
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-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) 
```

Рис. 2.4: Работа с файлом lab10-2.asm

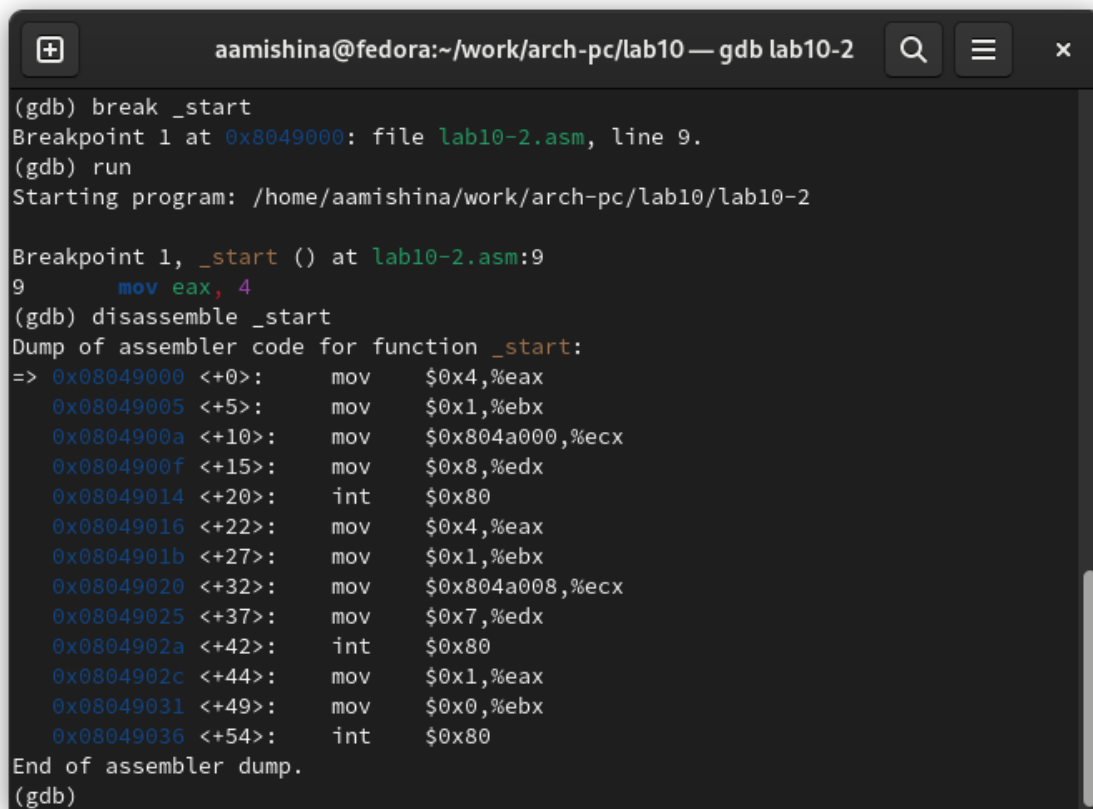
```
(gdb) run
Starting program: /home/aamishina/work/arch-pc/lab10/lab10-2

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 6207) exited normally]
(gdb) 
```

Рис. 2.5: Запуск программы lab10-2.asm

Устанавливаем брейкпоинт на метку `_start` и запускаем программу еще раз. Посмотрим на дисассимилированный код программы с помощью

команды `disassemble` начиная с метки `_start` (рис. 2.6). Переключимся на изображение с Intel'овским синтаксисом (рис. 2.7). Видим отличие в третьем столбце в изображении названий регистров и переменных. Также отличается порядок переменная-регистр и наоборот регистр-переменная.



```
aamishina@fedora:~/work/arch-pc/lab10 — gdb lab10-2
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab10-2.asm, line 9.
(gdb) run
Starting program: /home/aamishina/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:9
9      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. 2.6: Брейкпоинт и дисассимилированный код программы

```

(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. 2.7: Intel'овский синтаксис

Далее включаем режим псевдографики для более удобного анализа программы. С помощью команды `info breakpoints` смотрим, что уже установили одну точку остановки (рис. 2.8). Устанавливаем еще одну, используя адрес инструкции. Снова смотрим информацию о всех брейкпоинтах (рис. 2.9).

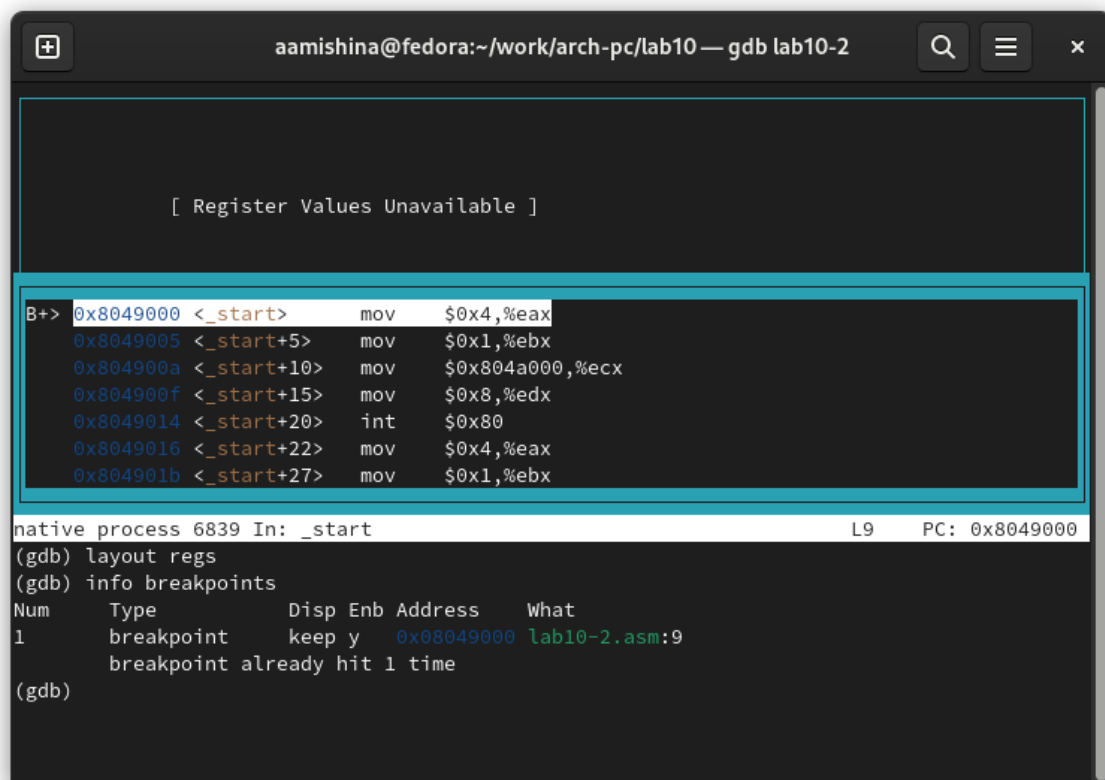


Рис. 2.8: Информация о брейкпоинте

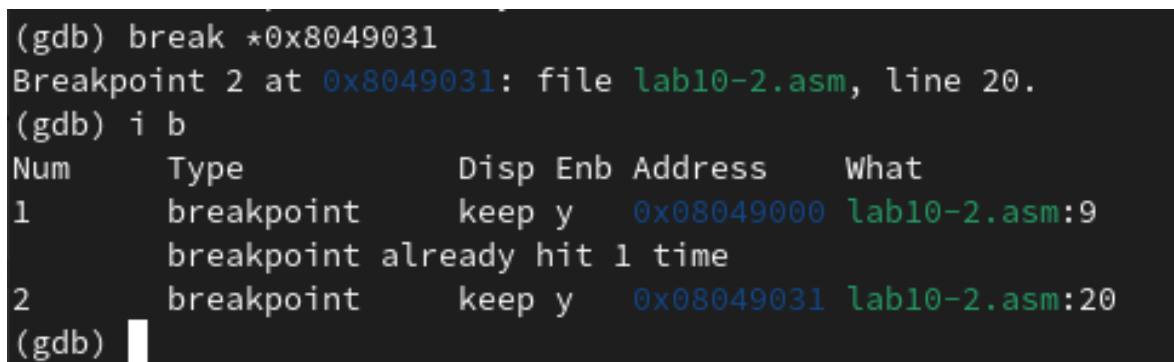


Рис. 2.9: Установка второго брейкпоинта, вывод информации

Далее выполняем ровно 5 раз команду `si`, видим изменения в регистрах `eax`, `ebx`, `ecx`, `edx`, `esp` (рис. 2.10). Смотрим содержимое регистров (рис. 2.11).

```

aamishina@fedora:~/work/arch-pc/lab10 — gdb lab10-2
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0

0x804900a <_start+10> mov $0x804a000,%ecx
0x804900f <_start+15> mov $0x8,%edx
0x8049014 <_start+20> int $0x80
> 0x8049016 <_start+22> mov $0x4,%eax
0x804901b <_start+27> mov $0x1,%ebx
0x8049020 <_start+32> mov $0x804a008,%ecx
0x8049025 <_start+37> mov $0x7,%edx

native process 6839 In: _start L14 PC: 0x8049016
1 breakpoint keep y 0x08049000 lab10-2.asm:9
  breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab10-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.10: Выполнение команды si

```

6      0x8049016 <_start+22>
eflags      0x202      [ IF ]
cs          0x23      35
ss          0x2b      43
ds          0x2b      43
es          0x2b      43
fs          0x0      0
gs          0x0      0
(gdb)

```

Рис. 2.11: Содержимое регистров

Теперь посмотрим значения переменных msg1 по имени и msg2 по адресу. Также смотрим инструкцию mov ecx, msg2 (рис. 2.12).

```

(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) x/1sb 0x8049020
0x8049020 <_start+32>:  "\271\b\240\004\b\272\a"
(gdb)

```

Рис. 2.12: Содержимое переменных и инструкции

Меняем символы в переменных msg1 и msg2 (рис. 2.13).

```

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

```

Рис. 2.13: Замена символов в переменных msg1 и msg2

Смотрим значения нескольких регистров (рис. 2.14). Самостоятельно смотрим значение регистра edx в шестнадцатеричном формате, в двоичном формате и в символьном виде (рис. 2.15).

```

(gdb) p/s $eax
$4 = 8
(gdb) p/t $eax
$5 = 1000
(gdb) p/s $ecx
$6 = 134520832
(gdb) p/x $ecx
$7 = 0x804a000
(gdb)

```

Рис. 2.14: Просмотр значений нескольких регистров


```
(gdb) p/s $edx
$8 = 8
(gdb) p/t $edx
$9 = 1000
(gdb) p/x $edx
$10 = 0x8
(gdb)
```

Рис. 2.15: Просмотр значений регистра edx

С помощью команды set меняем значение регистра ebx. Сначала выводится 50, так как это ASCII кодировка символа “2”, а затем 2 - уже как число (рис. 2.16).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$11 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$12 = 2
(gdb)
```

Рис. 2.16: Замена значения регистра ebx

Завершаем выполнение программы (рис. 2.17).

```
(gdb) c
Continuing.
World!

Breakpoint 2, _start () at lab10-2.asm:20
(gdb) c
Continuing.
[Inferior 1 (process 6839) exited normally]
(gdb)
```

Рис. 2.17: Завершение работы программы

Копируем файл lab9-2.asm в файл с именем lab10-3.asm. Создаем исполняемый файл с отладчиком gdb. Загружаем исполняемый файл в отладчик, указав аргументы (рис. 2.18).

```
[aamishina@fedora lab10]$ cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm
[aamishina@fedora lab10]$ nasm -f elf -g -l lab10-3.lst lab10-3.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-3 lab10-3.o
[aamishina@fedora lab10]$ gdb --args lab10-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora 11.2-3.fc36
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-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)
```

Рис. 2.18: Работа с файлом lab10-3.asm

Устанавливаем точку останова перед первой инструкцией в программе и запускаем ее. Смотрим, что хранится в регистре esp, получаем 5 аргументов (рис. 2.19).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 8.
(gdb) run
Starting program: /home/aamishina/work/arch-pc/lab10/lab10-3 аргумент1 аргумент 2 аргумент\ 3

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

Breakpoint 1, _start () at lab10-3.asm:8
8      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd170:    0x00000005
(gdb)
```

Рис. 2.19: Просмотр регистра esp

Теперь посмотрим остальные позиции стека (рис. 2.20). Шаг изменения равен 4, потому что шаг равен размеру переменной (4 байта).

```
(gdb) x/s *(void**)(esp+4)
0xffffd321:    "/home/aamishina/work/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp+8)
0xffffd34c:    "аргумент1"
(gdb) x/s *(void**)(esp+12)
0xffffd35e:    "аргумент"
(gdb) x/s *(void**)(esp+16)
0xffffd36f:    "2"
(gdb) x/s *(void**)(esp+20)
0xffffd371:    "аргумент 3"
(gdb) x/s *(void**)(esp+24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.20: Просмотр остальных позиций стека

3 Выполнение заданий самостоятельной работы

Для начала дорабатываем программу из 9-ой лабораторной работы. Добавляем подпрограмму для вычисления значения функции (рис. 3.1).

Программа lab10my.asm:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0
func: db 'f(x) = 12x - 7', 0

SECTION .text
global _start
_start:
mov eax, func
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

```

```

[aamishina@fedora lab10]$ nasm -f elf lab10my.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10my lab10my.o
[aamishina@fedora lab10]$ ./lab10my
f(x) = 12x - 7
Результат: 0
[aamishina@fedora lab10]$ ./lab10my 4 2 6 7
f(x) = 12x - 7
Результат: 200
[aamishina@fedora lab10]$

```

Рис. 3.1: Тестирование программы lab10my.asm

Вторым заданием было определить ошибку в программе, используя отладчик GDB и анализируя изменение значений регистров. Стало понятно, что в нескольких местах следует поменять регистр `ebx` на `eax`. Измененная программа отработала успешно (рис. 3.2), (рис. 3.3), (рис. 3.4).

Нерабочий код 10.3:

```

#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

```

Исправленный мной код:

```

#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

```

```
[aamishina@fedora lab10]$ touch lab10-4.asm
[aamishina@fedora lab10]$ nasm -f elf lab10-4.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o
[aamishina@fedora lab10]$ ./lab10-4
Результат: 10
```

Рис. 3.2: Убеждаемся в неработоспособности программы

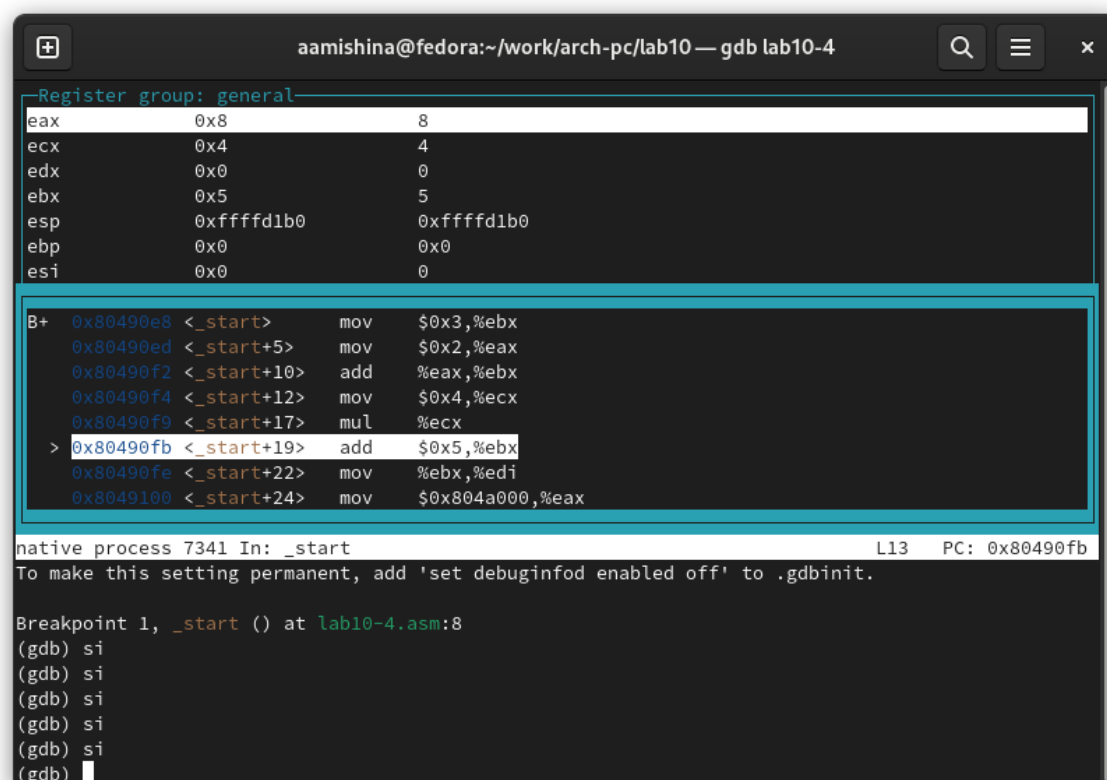


Рис. 3.3: Поиск ошибки

```
[aamishina@fedora lab10]$ nasm -f elf lab10-4.asm
[aamishina@fedora lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o
[aamishina@fedora lab10]$ ./lab10-4
Результат: 25
[aamishina@fedora lab10]$
```

Рис. 3.4: Успешная отработка программы

4 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм. Также я познакомилась с методами отладки при помощи GDB и его основными возможностями. Вся моя работа была записана и прокомментирована мной в данной лабораторной.