

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Лысенко Маргарита Олеговна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выполнение самостоятельной работы	14
6	Выводы	18

Список иллюстраций

4.1	Запуск программы	8
4.2	Запуск программы	8
4.3	Проверка программы	9
4.4	Подробный анализ	9
4.5	Просмотр кода	10
4.6	Переключение команд	10
4.7	Режим псевдографики	11
4.8	Проверка установки точки. Установка второй	11
4.9	Просмотр регистров	12
4.10	Просмотр значений	12
4.11	Изменение символов	12
4.12	Разные форматы	13
4.13	Загрузка файла в отладчик	13
4.14	Просмотр позиции стека	13
5.1	Преобразование программы	14
5.2	Поиск ошибок	16
5.3	Проверка программы	16

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

С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправить ее.

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

Создала каталог для выполнения лабораторной работы № 9, перешла в него и создала файл lab09-1.asm. Ввела в файл lab09-1.asm текст программы из листинга 9.1. Создала исполняемый файл и проверила его работу. (рис. 4.1).

```
molihsenko@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
molihsenko@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab09-1.o
molihsenko@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 4
2x+7=15
```

Рис. 4.1: Запуск программы

Изменила текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. (рис. 4.2).

```
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab09-1.o
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ./lab9-1
Введите x: 2
f(g(x))=6
```

Рис. 4.2: Запуск программы

Создала файл lab09-2.asm. Загрузила исполняемый файл в отладчик gdb. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 4.3).


```

molihsenko@dk8n74 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab9-2.asm
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab9-2.o
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
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-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/m/o/molihsenko/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4596) exited normally]
(gdb)

```

Рис. 4.3: Проверка программы

Для более подробного анализа программы установила брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустила её. (рис. 4.4).

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/o/molihsenko/work/arch-pc/lab09/lab09-2

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

```

Рис. 4.4: Подробный анализ

Посмотрела дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. 4.5).

```
(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.
```

Рис. 4.5: Просмотр кода

Переключилась на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 4.6).

```
(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. 4.6: Переключение команд

Включила режим псевдографики для более удобного анализа программы (рис. 4.7).

```

[ 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

native process 4626 In: _start L9 PC: 0x8049000
(gdb) Quit
(gdb) layout regs
(gdb) █

```

Рис. 4.7: Режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверила это с помощью команды `info breakpoints`. Установила еще одну точку останова по адресу инструкции.(рис. 4.8).

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc310 0xfffffc310
ebp      0x0      0x0

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 4797 In: _start L9 PC: 0x8049000
Breakpoint 3 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y   0x08049000 lab9-2.asm:9
       breakpoint already hit 1 time
2      breakpoint      keep y   0x08049000 lab9-2.asm:9
       breakpoint already hit 1 time
3      breakpoint      keep y   0x08049031 lab9-2.asm:20
(gdb) █

```

Рис. 4.8: Проверка установки точки. Установка второй

Посмотрела содержимое регистров также можно с помощью команды `info registers`(рис. 4.9).

```

native process 4797 In: _start
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc310 0xffffc310
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.9: Просмотр регистров

Посмотрите значение переменной `msg1` по имени и переменной `msg2` по адресу. (рис. 4.10).

```

(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a000
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 4.10: Просмотр значений

Изменила первый символ переменной `msg1`. Заменяла любой символ во второй переменной `msg2` (рис. 4.11).

```

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

```

Рис. 4.11: Изменение символов

Вывела в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра `edx`. С помощью команды `set` изменила значение регистра `ebx`. (рис. 4.12).

```

(gdb) p/c $ebx
$1 = 50 '2'
(gdb) p/t
$2 = 110010
(gdb) p/s
$3 = 50
(gdb) p/x
$4 = 0x32

```

Рис. 4.12: Разные форматы

Скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm. Создала исполняемый файл. Загрузила исполняемый файл в отладчик, указав аргументы (рис. 4.13).

```

molihsenko@dk8n74 ~/work/arch-pc/lab09 $ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'

```

Рис. 4.13: Загрузка файла в отладчик

Для начала установила точку останова перед первой инструкцией в программе и запустила ее. Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки. Число аргументов равно 5 – это имя программы lab09-3. Посмотрела остальные позиции стека – по адресу [esp+4] (рис. 4.14).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/o/molihsenko/work/arch-pc/lab09/lab09-3 а
ргумент1 аргумент 2 аргумент\ 3

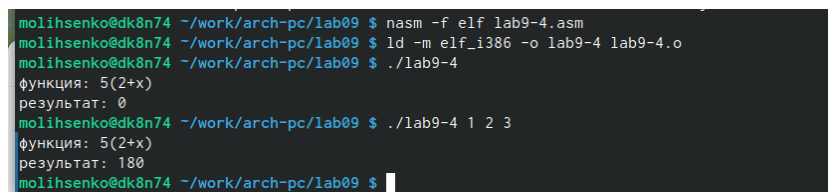
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xfffffc20: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xfffffc55e: "/afs/.dk.sci.pfu.edu.ru/home/m/o/molihsenko/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc5a5: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc5b7: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xfffffc5c8: "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc5ca: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 4.14: Просмотр позиции стека

5 Выполнение самостоятельной работы

Преобразовала программу из лабораторной работы №8, реализовав вычислительные значения функции $f(x)$ как подпрограмму (рис. 5.1).



```
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ./lab9-4
функция: 5(2+x)
результат: 0
molihsenko@dk8n74 ~/work/arch-pc/lab09 $ ./lab9-4 1 2 3
функция: 5(2+x)
результат: 180
molihsenko@dk8n74 ~/work/arch-pc/lab09 $
```

Рис. 5.1: Преобразование программы

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
f_x db "функция: 5(2+x)",0h
```

```
msg db 10,13,'результат: ',0h
```

```
SECTION .text
```

```
global _start
```

```
_f:
```

```
push ebx
```

```
dec eax
```

```
mov ebx, 10
```

```
mul ebx
```

```
pop ebx
```

```
ret
```

```
_start:
```

```
pop ecx
```

```
pop edx
```

```
sub ecx,1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx,0h
```

```
jz _end
```

```
pop eax
```

```
call atoi
```

```
call _f
```

```
add eax,2
```

```
mov ebx, 5
```

```
mul ebx
```

```
add esi, eax
```

```
loop next
```

```
_end:
```

```
mov eax, f_x
```

```
call sprint
```

```
mov eax, msg
```

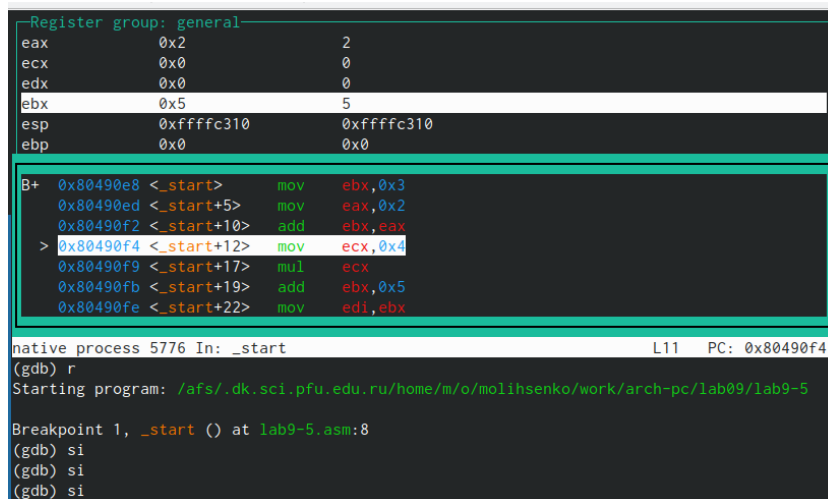
```
call sprint
```

```
mov eax, esi
```

```
call iprintLF
```

```
call quit
```

При запуске данная программа дает неверный результат. Получается 10. С помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку и исправила ее. (рис. 5.2).



```
Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x5      5
esp      0xfffffc310 0xfffffc310
ebp      0x0      0x0

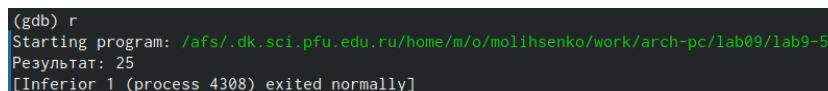
B+ 0x80490e8 <_start>    mov    ebx,0x3
0x80490ed <_start+5>    mov    eax,0x2
0x80490f2 <_start+10>   add    ebx,eax
> 0x80490f4 <_start+12> mov    ecx,0x4
0x80490f9 <_start+17>   mul    ecx
0x80490fb <_start+19>   add    ebx,0x5
0x80490fe <_start+22>   mov    edi,ebx

native process 5776 In: _start L11 PC: 0x80490f4
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/o/molihsenko/work/arch-pc/lab09/lab9-5

Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
```

Рис. 5.2: Поиск ошибок

Проверила работу исправленной программы. Ошибки в строчках: add ebx, eax mov ecx,4 mul ecx add ebx,5 mov edi,ebx (рис. 5.3).



```
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/o/molihsenko/work/arch-pc/lab09/lab9-5
Результат: 25
[Inferior 1 (process 4308) exited normally]
```

Рис. 5.3: Проверка программы

```
%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
```

6 Выводы

В ходе выполнения работ я приобрела навыки написания программ с использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.