# Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера Алексеев Илья Сергеевич

Содержание

Список иллюстраций

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

- 1. Реализация подпрограмм в NASM
- 2. Отладка программ с помощью GDB
- 3. Самостоятельное выполнение заданий по материалам лабораторной работы

### 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 пре- рывание выполнения

программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить доволь- но 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

#### 4.1 Релазиация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9 (рис. 1).

```
vboxuser@rabot:-$ cd work/study/2024-2025/Архитектура\ компьютера/arch-pc/
vboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdr lab09
vboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc$ touch lab9-1.asm
vboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 1: Создание рабочего каталога

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 2).

```
vboxuser@rabot:-/work/study/2024-2025/Aρχωττοκτγρα κοκπωστερα/arch-pc/lab09$ nasm -f elf lab9-1.asm vboxuser@rabot:-/work/study/2024-2025/Aρχωττοκτγρα κοκπωστερα/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o vboxuser@rabot:-/work/study/2024-2025/Aρχωτοκτγρα κοκπωστερα/arch-pc/lab09$ ./lab9-1
Beeдure x: 10
2x+7=27
```

Рис. 2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения f(g(x)) (рис. 3).

```
wboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab9-1.asm vboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o vboxuser@rabot:-/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab9-1
Введите х: 10
2(3x-1)x7=65
```

Puc. 3: Изменение программы первого листинга

#### Код программы:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите х: ', 0
result: DB '2(3x-1)+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
call atoi
call _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
_calcul:
push eax
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
pop eax
ret
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

### 4.1.1 Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 4).

```
Vboxuser@rabot:-/work/study/2024-2025/ApxurekTypa KommbuTepa/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-9ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLV3+: GNU GPL version 3 or later <a href="https://gnu.org/licenses/gpl.html">https://gnu.org/licenses/gpl.html</a>
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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<a href="https://www.gnu.org/software/gdb/bugs/">https://www.gnu.org/software/gdb/bugs/</a>>
Find the GDB manual and other documentation resources online at:
<a href="https://www.gnu.org/software/gdb/documentation/">https://www.gnu.org/software/gdb/documentation/</a>>
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) 

### Type "apropos word" to search for commands related to "word"...
```

Рис. 4: Запуск программы в отладчике

Запустив программу командой run, я убедился в том, что она работает исправно (рис. 5).

Puc. 5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку \_start и снова запускаю отладку (рис. 6).

```
(gdb) run
Starting program: /home/vboxuser/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 346294) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/vboxuser/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm;9

mov eax, 4
(gdb) [
```

Puc. 6: Запуск отладичка с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel *амд топчик* (рис. 7).

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ах, еах, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```
(gdb) disassemble start
Dump of assembler code for function start:
=> 0x08049000 <+0>:
   0x08049005 <+5>:
   0x0804900a <+10>:
                              $0x804a000, %ecx
   0x0804900f <+15>:
                              $0x8,%edx
   0x08049014 <+20>:
   0x08049016 <+22>:
   0x0804901b <+27>:
                              $0x804a008, %ecx
  0x08049020 <+32>:
  0x08049025 <+37>:
   0x0804902a <+42>:
   0x0804902c <+44>:
   0x08049031 <+49>:
   0x08049036 <+54>:
End of assembler dump.
(gdb) set disassemble-flavor intel
No symbol "disassemble" in current context.
(gdb) set disassembly-flavor intel
(gdb) disassemble start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:
   0x08049005 <+5>:
                                 ,0x1
   0x0804900a <+10>:
                                 ,0x804a000
   0x0804900f <+15>:
                                  ,0x8
   0x08049014 <+20>:
   0x08049016 <+22>:
   0x0804901b <+27>:
                                 ,0x1
   0x08049020 <+32>:
   0x08049025 <+37>:
                                 ,0x7
   0x0804902a <+42>:
   0x0804902c <+44>:
     (08049031 <+49>:
```

Рис. 7: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 8).

Рис. 8: Режим псевдографики

#### 4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).

Рис. 9: Список брейкпоинтов

Устаналиваю еще одну точку останова по адресу инструкции (рис. 10).

Рис. 10: Добавление второй точки останова

#### 4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. 11).

```
| Comparison | Com
```

Рис. 11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 12).

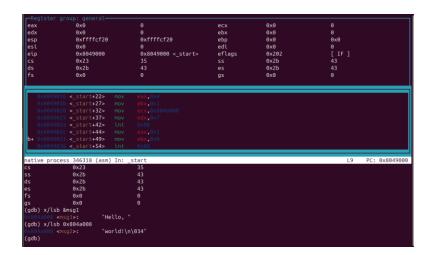


Рис. 12: Просмотр содержимого переменных двумя способами Меняю содержимое переменных по имени и по адресу (рис. 13).

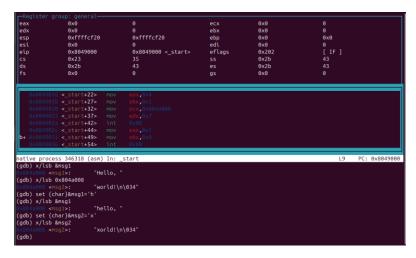


Рис. 13: Изменение содержимого переменных двумя способами Вывожу в различных форматах значение регистра edx (рис. 14).

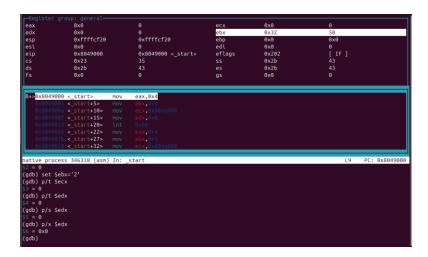
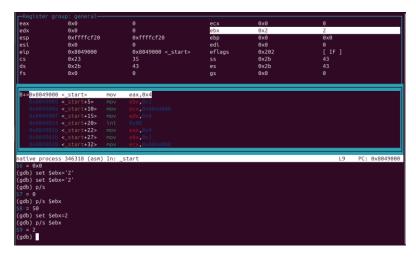


Рис. 14: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. 15).



Puc. 15: Примеры использования команды set

#### 4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и и создаю исполняемый файл с файлом листинга и отладки (рис. 16).

```
Vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa kommwerepa/arch-pc/lab095 cp -/work/study/2024-2025/Apxwrenrypa\ kommwerepa/arch-pc/lab08/lab0-3.asm vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa\ kommwerepa/arch-pc/lab09/lab0-3.asm vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa\ kommwerepa/arch-pc/lab095 nasm -f elf -g -l lab09-3.lst lab09-3.asm nasm: fatal: unable to open input file 'lab09-3.asm' No such file or directory vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa xommwerepa/arch-pc/lab095 ls in_out.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-1.asm lab0-3.asm vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa xommwerepa/arch-pc/lab095 nasm -f elf -g -l lab09-3.lst lab0-3.asm vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa xommwerepa/arch-pc/lab095 id -n elf_1306 -o lab0-3 lab0-3.osm vboxuser@rabot: /work/study/2024-2025/Apxwrenrypa/xommwerepa/arch-pc/lab095 id -n elf_1306 -o lab0-3 lab0-3.osm
```

Рис. 16: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а

указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились. (рис. 17).

Рис. 17: Проверка работы стека

#### 4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 18).

```
File Edit Search View Document Help

Kinclude 'in_out.asm'

SECTION .data

msg_func db "θyнкция: f(x) = 10x - 4", 0

msg_result db "PeaynbTaT: ", 0

SECTION .text

GLOBAL_start

_start:

_start:

mov eax, msg_func

call sprintlF

pop ecx
pop edx
sub ecx, 1
mov est, 0

next:
cmp ecx, 6h
jz _end
pop eax
call atoi

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
```

Рис. 18: Измененная программа предыдущей лабораторной работы

#### Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start
```

```
_start:
mov eax, msg_func
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, ∅
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
call _calculate_fx
add esi, eax
loop next
end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4
```

2. Запускаю программу в режике отладичка и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul есх можно заметить, что результат умножения записывается в регистр еах, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. 19).

Рис. 19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 20).

```
Vboxuser@rabot:-/work/study/2024-2025/Архитектура коминаетера/arch-pc/lab095 in lab9-5.asm vboxuser@rabot:-/work/study/2024-2025/Архитектура коминаетера/arch-pc/lab095 nasm.rf elf lab9-5.asm vboxuser@rabot:-/work/study/2024-2025/Архитектура коминаетера/arch-pc/lab095 ld -m elf_1386 -o lab9-5 lab9-5.o vboxuser@rabot:-/work/study/2024-2025/Архитектура коминаетера/arch-pc/lab095 ./lab9-5
Pezymbara: 25
```

Рис. 20: Проверка корректировок в программме

Код измененной программы:

```
%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 eax, ebx
mov ecx, 4
mul ecx
add eax, 5
mov edi, eax
mov eax, div
call sprint
mov eax, edi
call iprintLF
```

## 5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при поомщи GDB и его основными возможностями.

## 6 Список литературы

- 1. Курс на ТУИС
- 2. Лабораторная работа №9
- 3. Программирование на языке ассемблера NASM Столяров А. В.