РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра физико-математических и естественных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Попова Алиса Владимировна

Группа: НПИ-03-25

МОСКВА

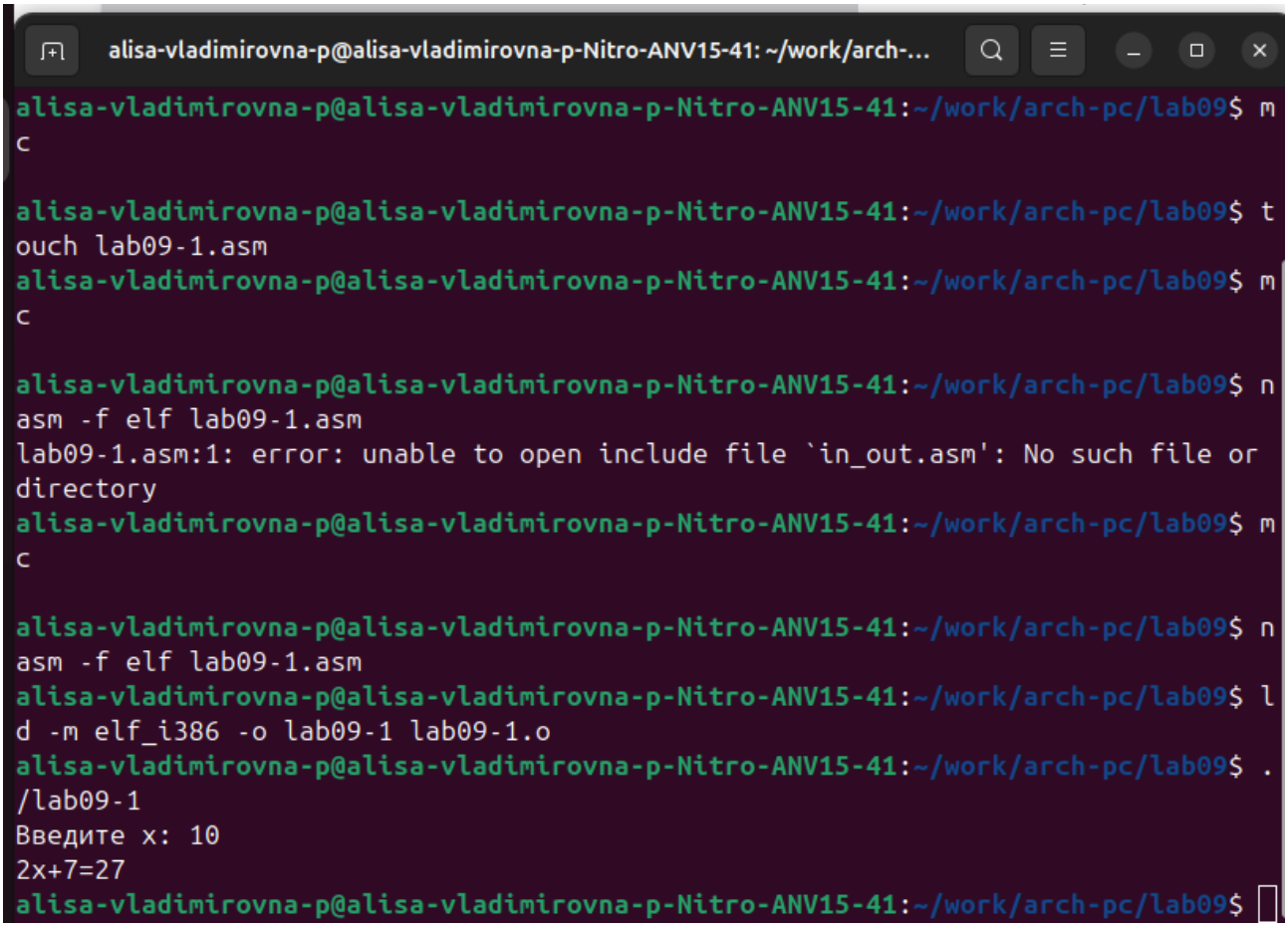
2025г.

Цель

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

Задание 1

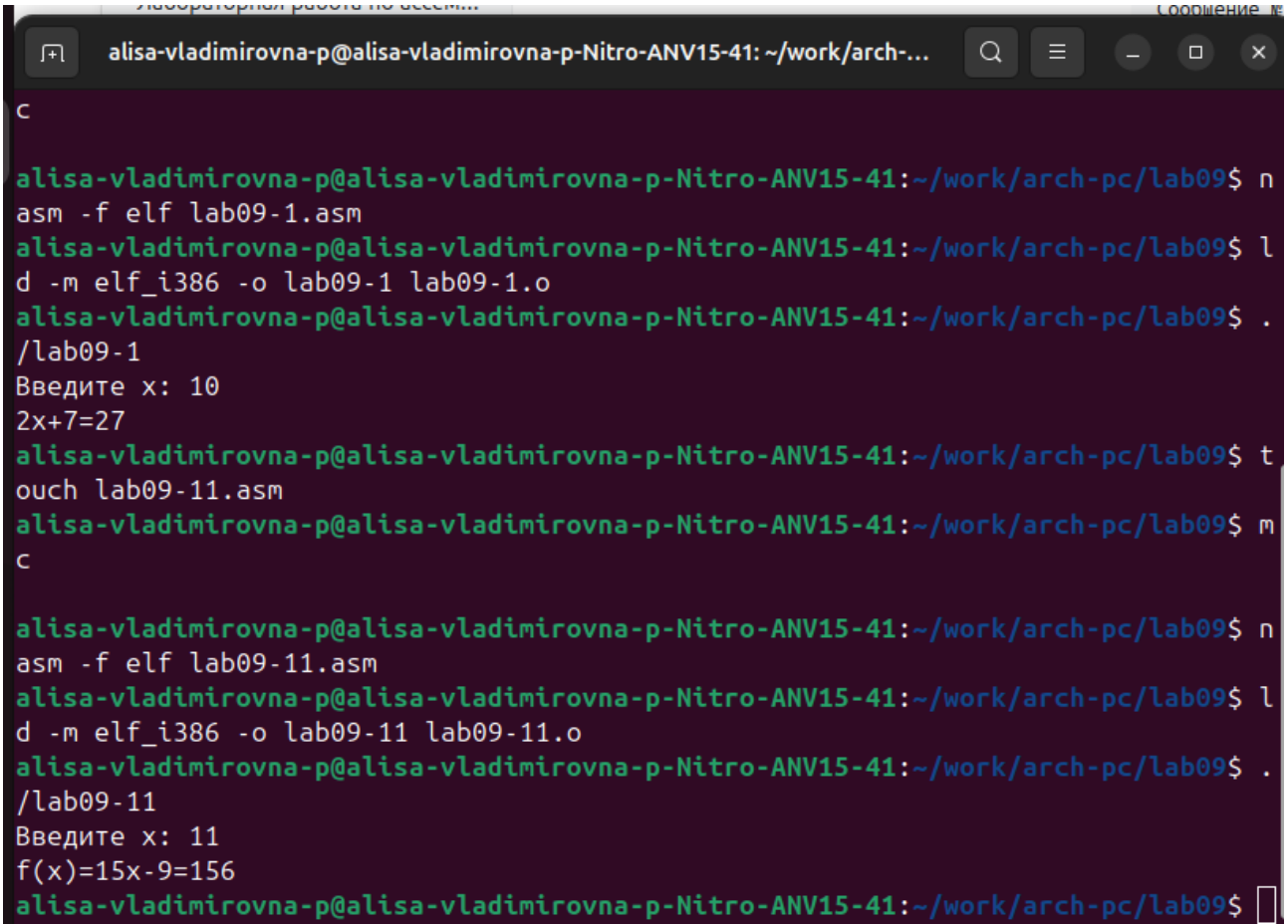
1. Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте файл lab09-1.asm
2. В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы _calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Внимательно изучите текст программы



Измените текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul,
для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) =$
$3x - 1$. Т.е. x передается в подпрограмму _calcul из нее в подпрограмму _subcalcul, где
вычисляется выражение $g(x)$, результат возвращается в _calcul и вычисляется выражение
$f(g(x))$. Результат возвращается в основную программу для вывода результата на экран

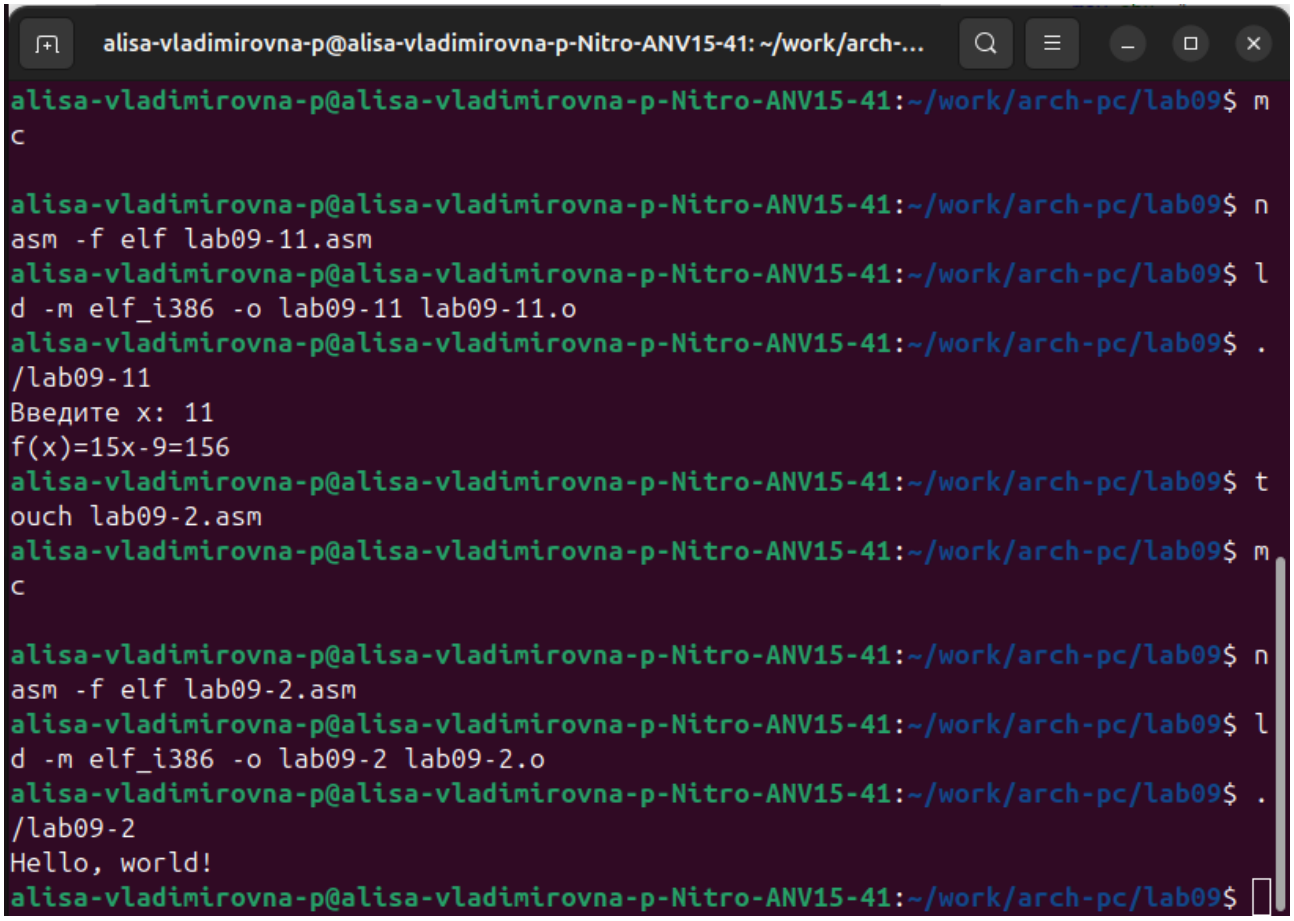
```
%include 'in_out.asm'

SECTION .data
    msg db 'Введите x: ',0
    result db 'f(x)=15x-9=',0
```

```
SECTION .bss
    x resb 10
    res resd 1

SECTION .text
global _start

_start:
    ; Ввод x
    mov eax, msg
    call sprint

    mov ecx, x
    mov edx, 10
    call sread

    mov eax, x
    call atoi          ; eax = x

    ; Вызов подпрограммы
    call calc_fx       ; eax = f(x)

    ; Вывод результата
    mov [res], eax
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit

; --- Подпрограмма вычисления f(x)=15x-9 ---
calc_fx:
    imul eax, 15       ; eax = 15*x
    sub eax, 9         ; eax = 15x - 9
    ret
```



Задание 2

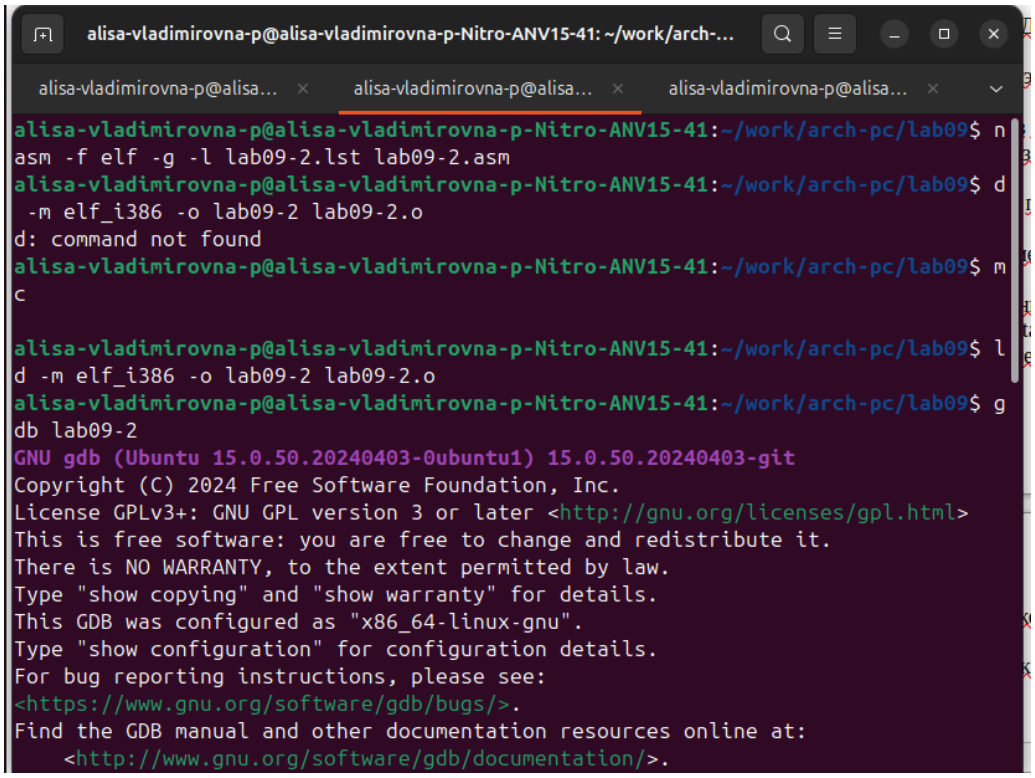
Создайте файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!)



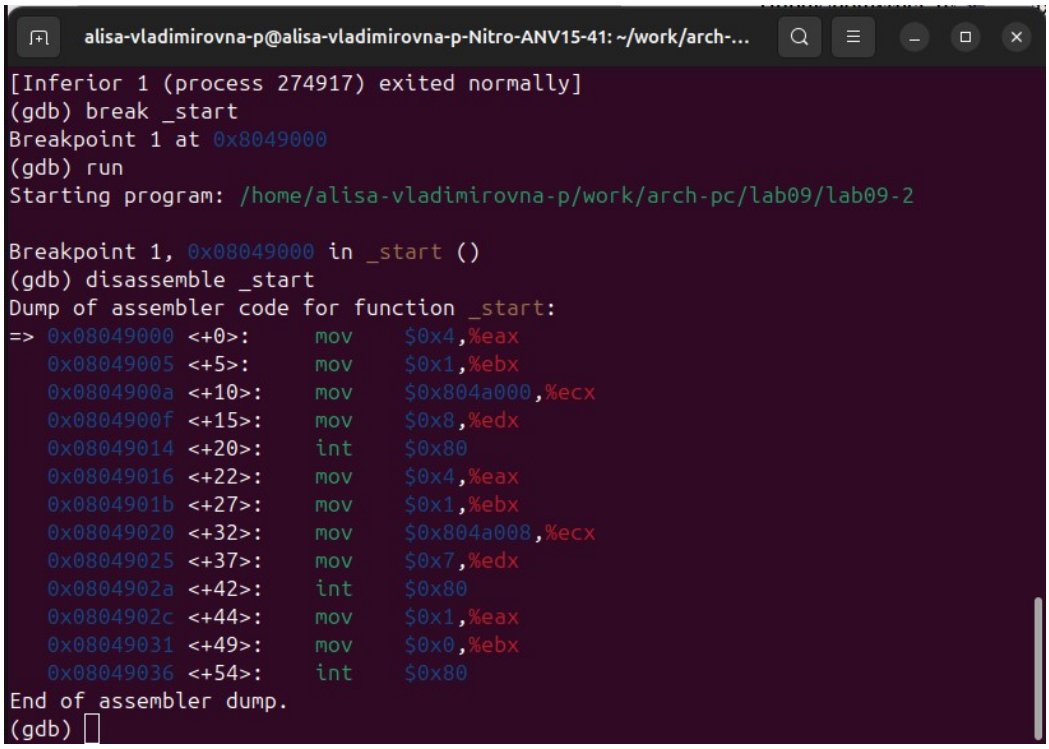
Создан исполняемый файл lab09-2.asm исполняющий программу из листинга 9.2 выводящую слова “Hello, world!”

Получите исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить
отладочную информацию, для этого трансляцию программ необходимо проводить с ключом
Загрузите исполняемый файл в отладчик gdb
Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).
Для более подробного анализа программы установите брейкпоинт на метку _start, с
которой начинается выполнение любой ассемблерной программы, и запустите её.
Посмотрите дисассимилированный код программы с помощью команды disassemble начиная с метки _start.
Переключитесь на отображение команд с Intel’овским синтаксисом, введя команду set.

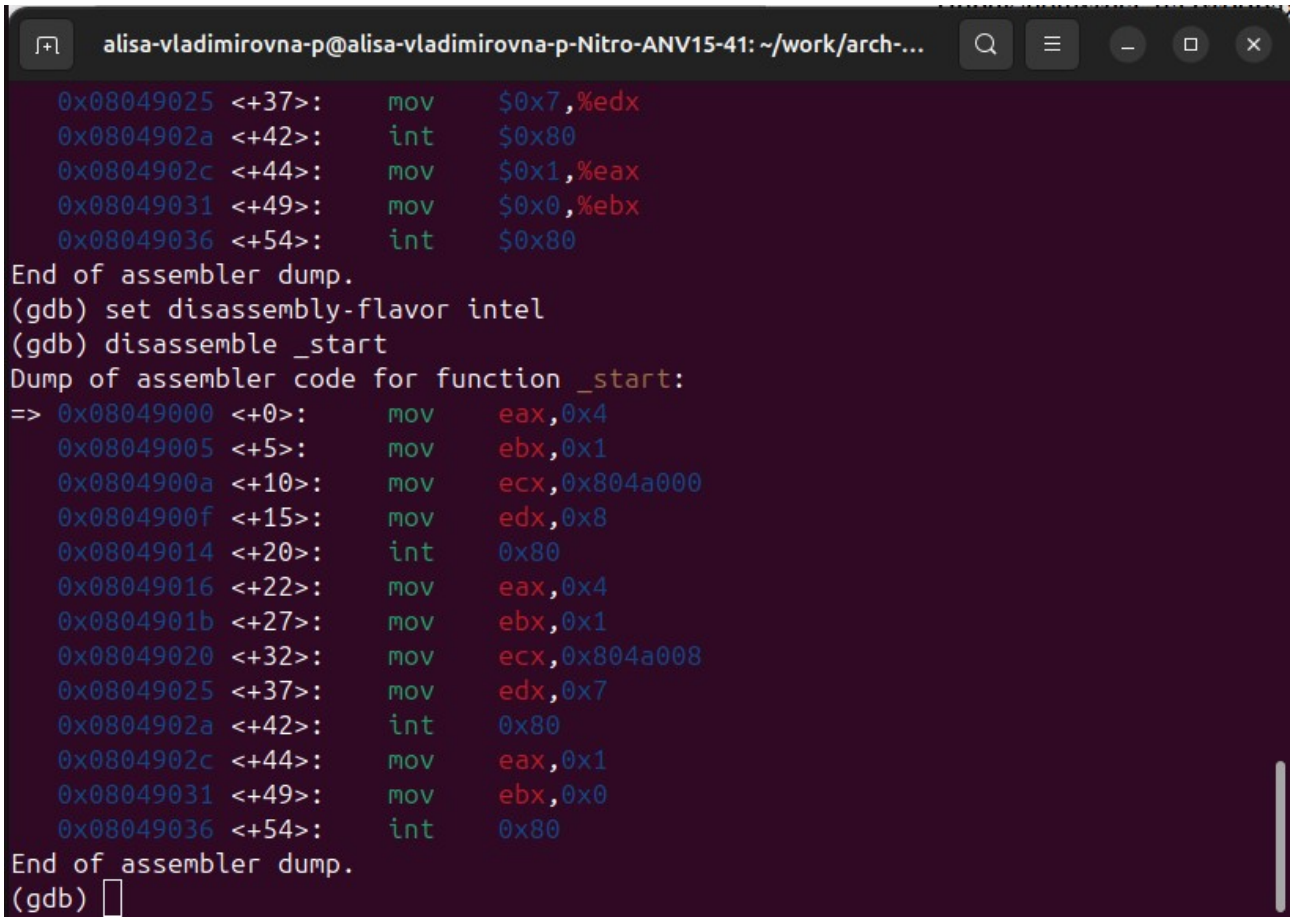
Перечислите различия отображения синтаксиса машинных команд в режимах ATT и Intel.
Включите режим псевдографики для более удобного анализа программы.



Созданы и скомпелированы файлы по указанныйм инструкциям. Файл загружен в GBD. Рассмотрен диссамилированный код программы.



В GDB команда disassemble _start показала код в синтаксисе ATT: регистры с %, значения с $, порядок mov $0x4,%eax. После set disassembly-flavor intel синтаксис сменился на Intel: регистры без префиксов, порядок mov eax,0x4. Это иллюстрирует ключевое различие: в ATT порядок «источник → приёмник», в Intel — «приёмник ← источник». Программа выводит две строки через syscall int 0x80

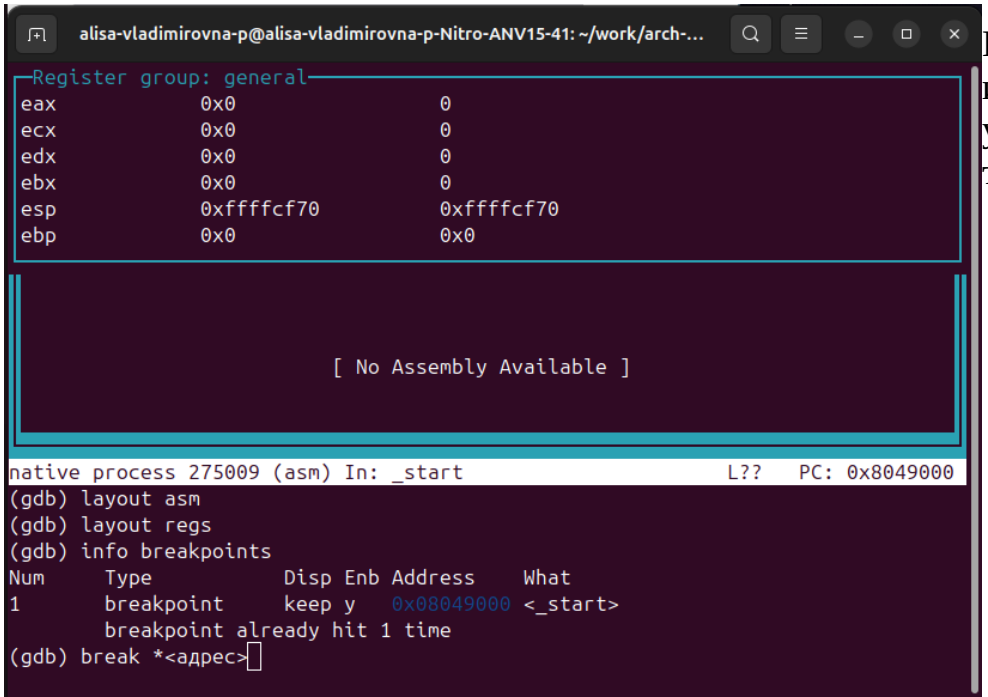


Задание 3

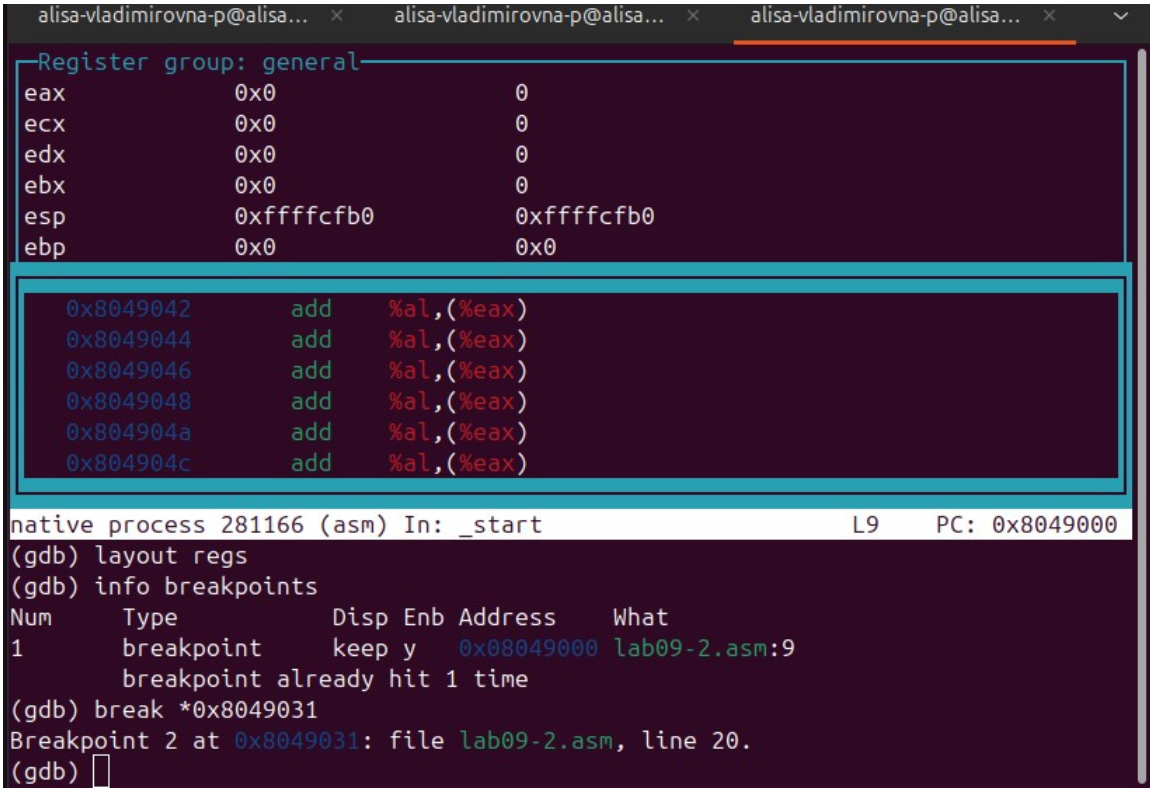
На предыдущих шагах была установлена точка останова по имени метки (_start). Проверьте это с помощью команды info breakpoints.

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 останова.



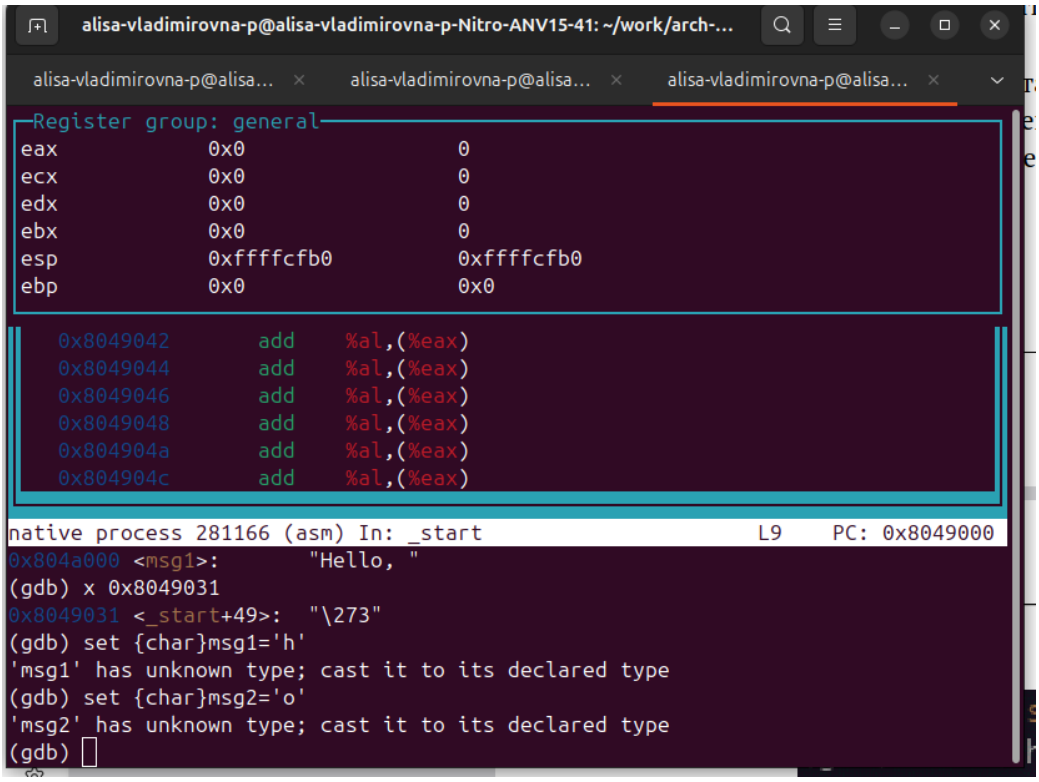
Посмотрите информацию о всех установленных точках останова.

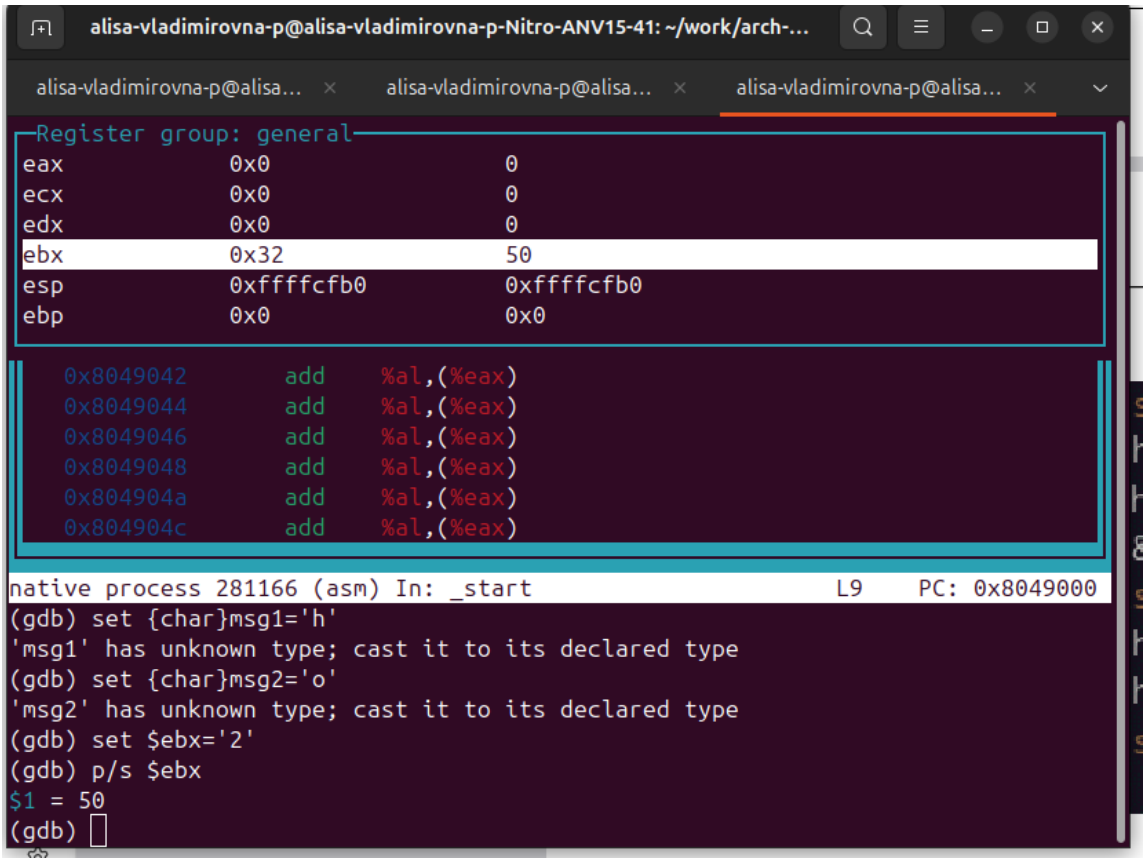


Рассмотрено расположение уже созданной точки остановы, создана вторая точка остановы.

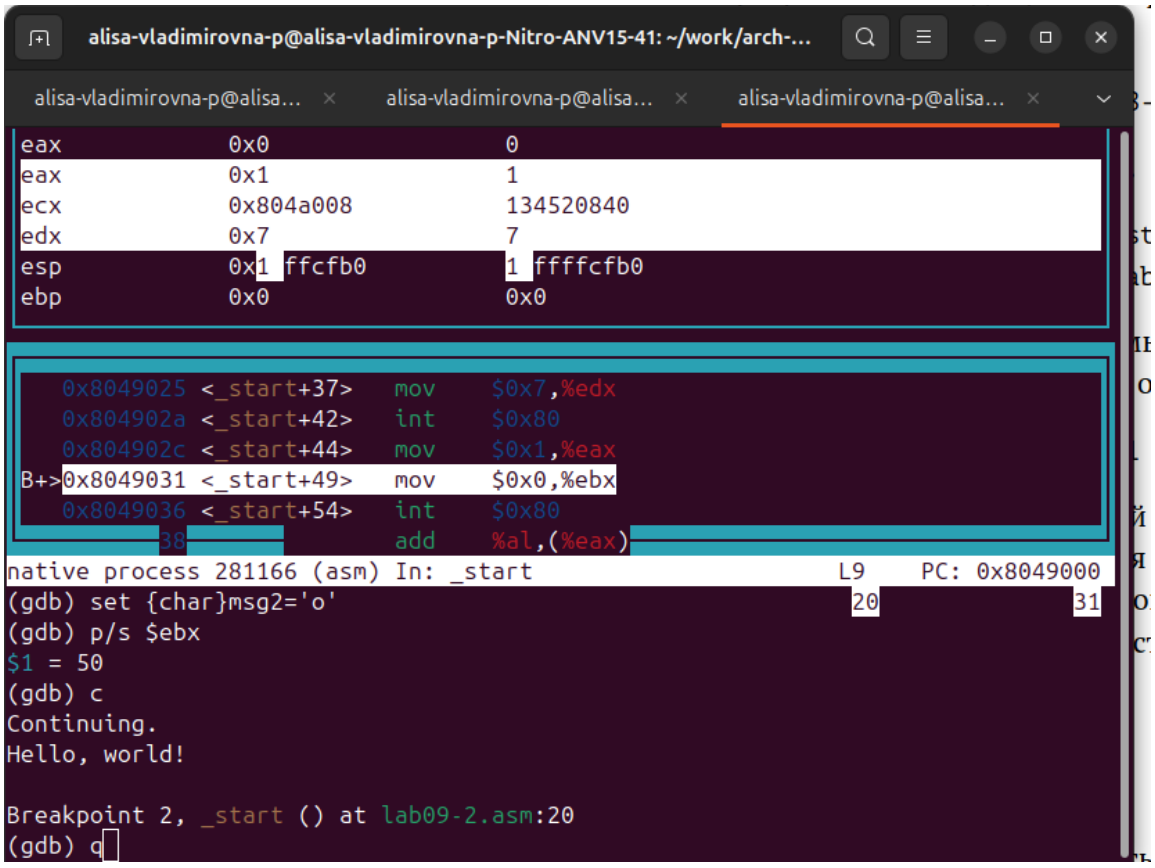
Задание 4

С помощью команды x &<имя переменной> также можно посмотреть содержимое переменной.
Посмотрите значение переменной msg1 по имени
Измените первый символ переменной msg1
Замените любой символ во второй переменной msg2





Завершите выполнение программы с помощью команды continue (сокращенно c) или
stepi (сокращенно si) и выйдите из GDB с помощью команды quit



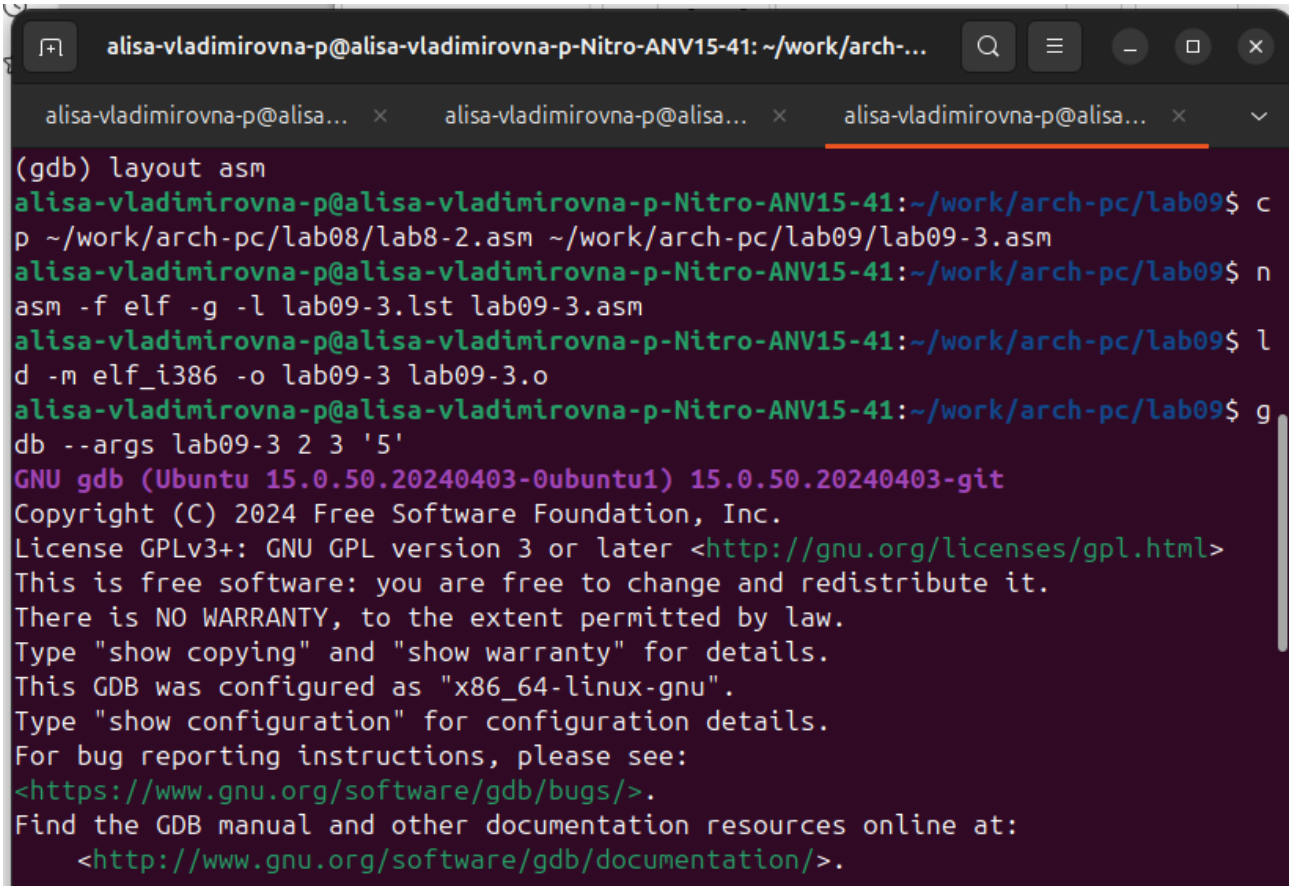
Задание 4

Скопируйте файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm

Создайте исполняемый файл

Для загрузки в gdb программы с аргументами необходимо использовать ключ --args.

Загрузите исполняемый файл в отладчик, указав аргументы



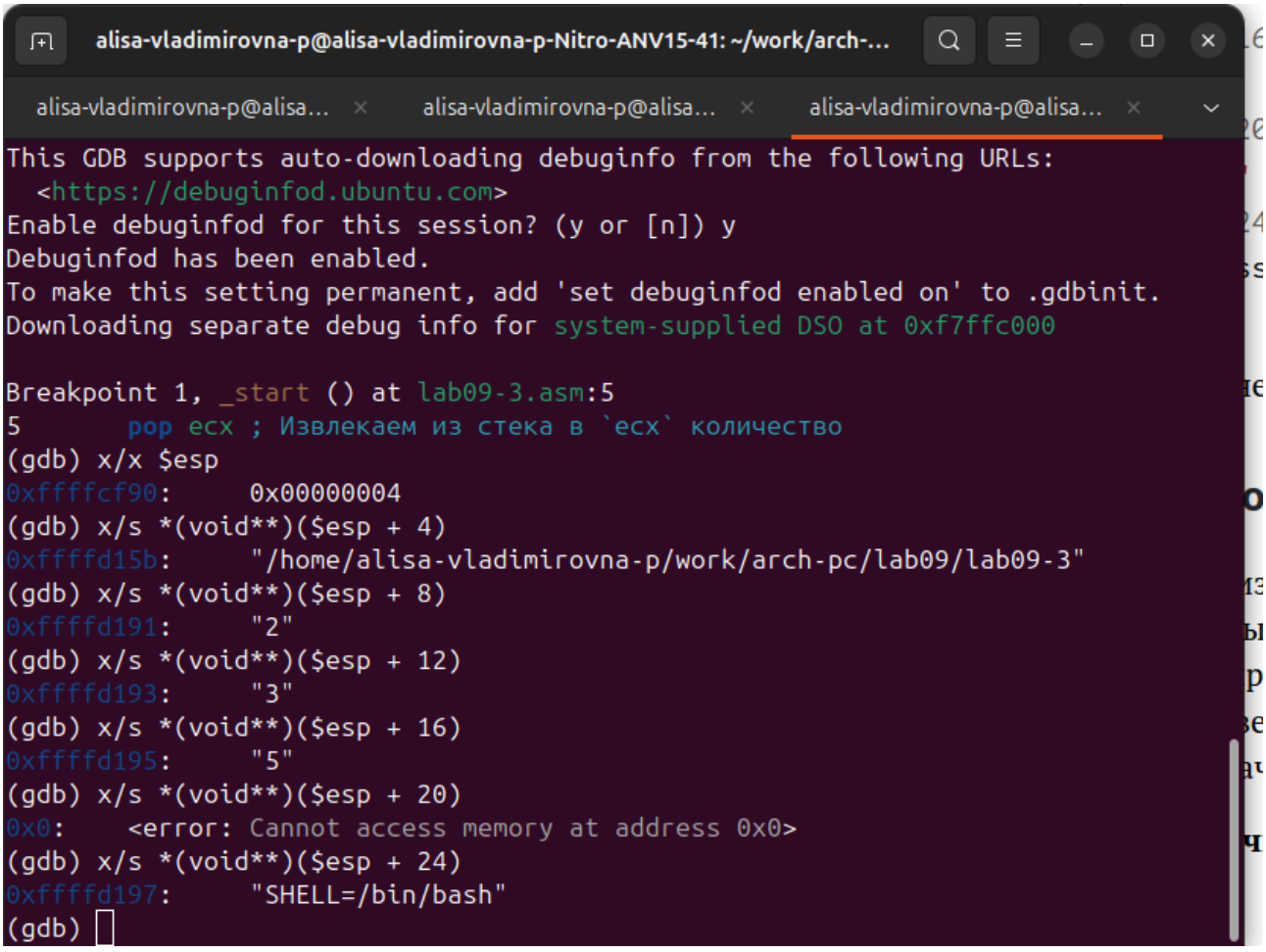
Посмотрите остальные позиции стека – по адесу [esp+4] располагается адрес в памяти

где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по

аресу [esp+12] – второго и т.д

Объясните, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и т.д.).

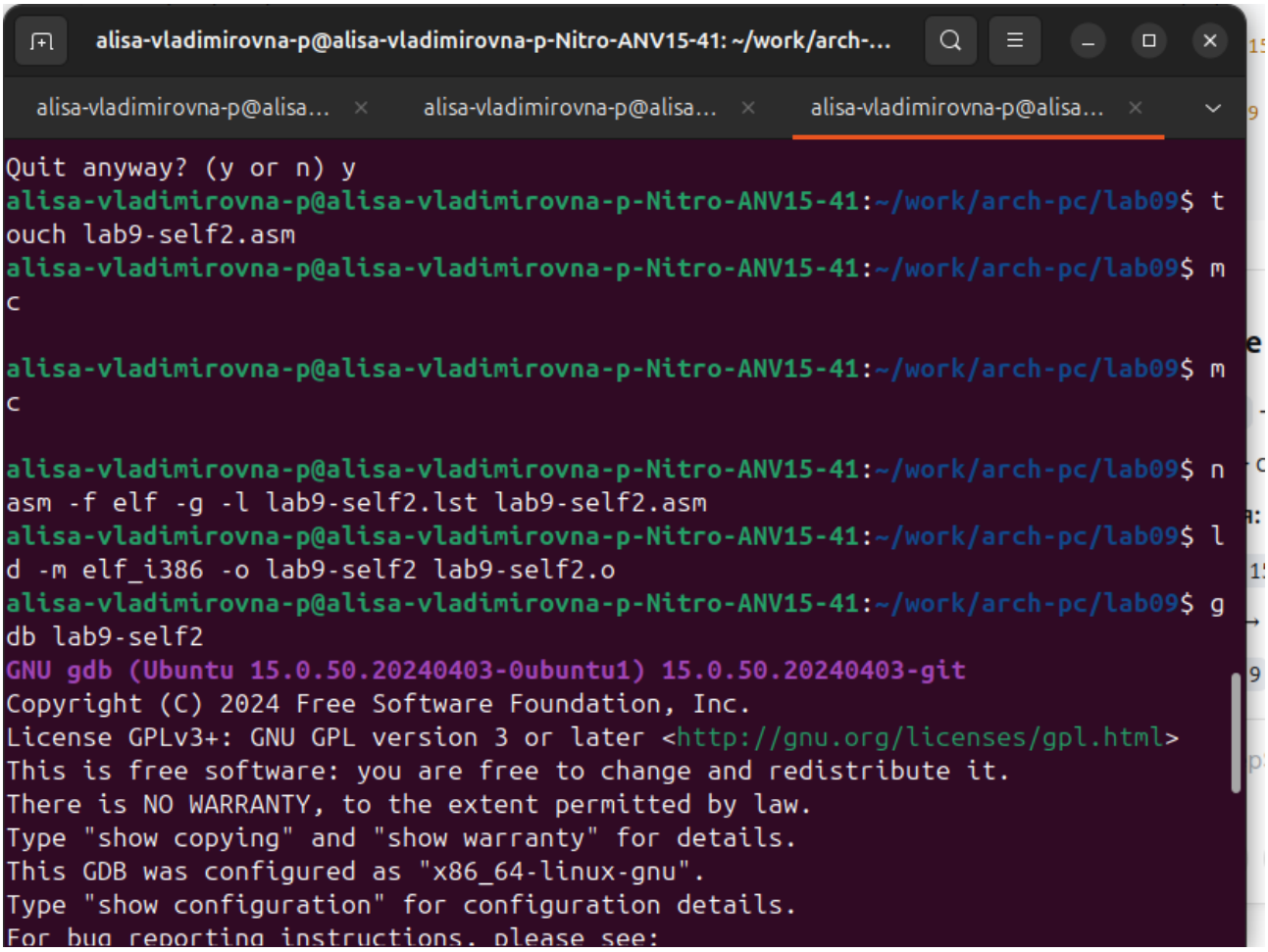
Объяснение: Шаг изменения адреса равен 4, потому что в архитектуре x86 каждый элемент в стеке занимает 4 байта (размер машинного слова). Адресная арифметика работает в байтах, поэтому для перехода к следующему элементу стека нужно прибавить 4 к текущему адресу. Например, `[esp]` указывает на первый элемент (количество аргументов), `[esp+4]` — на следующий (адрес имени программы), `[esp+8]` — на адрес первого аргумента и так далее. Это связано с тем, что в стеке хранятся 32-битные указатели, каждый из которых занимает ровно 4 байта памяти.


alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-...
alisa-vladimirovna-p@alisa...
alisa-vladimirovna-p@alisa...
alisa-vladimirovna-p@alisa...
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Breakpoint 1, _start () at lab09-3.asm:5
5 pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffcf90: 0x00000004
(gdb) x/s *(void**)($esp + 4)
0xffffd15b: "/home/alisa-vladimirovna-p/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd191: "2"
(gdb) x/s *(void**)($esp + 12)
0xffffd193: "3"
(gdb) x/s *(void**)($esp + 16)
0xffffd195: "5"
(gdb) x/s *(void**)($esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)($esp + 24)
0xffffd197: "SHELL=/bin/bash"
(gdb)

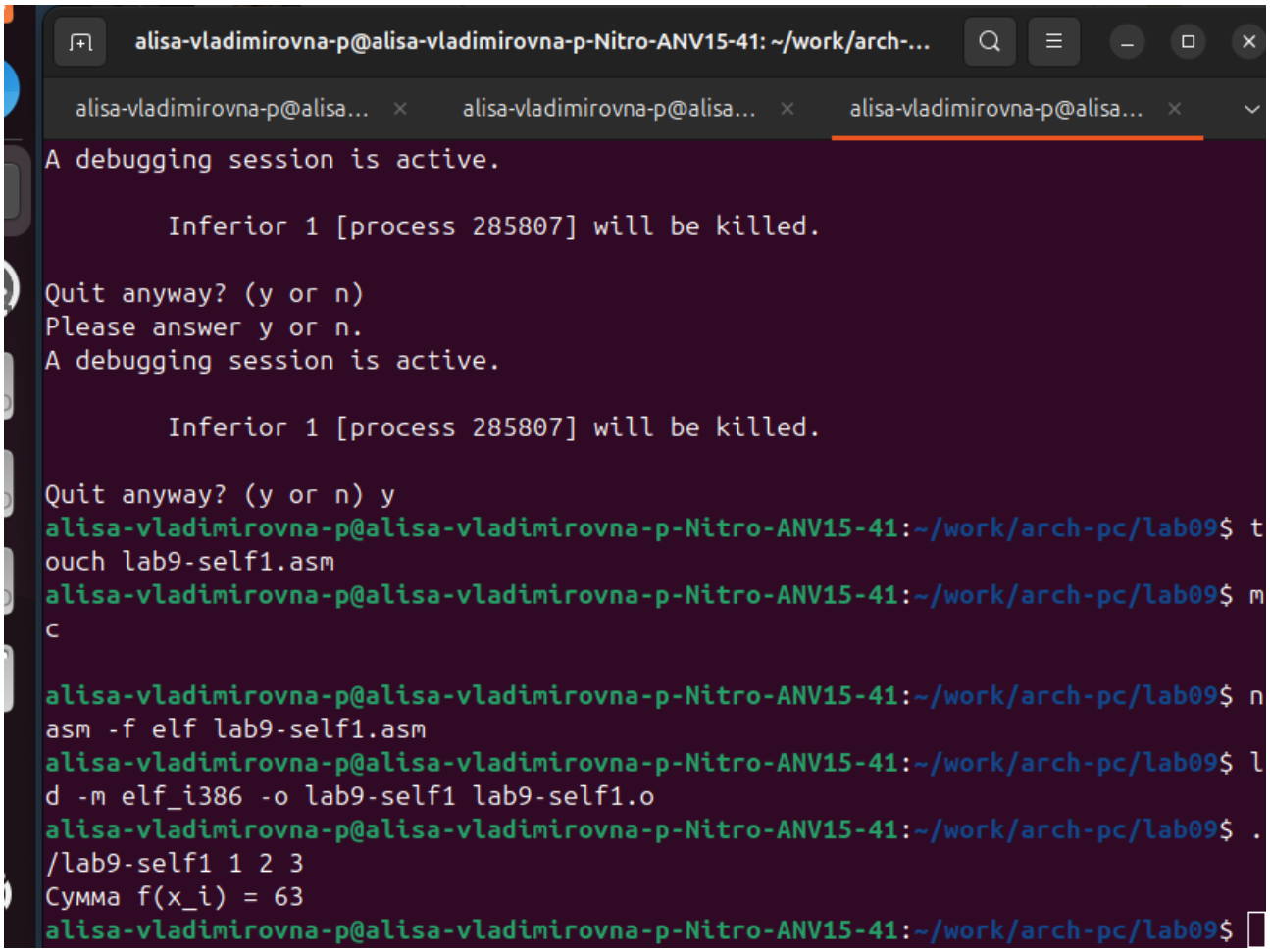
Задание для самостоятельной работы

1. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятель-
ной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.

2. В листинге 9.3 приведена программа вычисления выражения (3 + 2) * 4 + 5. При запуске
данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB,
анализируя изменения значений регистров, определите ошибку и исправьте ее.



1


alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-...
alisa-vladimirovna-p@alisa...
alisa-vladimirovna-p@alisa...
alisa-vladimirovna-p@alisa...
A debugging session is active.
Inferior 1 [process 285807] will be killed.
Quit anyway? (y or n)
Please answer y or n.
A debugging session is active.
Inferior 1 [process 285807] will be killed.
Quit anyway? (y or n) y
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ touch lab9-self1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ mc
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ nasm -f elf lab9-self1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-self1 lab9-self1.o
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ ./lab9-self1 1 2 3
Сумма f(x_i) = 63
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$

2

Вывод

В лабораторной работе №9 изучены подпрограммы в NASM (call/ret) и работа с отладчиком GDB. Созданы программы с вызовом подпрограмм, проведена отладка: установлены точки останова, проанализирован код в синтаксисах ATT/Intel, исследованы регистры, память и аргументы командной строки. Приобретены навыки отладки и структурирования ассемблерного кода.

Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL:
https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.
org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. —
354 с. — (In a Nutshell). — ISBN 0596009658. — URL:
http://www.amazon.com/Learning-
bash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL:
https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-
Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. —
М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. —
(Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
2015. — 1120 с. — (Классика Computer Science)