

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

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-... 
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ t
ouch lab09-1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n
asm -f elf lab09-1.asm
lab09-1.asm:1: error: unable to open include file `in_out.asm': No such file or
directory
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n
asm -f elf lab09-1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ l
d -m elf_i386 -o lab09-1 lab09-1.o
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ .
/lab09-1
Введите x: 10
2x+7=27
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ 
```

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

```

The screenshot shows a terminal window with the following content:

```

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-...

```

The terminal shows the following steps:

- Compilation of the assembly code:

 - Running `nasm -f elf lab09-1.asm` to generate an ELF file.
 - Linking with `ld -m elf_i386 -o lab09-1 lab09-1.o` to produce `lab09-1`.
 - Execution of the program:

Output from the program:

```

Ведите x: 10
2x+7=27

```

Recompilation and linking of a second version:

```

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-...

```

Output from the second program run:

```

Ведите x: 11
f(x)=15x-9=156

```

Задание 2

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

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-... $ m  
c  
  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n  
asm -f elf lab09-11.asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ l  
d -m elf_i386 -o lab09-11 lab09-11.o  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ .  
/lab09-11  
Введите x: 11  
f(x)=15x-9=156  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ t  
ouch lab09-2.asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m  
c  
  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n  
asm -f elf lab09-2.asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ l  
d -m elf_i386 -o lab09-2 lab09-2.o  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ .  
/lab09-2  
Hello, world!  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ █
```

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.

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-...
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
d: command not found
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-pc/lab09$ l
d -m elf_i386 -o lab09-2 lab09-2.o
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-pc/lab09$ g
db lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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/>.
```

Созданы и скомпилированы файлы по указанным инструкциям. Файл загружен в GDB. Рассмотрен дисассамилированный 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 префиксров, 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

```
[Inferior 1 (process 274917) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/alisa-vladimirovna-p/work/arch-pc/lab09/lab09-2

Breakpoint 1, 0x08049000 in _start ()
(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)
```

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... Q ⌂ ⌄ ×

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) █
```

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.

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... Q ⌂ ⌄ ×

Register group: general-
eax          0x0            0
ecx          0x0            0
edx          0x0            0
ebx          0x0            0
esp        0xfffffcf70      0xfffffcf70
ebp          0x0            0x0
[ No Assembly Available ]

native process 275009 (asm) In: _start          L??    PC: 0x8049000
(gdb) layout asm
(gdb) layout regs
(gdb) info breakpoints
Num  Type      Disp Enb Address  What
1   breakpoint keep y  0x08049000 <_start>
                                breakpoint already hit 1 time
(gdb) bbreak *<адрес>█
```

Посмотрите информацию о всех установленных точках останова.

```
alisa-vladimirovna-p@alisa... x alisa-vladimirovna-p@alisa... x alisa-vladimirovna-p@alisa... x
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcfb0 0xfffffcfb0
ebp          0x0          0x0

0x8049042    add    %al,(%eax)
0x8049044    add    %al,(%eax)
0x8049046    add    %al,(%eax)
0x8049048    add    %al,(%eax)
0x804904a    add    %al,(%eax)
0x804904c    add    %al,(%eax)

native process 281166 (asm) In: _start          L9      PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num      Type      Disp Enb Address      What
1        breakpoint      keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) 
```

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`

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-...
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcfb0 0xfffffcfb0
ebp          0x0          0x0

0x8049042    add    %al,(%eax)
0x8049044    add    %al,(%eax)
0x8049046    add    %al,(%eax)
0x8049048    add    %al,(%eax)
0x804904a    add    %al,(%eax)
0x804904c    add    %al,(%eax)

native process 281166 (asm) In: _start          L9      PC: 0x8049000
0x804a000 <msg1>:      "Hello, "
(gdb) x 0x8049031
0x8049031 <_start+49>: "\273"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}msg2='o'
'msg2' has unknown type; cast it to its declared type
(gdb) 
```

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-... Q - x
alisa-vladimirovna-p@alisa... alisa-vladimirovna-p@alisa... alisa-vladimirovna-p@alisa...
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x32         50
esp          0xfffffcfb0  0xfffffcfb0
ebp          0x0          0x0

0x8049042      add    %al,(%eax)
0x8049044      add    %al,(%eax)
0x8049046      add    %al,(%eax)
0x8049048      add    %al,(%eax)
0x804904a      add    %al,(%eax)
0x804904c      add    %al,(%eax)

native process 281166 (asm) In: _start          L9   PC: 0x8049000
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}msg2='o'
'msg2' has unknown type; cast it to its declared type
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb)
```

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

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-... Q - x
alisa-vladimirovna-p@alisa... alisa-vladimirovna-p@alisa... alisa-vladimirovna-p@alisa...
Register group: general
eax          0x0          0
eax          0x1          1
ecx          0x804a008     134520840
edx          0x7          7
esp          0x1_ffcfb0    1_ffcfb0
ebp          0x0          0x0

0x8049025 <_start+37>  mov    $0x7,%edx
0x804902a <_start+42>  int    $0x80
0x804902c <_start+44>  mov    $0x1,%eax
B+>0x8049031 <_start+49> mov    $0x0,%ebx
0x8049036 <_start+54>  int    $0x80
38          add    %al,(%eax)

native process 281166 (asm) In: _start          L9   PC: 0x8049000
(gdb) set {char}msg2='o'                         20   31
(gdb) p/s $ebx
$1 = 50
(gdb) c
Continuing.
Hello, world!

Breakpoint 2, _start () at lab09-2.asm:20
(gdb) q
```

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

The screenshot shows a terminal window with three tabs. The active tab displays the following command sequence:

```
(gdb) layout asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-...$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ nm -f elf -g -l lab09-3.lst lab09-3.asm  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o  
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '5'  
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git  
Copyright (C) 2024 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-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/>.
```

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

где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по

адресу [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.

```
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  
0xfffffcf90: 0x00000004  
(gdb) x/s *(void**)( $esp + 4)  
0xfffffd15b: "/home/alisa-vladimirovna-p/work/arch-pc/lab09/lab09-3"  
(gdb) x/s *(void**)( $esp + 8)  
0xfffffd191: "2"  
(gdb) x/s *(void**)( $esp + 12)  
0xfffffd193: "3"  
(gdb) x/s *(void**)( $esp + 16)  
0xfffffd195: "5"  
(gdb) x/s *(void**)( $esp + 20)  
0x0: <error: Cannot access memory at address 0x0>  
(gdb) x/s *(void**)( $esp + 24)  
0xfffffd197: "SHELL=/bin/bash"  
(gdb) █
```

Задание для самостоятельной работы

1. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной 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.

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... 
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... 
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... 

Quit anyway? (y or n) y
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ t
ouch lab9-self2.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n
asm -f elf -g -l lab9-self2.lst lab9-self2.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ l
d -m elf_i386 -o lab9-self2 lab9-self2.o
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ g
db lab9-self2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
```

```
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41: ~/work/arch-... Q - x
alisa-vladimirovna-p@alisa... x alisa-vladimirovna-p@alisa... x alisa-vladimirovna-p@alisa... x v

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$ t
ouch lab9-self1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ m
c

alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ n
asm -f elf lab9-self1.asm
alisa-vladimirovna-p@alisa-vladimirovna-p-Nitro-ANV15-41:~/work/arch-pc/lab09$ l
d -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 AT&T/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)