

№9

:

Пустобаев Леонид Сергеевич

3

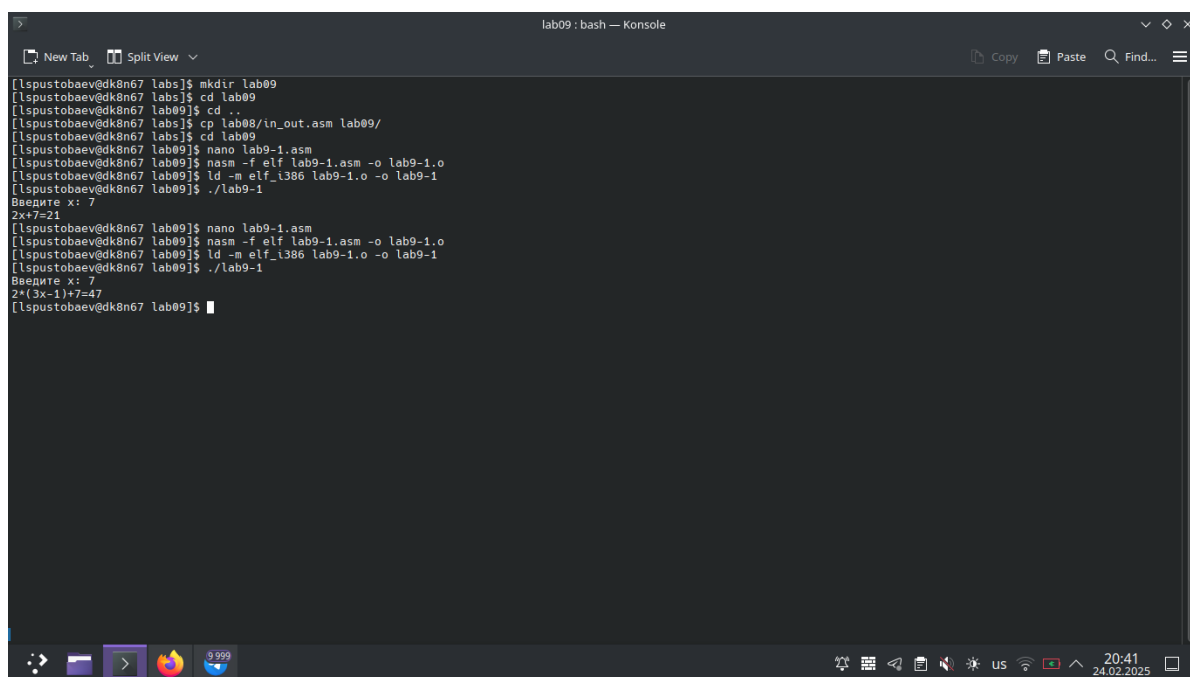
4

13

15

Номер лабораторной работы: 9 ФИО студента: Пустобаев Леонид Сергеевич Группа:
НММбд03-2024

Целью данной лабораторной работы является Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.



```
lab09 : bash — Konsole
[lsputobaev@dk8n67 labs]$ mkdir lab09
[lsputobaev@dk8n67 labs]$ cd lab09
[lsputobaev@dk8n67 lab09]$ cd ..
[lsputobaev@dk8n67 labs]$ cp lab08/in_out.asm lab09/
[lsputobaev@dk8n67 labs]$ cd lab09
[lsputobaev@dk8n67 lab09]$ nano lab9-1.asm
[lsputobaev@dk8n67 lab09]$ nasm -f elf lab9-1.asm -o lab9-1.o
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[lsputobaev@dk8n67 lab09]$ ./lab9-1
Введите x: 7
2x+7=21
[lsputobaev@dk8n67 lab09]$ nano lab9-1.asm
[lsputobaev@dk8n67 lab09]$ nasm -f elf lab9-1.asm -o lab9-1.o
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[lsputobaev@dk8n67 lab09]$ ./lab9-1
Введите x: 7
2*(3x-1)+7=47
[lsputobaev@dk8n67 lab09]$
```

На первом скриншоте я демонстрирую создание программы и её проверку.

```
lab09 : bash — Konsole
New Tab Split View
Copy Paste Find...

[lsputobaev@dk8n67 lab09]$ nano lab9-2.asm
[lsputobaev@dk8n67 lab09]$ nasm -f elf lab9-2.asm -o lab9-2.o
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 lab9-2.o -o lab9-2
[lsputobaev@dk8n67 lab09]$ ./lab9-2
Hello, world!
[lsputobaev@dk8n67 lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[lsputobaev@dk8n67 lab09]$ cd -m elf_i386 -o lab9-2 lab9-2.o
bash: cd: -m: invalid option
cd: usage: cd [-L][-P [-e]] [-@]] [dir]
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[lsputobaev@dk8n67 lab09]$ gdb lab9-2
bash: gdb: command not found
[lsputobaev@dk8n67 lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[lsputobaev@dk8n67 lab09]$ gdb lab9-2
bash: gdb: command not found
[lsputobaev@dk8n67 lab09]$ cat lab9-2.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
[lsputobaev@dk8n67 lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[lsputobaev@dk8n67 lab09]$ gdb lab9-2
bash: gdb: command not found
[lsputobaev@dk8n67 lab09]$
```

На втором скриншоте я демонстрирую создание второй программы, её проверку, а также осазнаие того, что у меня нет gdb и его придётся скачать.

```
lab09 : gdb — Konsole
New Tab Split View
Copy Paste Find...

lab09: gdb x lab09: bash x
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/lsputobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.archlinux.org>
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.
Hello, world!
[Inferior 1 (process 13932) exited normally]
(gdb) break _start
Breakpoint 1 at 0x00400000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /home/lsputobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov     eax, 4
(gdb) disassemble _start
Undefined command: "disassemble". Try "help".
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00400000 <+0>: mov     $0x4,%eax
0x00400005 <+5>: mov     $0x1,%ebx
0x0040000a <+10>: mov     $0x004a000,%ecx
0x0040000f <+15>: mov     $0x8,%edx
0x00400014 <+20>: int     $0x80
0x00400016 <+22>: mov     $0x4,%eax
0x0040001b <+27>: mov     $0x1,%ebx
0x00400020 <+32>: mov     $0x004a000,%ecx
0x00400025 <+37>: mov     $0x7,%edx
0x0040002a <+42>: int     $0x80
0x0040002c <+44>: mov     $0x1,%eax
0x00400031 <+49>: mov     $0x0,%ebx
0x00400036 <+54>: int     $0x80
End of assembler dump.
(gdb)
```

На третьем скриншоте я демонстрирую заработваший gdb, я показываю как я устанавливаю breakpoint и запускаю отладчик. Также я показываю как не с первой попытки написал disassemble и работу этой инструкции

```
lab09: gdb — Konsole
New Tab Split View
lab09: gdb x lab09: bash x
(gdb) r
Starting program: /home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-2
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.
Hello, world!
[Inferior 1 (process 13932) exited normally]
(gdb) break _start
Breakpoint 1 at 0x00490000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-2
Breakpoint 1, _start () at lab9-2.asm:9
9   mov     eax, 4
(gdb) disassemble _start
Undefined command: "disassemble". Try "help".
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00490000 <+8>: mov     $0x4,%eax
0x0049000a <+10>: mov     $0x004a000,%ecx
0x0049000f <+15>: mov     $0x8,%edx
0x00490014 <+20>: lint    $0x80
0x00490016 <+22>: mov     $0x4,%eax
0x0049001b <+27>: mov     $0x1,%ebx
0x00490020 <+32>: mov     $0x004a008,%ecx
0x00490025 <+37>: mov     $0x7,%edx
0x0049002a <+42>: lint    $0x80
0x0049002c <+44>: mov     $0x1,%eax
0x00490031 <+49>: mov     $0x0,%ebx
0x00490036 <+54>: lint    $0x80
End of assembler dump.
(gdb) set disassemble-flavor intel
No symbol "disassemble" in current context.
(gdb) set disassemble-flavor intel
(gdb) disassemble _start
No symbol "_start" in current context.
(gdb) |
```

На четвёртом скриншоте я демонстрирую ошибку, которая у меня возникла при выполнении лабораторной работы, мне не удалось её исправить.

```
lab09: gdb — Konsole
New Tab Split View
lab09: gdb x lab09: bash x
B=> 0x00490000 <_start>: mov     eax,0x4
0x00490005 <+_start+5>: mov     ebx,0x1
0x00490008 <+_start+8>: mov     ecx,0x004a000
0x0049000f <+_start+15>: mov     edx,0x8
0x00490014 <+_start+20>: lint    0x80
0x00490016 <+_start+22>: mov     eax,0x4
0x0049001b <+_start+27>: mov     ebx,0x1
0x00490020 <+_start+32>: mov     ecx,0x004a008
0x00490025 <+_start+37>: mov     edx,0x7
0x0049002a <+_start+42>: lint    0x80
0x0049002c <+_start+44>: mov     eax,0x1
0x00490031 <+_start+49>: mov     ebx,0x0
0x00490036 <+_start+54>: lint    0x80
0x00490038      add     BYTE PTR [eax],al
0x0049003a      add     BYTE PTR [eax],al
0x0049003c      add     BYTE PTR [eax],al
0x0049003e      add     BYTE PTR [eax],al
0x00490040      add     BYTE PTR [eax],al
0x00490042      add     BYTE PTR [eax],al
0x00490044      add     BYTE PTR [eax],al
0x00490046      add     BYTE PTR [eax],al
0x00490048      add     BYTE PTR [eax],al
0x0049004a      add     BYTE PTR [eax],al
0x0049004c      add     BYTE PTR [eax],al
native process 13972 (asm) In: _start
(gdb) |
```

На пятом скриншоте я демонстрирую как меняется экран при вводе инструкции layout asm

The screenshot shows a GDB console window titled 'lab09: gdb — Konsole'. The top panel displays the 'Register window: general' with the following values:

Register	Value	Register	Value	Register	Value	Register	Value
eax	0x0	ecx	0x0	edx	0x0		
ebx	0x0	esp	0xffffd3b0	ebp	0x0		
esi	0x0	edi	0x0	eip	0x8049000		
eflags	0x202	cs	0x23	ss	0x2b		
ds	0x2b	es	0x2b	fs	0x0		
gs	0x0						

The bottom panel shows the assembly code for the function '_start' at address 0x8049000:

```
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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0
```

The status bar at the bottom indicates 'native process 13972 (asm) In: _start', 'L9', and 'PC: 0x8049000'.

На шестом скриншоте я демонстрирую финальную версию окна gdb ещё и с окном регистров

The screenshot shows a GDB console window titled 'lab09: gdb — Konsole'. The top panel displays the 'Register window: general' with the same values as the previous screenshot.

The bottom panel shows the assembly code for the function '_start' at address 0x8049000, which is the same as the previous screenshot.

The status bar at the bottom indicates 'native process 13972 (asm) In: _start', 'L9', and 'PC: 0x8049000'.

The bottom panel also shows the 'breakpoints' window with the following information:

Num	Type	Disp	Enb	Address	What
1	breakpoint	keep	y	0x8049000	lab0-2.asm:9

The status bar at the bottom indicates 'breakpoint already hit 1 time'.

На седьмом скриншоте я демонстрирую инструкцию info breakpoints

```
lab09: gdb — Konsole
lab09: gdb x lab09: bash x
--Register group: general--
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xffffd3b0 0 ebp 0x0 0
esi 0x0 0 edi 0x0 0 eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ] cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0

0+ 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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0+ 0x8049031 <_start+49> mov ebx,0x0

native process 13972 (asm) In: _start
L9 PC: 0x8049000
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb) break _start+49
Function "_start+49" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x8049031 lab9-2.asm:20
(gdb) |
```

На восьмом скриншоте я демонстрирую инструкцию i b

```
lab09: gdb — Konsole
lab09: gdb x lab09: bash x
--Register group: general--
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xffffd3b0 0 ebp 0x0 0
esi 0x0 0 edi 0x0 0 eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ] cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0

0+ 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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0+ 0x8049031 <_start+49> mov ebx,0x0

native process 13972 (asm) In: _start
L9 PC: 0x8049000
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffd3b0 0xffffd3b0
ebp 0x0 0
esi 0x0 0
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
fs 0x0 0
gs 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--
```

На девятом скриншоте я демонстрирую исполнение работы следующей инструкции


```
lab09: gdb — Konsole
lab09: gdb x lab09: bash x
--Register group: general--
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xffffd3b0 0 ebp 0x0 0
esi 0x0 0 edi 0x0 0 eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ] cs 0x23 35 ss 0x2b 43
ds 0x0 0 es 0x2b 43 fs 0x0 0
gs 0x0 0

0+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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0+0x8049031 <_start+49> mov ebx,0x0

native process 13972 (asm) In: _start
esp 0xffffd3b0 0xffffd3b0
ebp 0x0 0
esi 0x0 0
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x0 0
es 0x2b 43
fs 0x0 0
gs 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/15b &msg1
0x804a000 <msg1>: "Hello, "
(gdb)
```

На десятом скриншоте я демонстрирую содержимое msg1

```
lab09: gdb — Konsole
lab09: gdb x lab09: bash x
--Register group: general--
eax 0x0 0 ecx 0x0 0 edx 0x0 0
ebx 0x0 0 esp 0xffffd3b0 0xffffd3b0
esi 0x0 0 edi 0x0 0 eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ] cs 0x23 35 ss 0x2b 43
ds 0x0 0 es 0x2b 43 fs 0x0 0
gs 0x0 0

0+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
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0+0x8049031 <_start+49> mov ebx,0x0

native process 13972 (asm) In: _start
'msg1' has unknown type; cast it to its declared type
(gdb) x/15b &msg1
0x804a000 <msg1>: "Hello, "
(gdb) set {char}msg2='T'
'msg2' has unknown type; cast it to its declared type
(gdb) x/15b &msg2
0x804a000 <msg2>: "world!\n\034"
(gdb) p/f $ebx
No symbol "F" in current context.
(gdb) p/f $edx
No symbol "F" in current context.
(gdb) print /F $edx
No symbol "F" in current context.
(gdb)
```

На одиннадцатом скриншоте я демонстрирую содержимое msg2 и невозможность исполнение инструкции p/F, ошибка не была устранена


```
lab09: gdb x lab09: bash x
eax 0x0 0 ecx 0x0 0 edx 0x0 0
eax 0x1 1 ecx 0x804a008 134520840 edx 0x7 7
esi 0x1 1 edi 0x0 0 eip 0x8049000 0x8049000 <_start>
eflags 0x202 [ IF ] cs 0x23 35 eip 0x8049031 0x8049031 <_start+4>
ds 0x2b 43 es 0x2b 43 fs 0x0 0
gs 0x0 0

0x8049000 <_start> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov ed,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0
0x8049031 <_start+49> mov ebx,0x0
0x804903b <_start+54> int 0x80
0x804903b add BYTE PTR [eax],1
0x804903c add BYTE PTR [eax],1
0x804903c add BYTE PTR [eax],1
0x804903e add BYTE PTR [eax],1

Native process 13972 (asm) in: _start
No symbol "f" in current context.
(gdb) set $ebx=2
(gdb) p/s $ebx
$1 = 50
(gdb) $3 = 50
Undefined command: "$3". Try "help".
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 2
(gdb) cHello, world!
Continuing.

Breakpoint 2, _start () at lab9-2.asm:20
(gdb)
```

На четырнадцатом скриншоте я демонстрирую продолжение работы программы после инструкции continue

```
lab09: gdb x lab09: bash x
(gdb) disassemble -start
No symbol "start" in current context.
(gdb) layout asm
[lsputobaev@dk8n67 lab09]$ cd ..
[lsputobaev@dk8n67 labs]$ cp lab08/lab9-2.asm lab09/lab9-3.asm
[lsputobaev@dk8n67 labs]$ cd lab09
[lsputobaev@dk8n67 lab09]$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[lsputobaev@dk8n67 lab09]$ gdb --args lab9-3 1 2 3
GNU gdb (GDB) 16.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/lsputobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-3 1 2 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.archlinux.org>
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.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в ecx количество
(gdb) x/x $esp
0xffffd390: 0x00000004
(gdb)
```

На пятнадцатом скриншоте я демонстрирую второй запуск gdb, на этот раз с другой программой. Исследую стек программы, обнаруживаю, что в стеке всего 4 элемента, а не 5

```
lab09: gdb — Konsole
New Tab Split View
lab09: gdb lab09: bash
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-pc-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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-3 1 2 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.archlinux.org>
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.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop     ecx ; Извлекаем из стека в ecx количество
(gdb) x/x $esp
0xffffd398: 0x00000004
(gdb) x/s *(void**)(($esp +4)
0xffffd5bd: "/home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(($esp +8)
0xffffd623: "1"
(gdb) x/s *(void**)(($esp +12)
0xffffd625: "2"
(gdb) x/s *(void**)(($esp +20)
0x0: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(($esp + 24)
0xffffd629: "SHELL=/bin/bash"
(gdb) |
```

На шестнадцатом скриншоте я демонстрирую содержимое каждой из переменных записанных в стеке. Подмечаю, что шаг равен 4 битам, именно такой размер у ячейки стека. По не понятной мне причине в стеке всего два элеиента, а не три

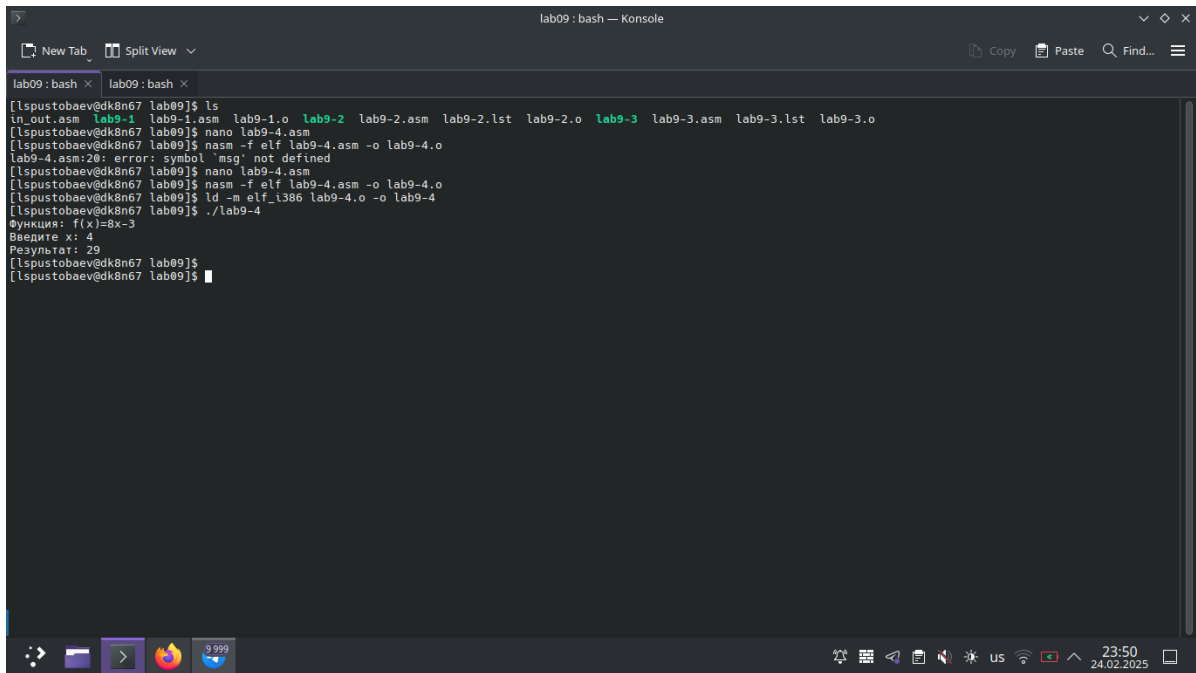
```
lab09: gdb — Konsole
lab09: gdb x lab09: bash x
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-3 1 2 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.archlinux.org>
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.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в ехх количество
(gdb) x/x $esp
0xffffd39d: 0x00000004
(gdb) x/s *(void**)(esp+4)
0xffffd5bd: "/home/lspustobaev/work/study/2024-2025/Computer Architecture/study_2024-2025_arh-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(esp+8)
0xffffd623: "1"
(gdb) x/s *(void**)(esp+12)
0xffffd625: "2"
(gdb) x/s *(void**)(esp+20)
0x0: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(esp+24)
0xffffd629: "SHELL=/bin/bash"
(gdb)
```

На семнадцатом скриншоте я продемонстрировал выполнение самостоятельной работы, создание файла с кодом, создание исполняемого файла и его проверку, содержимое всех файлов можно посмотреть в git



```
lab09 : bash — Konsole
lab09 : bash x lab09 : bash x
[lsputobaev@dk8n67 lab09]$ ls
in_out.asm lab9-1 lab9-1.o lab9-2 lab9-2.asm lab9-2.lst lab9-2.o lab9-3 lab9-3.asm lab9-3.lst lab9-3.o
[lsputobaev@dk8n67 lab09]$ nano lab9-4.asm
[lsputobaev@dk8n67 lab09]$ nasm -f elf lab9-4.asm -o lab9-4.o
lab9-4.asm:20: error: symbol 'msg' not defined
[lsputobaev@dk8n67 lab09]$ nano lab9-4.asm
[lsputobaev@dk8n67 lab09]$ nasm -f elf lab9-4.asm -o lab9-4.o
[lsputobaev@dk8n67 lab09]$ ld -m elf_i386 lab9-4.o -o lab9-4
[lsputobaev@dk8n67 lab09]$ ./lab9-4
Функция: f(x)=8x-3
Введите x: 4
Результат: 29
[lsputobaev@dk8n67 lab09]$
[lsputobaev@dk8n67 lab09]$
```

На втором скриншоте я продемонстрировал выполнение самостоятельной работы, создание файла с кодом, создание исполняемого файла и его проверку, содержимое всех файлов можно посмотреть в git

Цель лабораторной работы достигнута. Я приобрел навыки написания программ с использованием подпрограмм, а также познакомился с методами отладки при помощи GDB и его основными возможностями.