

Отчет по лабораторной работе №9

Архитектура компьютеров и операционные системы

Никита Сергеевич Кокшаров

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Реализация подпрограмм в NASM	5
2.2	Отладка программ с помощью GDB	8
2.2.1	Добавление точек останова	10
2.2.2	Работа с данными программы в GDB	11
2.2.3	Обработка аргументов командной строки в GDB	15
2.3	Задание для самостоятельной работы	16
3	Выводы	21

Список иллюстраций

2.1	Создание lab09-1.asm	5
2.2	Код в lab09-1.asm	6
2.3	Запуск lab09-1	6
2.4	Изменение кода в lab09-1.asm	7
2.5	Запуск измененного lab09-1	7
2.6	Код в lab09-2.asm	8
2.7	Запуск lab09-2 с помощью отладчика	8
2.8	Запуск lab09-2 с брейкпоинтом	9
2.9	Дизассемблированный код от отметки _start	9
2.10	Дизассемблированный код от отметки _start (Intel-синтаксис)	9
2.11	Режим псевдографики	10
2.12	Точки останова при отладке lab09-2	10
2.13	Установка второго брейкпоинта	11
2.14	Изменение регистра eax	11
2.15	Изменение регистра ebx	12
2.16	Изменение регистра ecx	12
2.17	Изменение регистра edx	13
2.18	Изменение регистра eax	13
2.19	Содержимое регистров	14
2.20	Содержимое msg1, msg2	14
2.21	Изменение msg1	14
2.22	Изменение msg2	14
2.23	Значения регистра edx	15
2.24	Изменение значения регистра ebx	15
2.25	Создание lab09-3	15
2.26	Отладка lab09-3	16
2.27	Обращение к позициям стека	16
2.28	Код в lab09-4.asm	17
2.29	Запуск lab09-4	17
2.30	Код в lab09-5.asm	18
2.31	Запуск lab09-5	18
2.32	Отладка lab09-5	19
2.33	Исправленный код в lab09-5.asm	19
2.34	Запуск исправленного lab09-5	20

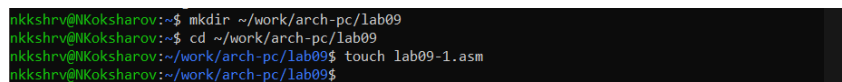
1 Цель работы

Целью работы является приобретение навыков написания программ с использование подпрограмм, ознакомление с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

Создаю файл lab09-1.asm в новой директории lab09 (рис. 2.1).



```
nkkshrv@Nkoksharov:~$ mkdir ~/work/arch-pc/lab09
nkkshrv@Nkoksharov:~$ cd ~/work/arch-pc/lab09
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ touch lab09-1.asm
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание lab09-1.asm

Пишу код программы из листинга 9.1 (рис. 2.2)

```

mc [nkshrv@NKoksharov]:~/work/arch-pc/lab09
/home/nkshrv/work~lab09/lab09-1.asm [-M--] 7 L: [ 1+39 40/ 40] *(431 / 431b) <EOF> [*][X] ^
%include "in_out.asm"

SECTION .data
    msg: DB "Введите x: ", 0
    result: DB "2x+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 ecx, [res]
    call iprintf

    call quit

_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax

    ret

```

Рис. 2.2: Код в lab09-1.asm

Создаю исполняемый файл и запускаю его (рис. 2.3)

```

nkshrv@NKoksharov:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
nkshrv@NKoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
nkshrv@NKoksharov:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 10
2x+7=27
nkshrv@NKoksharov:~/work/arch-pc/lab09$

```

Рис. 2.3: Запуск lab09-1

Ответ верный.

Изменяю код программы, добавляя подпрограмму `_subcalcul`, вычисляя значение $f(g(x))$, где $f(x) = 2x + 7$, $g(x) = 3x - 1$ (рис. 2.4)



```
mc [nkshrv@NKoksharov:~/work/arch-pc/lab09
/home/nkshrv/work~lab09/lab09-1.asm  [----] 26 L: [ 1+ 4 5/ 49] *(99 / 521b) 0039 0x027 [*][X] ^
#include "in_out.asm"

SECTION .data
    msg: DB "Введите x: ", 0
    result: DB "f(g(x)) = ", 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 ecx, [res]
    call iprintf

    call quit

_calcul:
    call _subcalcul

    mov ebx, 2
    mul ebx
    add ecx, 7
    mov [res], ecx

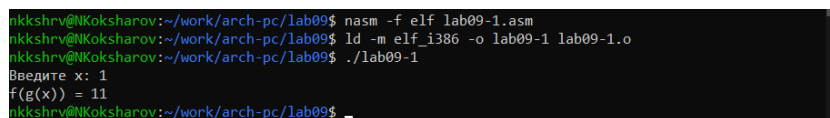
    ret

_subcalcul:
    mov ebx, 3
    mul ebx
    sub ecx, 1

    ret
```

Рис. 2.4: Изменение кода в lab09-1.asm

Создаю исполняемый файл и запускаю его (рис. 2.5)



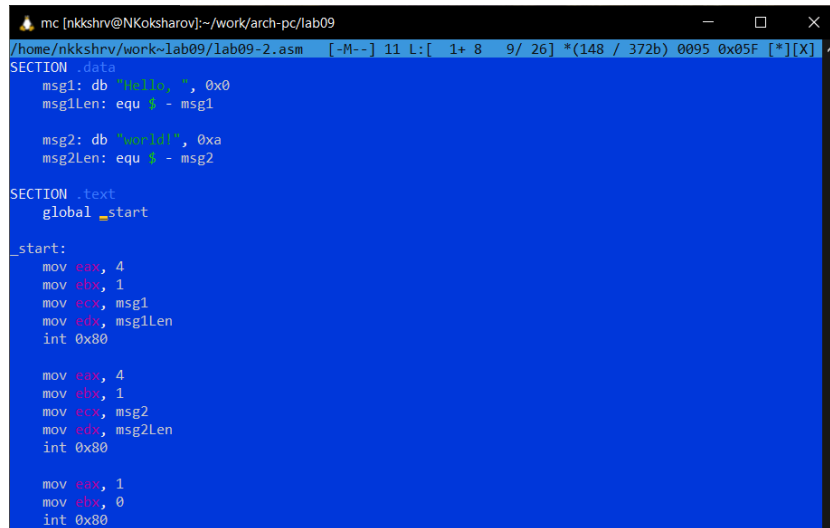
```
nkshrv@NKoksharov:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
nkshrv@NKoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
nkshrv@NKoksharov:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 1
f(g(x)) = 11
nkshrv@NKoksharov:~/work/arch-pc/lab09$
```

Рис. 2.5: Запуск измененного lab09-1

Ответ верный.

2.2 Отладка программ с помощью GDB

Создаю lab09-2.asm, пишу туда код из листинга 9.2 (рис. 2.6)



```
mc [nkshrv@NKoksharov:~/work/arch-pc/lab09]
/home/nkshrv/work~lab09/lab09-2.asm [-M--] 11 L: [ 1+ 8 9/ 26] *(148 / 372b) 0095 0x05F [*][X]
SECTION .data
msg1: db "Hello, ", 0x0
msg1len: equ $ - msg1

msg2: db "world!", 0xa
msg2len: equ $ - msg2

SECTION .text
global _start

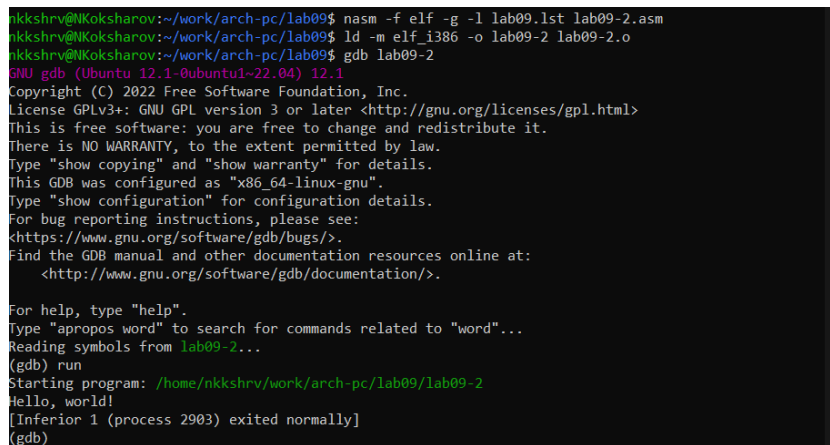
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1len
int 0x80

mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2len
int 0x80

mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 2.6: Код в lab09-2.asm

Создаю исполняемый файл и запускаю его с помощью отладчика (рис. 2.7)



```
nkshrv@NKoksharov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09.lst lab09-2.asm
nkshrv@NKoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
nkshrv@NKoksharov:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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-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 lab09-2...
(gdb) run
Starting program: /home/nkshrv/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 2903) exited normally]
(gdb)
```

Рис. 2.7: Запуск lab09-2 с помощью отладчика

Ставлю точку останова на метку _start и запускаю программу(рис. 2.8)


```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 12.
(gdb) run
Starting program: /home/nkshrv/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:12
12      mov eax, 4
(gdb)
```

Рис. 2.8: Запуск lab09-2 с брейкпоинтом

Изучаю дизассемблированный код от отметки `_start` (рис. 2.9)

```
(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)
```

Рис. 2.9: Дизассемблированный код от отметки `_start`

Переключаюсь на отображение команд с синтаксисом Intel (рис. 2.10)

```
(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)
```

Рис. 2.10: Дизассемблированный код от отметки `_start` (Intel-синтаксис)

Intel-синтаксис более читабелен в отличие от АТТ-синтаксиса, поскольку порядок указания регистров идентичен тому, как он указан в коде.

Включаю режим псевдографики (рис. 2.11)

```

nkkshrv@NKKoksharov: ~/work/arch-pc/lab09
--Register group: general--
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
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

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
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
0x8049036 <_start+54> int     0x80
0x8049038 add     BYTE PTR [eax],al
0x804903a add     BYTE PTR [eax],al
0x804903c add     BYTE PTR [eax],al

native process 2953 In: _start L12 PC: 0x8049000
(gdb) layout regs
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/nkkshrv/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:12
(gdb)

```

Рис. 2.11: Режим псевдографики

2.2.1 Добавление точек останова

С помощью команды `info breakpoints (i b)` узнаю, где установлены точки останова (рис. 2.12)

```

(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint        keep y   0x08049000 lab09-2.asm:12
      breakpoint already hit 1 time
(gdb)

```

Рис. 2.12: Точки останова при отладке lab09-2

Ставлю точку останова на предпоследней инструкции (рис. 2.13)

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 25.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y  0x08049000 lab09-2.asm:12
          breakpoint     already hit 1 time
2        breakpoint     keep y  0x08049031 lab09-2.asm:25
(gdb)
```

Рис. 2.13: Установка второго брейкпоинта

2.2.2 Работа с данными программы в GDB

Выполняю 5 инструкций с помощью stepi. Изменяются значения регистров eax, ebx, ecx, edx (рис. 2.14 - 2.18)

The screenshot shows the GDB interface with the 'Register group: general' window at the top. The registers and their values are as follows:

Register	Value
eax	0x4
ecx	0x0
edx	0x0
ebx	0x0
esp	0xffffd1d0
ebp	0x0
esi	0x0
edi	0x0
eip	0x8049005
eflags	0x202
cs	0x23
ss	0x2b
ds	0x2b

Below the register window, the assembly instructions are displayed. The instruction at address 0x8049005 is highlighted:

```
0x8049005 <_start+5> mov ebx,0x1
```

The bottom panel shows the GDB command line with the following commands and output:

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y  0x08049000 lab09-2.asm:12
          breakpoint     already hit 1 time
(gdb) mov ebx, 0x0
Undefined command: "mov". Try "help".
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 25.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y  0x08049000 lab09-2.asm:12
          breakpoint     already hit 1 time
2        breakpoint     keep y  0x08049031 lab09-2.asm:25
(gdb) si
(gdb)
```

Рис. 2.14: Изменение регистра eax

```

nkshrv@NKoksharov: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x804900a 0x804900a <_start+10>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

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
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
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al

native process 2981 In: _start L14 PC: 0x804900a

```

Рис. 2.15: Изменение регистра ebx

```

nkshrv@NKoksharov: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x0      0
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x804900f 0x804900f <_start+15>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

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
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
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al

native process 2981 In: _start L15 PC: 0x804900f

```

Рис. 2.16: Изменение регистра ecx

```

nkshrv@NKoksharov: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049014 0x8049014 <_start+20>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

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
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
0x8049036 <_start+54>   int    0x80
0x8049038          add    BYTE PTR [eax],al

native process 2981 In: _start L16 PC: 0x8049014

```

Рис. 2.17: Изменение регистра edx

```

nkshrv@NKoksharov: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

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
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
0x8049036 <_start+54>   int    0x80
0x8049038          add    BYTE PTR [eax],al

native process 2981 In: _start L18 PC: 0x8049016

```

Рис. 2.18: Изменение регистра eax

Узнаю содержимое регистров (рис. 2.19)

```

eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 2.19: Содержимое регистров

Узнаю содержимое msg1 по имени, msg2 по имени и адресу (рис. 2.20)

```

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

```

Рис. 2.20: Содержимое msg1, msg2

Меняю первый символ переменной msg1 (рис. 2.21)

```

(gdb) set{char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb)

```

Рис. 2.21: Изменение msg1

Меняю символ переменной msg2 (рис. 2.22)

```

(gdb) set{char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "Wor1d!\n\034"
(gdb)

```

Рис. 2.22: Изменение msg2

Вывожу значения регистра edx в шестнадцатеричном, двоичном и символьном видах (рис. 2.23)

```
(gdb) p/x $edx
$9 = 0x8
(gdb) p/t $edx
$10 = 1000
(gdb) p/s $edx
$11 = 8
(gdb)
```

Рис. 2.23: Значения регистра edx

Меняю значение регистра ebx (рис. 2.24)

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$12 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$13 = 2
(gdb)
```

Рис. 2.24: Изменение значения регистра ebx

Разница вывода команд `p/s $ebx` заключается в том, что сначала регистру присвоена строка `'2'`, а потом число `2`. Эти символы соответствуют разным кодам в UTF-8 (запрашивается значение именно в символьном виде).

2.2.3 Обработка аргументов командной строки в GDB

Копирую файл `lab8-2.asm`, создаю исполняемый файл (рис. 2.25)

```
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$
```

Рис. 2.25: Создание lab09-3

Запускаю отладку `lab09-3`, ставлю брейкпоинт перед первой инструкцией (рис. 2.26)

```

nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ gdb --args lab09-3 1 2 '3'
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 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-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 lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 7.
(gdb) run
Starting program: /home/nkkshrv/work/arch-pc/lab09/lab09-3 1 2 3

Breakpoint 1, _start () at lab09-3.asm:7
7       pop ecx
(gdb)

```

Рис. 2.26: Отладка lab09-3

Обращаюсь к первым пяти позициям стека, где хранятся число аргументов командной строки, имя программы, и три аргумента (рис. 2.27)

```

(gdb) x/x $esp
0xffffd1b0: 0x00000004
(gdb) x/x $(esp+4)
No symbol "esp" in current context.
(gdb) x/s *(void**)(esp + 4)
0xffffd31a: "/home/nkkshrv/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd343: "1"
(gdb) x/s *(void**)(esp + 12)
0xffffd345: "2"
(gdb) x/s *(void**)(esp + 16)
0xffffd347: "3"
(gdb) x/s *(void**)(esp + 20)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 2.27: Обращение к позициям стека

Шаг равен изменению адреса равен 4, потому что каждая ячейка занимает 4 байта в памяти.

2.3 Задание для самостоятельной работы

Преобразовываю код программы из лабораторной работы №8, реализуя вычисление значения функции как подпрограмму (рис. 2.28)


```

mc [nkkshrv@Nkoksharov:~/work/arch-pc/lab09
/home/nkkshrv/work~lab09/lab09-4.asm  [----] 14 L: [ 1+22 23/ 40] *(240 / 422b) 0010 0x00A [*][X] ^
#include "in_out.asm"

SECTION .data
msg db "Peppermint ", 0

SECTION .text
global _start

_start:

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

next:
    cmp ecx, 0h
    jz _end

    pop eax
    call atoi

    call _func

    loop next

_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintf
    call quit

_func:
    mov ebx, 5
    imul ebx
    add esi, eax
    add esi, 17
    ret

```

Рис. 2.28: Код в lab09-4.asm

Создаю исполняемый файл и запускаю его (рис. 2.29)

```

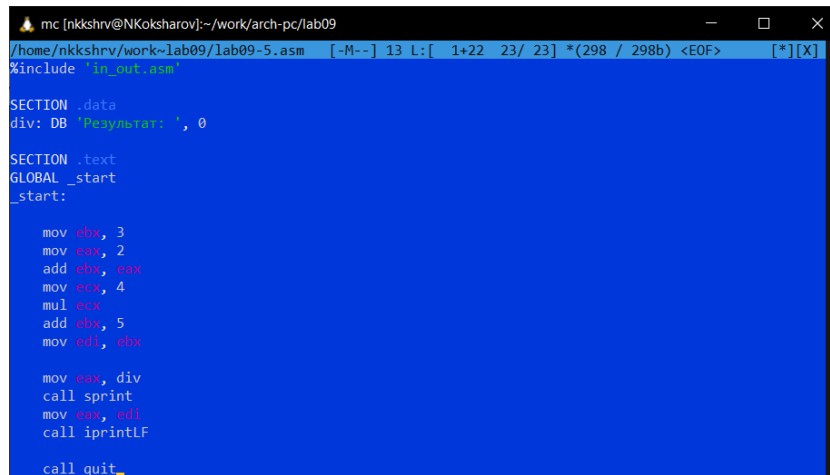
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ./lab09-4 1
-bash: ./lab09-4: No such file or directory
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ./lab09-4 1
Результат: 22
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ./lab09-4 1 1 1 1
Результат: 88
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ./lab09-4 1 1 2
Результат: 71
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$

```

Рис. 2.29: Запуск lab09-4

Ответ для разного набора аргументов верный.

Создаю файл lab09-5.asm и пишу в нем код программы из листинга 9.3 (рис. 2.30)



```
mc [nkkshrv@NKKoksharov]:~/work/arch-pc/lab09
/home/nkkshrv/work~lab09/lab09-5.asm [-M--] 13 L: [ 1+22 23/ 23] *(298 / 298b) <EOF> [*][X]
#include 'in_out.asm'

SECTION .data
div: DB 'Peaymar: ', 0

SECTION .text
GLOBAL _start
_start:

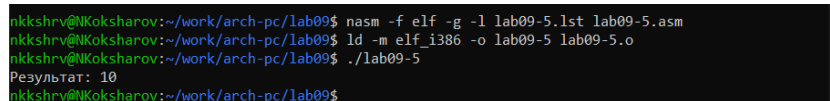
    mov ebx, 3
    mov eax, 2
    add ebx, eax
    mov ecx, 4
    mul ecx
    add ebx, 5
    mov edi, ebx

    mov eax, div
    call sprint
    mov ecx, edi
    call iprintf

    call quit
```

Рис. 2.30: Код в lab09-5.asm

Создаю исполняемый файл и запускаю его (рис. 2.31)



```
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
nkkshrv@NKKoksharov:~/work/arch-pc/lab09$
```

Рис. 2.31: Запуск lab09-5

Результат действительно неверный: $(3 + 2) * 4 + 5 = 25$

С помощью отладчика пытаюсь найти ошибку в коде (рис. 2.32)

```

nkkshrv@NKKoksharov: ~/work/arch-pc/lab09
--Register group: general--
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1b0 0xffffd1b0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0

B+> 0x80490e8 <_start> mov $0x3,%ebx
0x80490ed <_start+5> mov $0x2,%eax
0x80490f2 <_start+10> add %eax,%ebx
0x80490f4 <_start+12> mov $0x4,%ecx
0x80490f9 <_start+17> mul %ecx
0x80490fb <_start+19> add $0x5,%ebx
0x80490fe <_start+22> mov %ebx,%edi
0x8049100 <_start+24> mov $0x804a000,%eax
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mov %edi,%eax
0x804910c <_start+36> call 0x8049086 <iprintfLF>
0x8049111 <_start+41> call 0x80490db <quit>
0x8049116 add %al,(%eax)
0x8049118 add %al,(%eax)
0x804911a add %al,(%eax)
0x804911c add %al,(%eax)

native process 3867 In: _start L10 PC: 0x80490e8
(gdb) layout regs
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/nkkshrv/work/arch-pc/lab09/lab09-5
Breakpoint 1, _start () at lab09-5.asm:10
(gdb)

```

Рис. 2.32: Отладка lab09-5

Нашедши ошибку, исправляю код (рис. 2.33)

```

mc [nkkshrv@NKKoksharov:~/work/arch-pc/lab09]
/home/nkkshrv/work~lab09/lab09-5.asm [-M--] 16 L: [ 1+15 16/ 23] *(214 / 298b) 0010 0x00A [*][X]
#include "in_out.asm"

SECTION .data
div: DB "Pesymotat: ", 0

SECTION .text
GLOBAL _start
_start:

    mov ebx, 3
    mov ecx, 2
    add eax, ebx
    mov ecx, 4
    mul ecx
    add eax, 5
    mov edi, eax

    mov ecx, div
    call sprint
    mov eax, edi
    call iprintfLF

    call quit

```

Рис. 2.33: Исправленный код в lab09-5.asm

Создаю исполняемый файл и запускаю его (рис. 2.34)

```
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
nkkshrv@Nkoksharov:~/work/arch-pc/lab09$
```

Рис. 2.34: Запуск исправленного lab09-5

Ответ верный.

3 Выводы

При выполнении лабораторной работы я приобрел навыки написания программ с использованием подпрограмм и ознакомился с возможностями отладки с помощью GDB.