Отчёт по лабораторной работе №9

По теме: Понятие подпрограммы. Отладчик GDB.

Выполнил: Пателепень Филипп Максимович, НММбд-04-24.

9.1. Цель работы

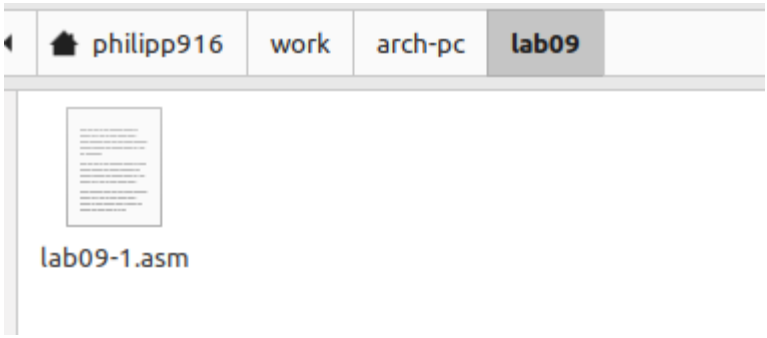
Приобретение навыков написания программ с использованием подпрограмм.
Знакомство с методами отладки при помощи GDB и его основными возможностями.

9.4. Ход выполнения лабораторной работы

9.4.1. Реализация подпрограмм в NASM

1. Я создал каталог для выполнения лабораторной работы № 9, перешёл в него и создал файл lab09-1.asm:

```
philipp916@philipp916-VirtualBox:~$ mkdir ~/work/arch-pc/lab09
philipp916@philipp916-VirtualBox:~$ cd ~/work/arch-pc/lab09
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ touch lab09-1.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$
```



2. Я ввел в файл lab09-1.asm текст программы из листинга 9.1. Создал исполняемый файл и проверьте его работу:

```
 GNU nano 4.8                /home/philipp916/work/arch-pc/lab09/lab09-1.asm
%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
```

```
 GNU nano 4.8                /home/philipp916/work/arch-pc/lab09/lab09-1.asm
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:
        mov ebx,2
        mul ebx
        add eax,7
        mov [res],eax

        ret

mov eax, msg
call sprint
```

```
mov ecx, x
mov edx, 80
call sread

mov eax,x
call atoi

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

ret
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 4
2x+7=15
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 11
2x+7=29
```

3. Я изменил 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:

```
 GNU nano 4.8              /home/philipp916/work/arch-pc/lab09/lab09-1.asm              Изменён
%include 'in_out.asm'
SECTION .data
        msg: DB 'Введите x: ',0
        pr1:  DB 'f(x) = 2x+7',0
        pr2:  DB 'g(x) = 3x-1',0
        result: DB 'f(g(x)) = ',0
SECTION .bss
        x: RESB 80
        res: RESB 80
SECTION .text
GLOBAL _start
        _start:

mov eax, pr1
call sprintLF
mov eax, pr2
call sprintLF
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:
        call _subcalcul
        mov ebx,2
        mul ebx
        add eax,7
        mov [res],eax

        ret

_subcalcul:
        mov ebx, 3
        mul ebx
        sub eax,1
        ret
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 2
f(g(x)) = 17
```

9.4.2. Отладка программ с помощью GDB

4. Я Создал файл lab09-2.asm с текстом программы из Листинга 9.2.:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ touch lab09-2.asm
```

```
 GNU nano 4.8                 /home/philipp916/work/arch-pc/lab09/lab09-2.asm                 Изменён
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
```

5. Я получил исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. Я загрузил исполняемый файл в отладчик gdb и проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r):

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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:
<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 lab09-2...
(gdb) run
Starting program: /home/philipp916/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 3023) exited normally]
(gdb)
```

6. Для более подробного анализа программы установил брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустил её:

```
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/philipp916/work/arch-pc/lab09/lab09-2

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

7. Далее я внимательно просмотрел дисассимилированный код программы с помощью команды disassemble начиная с метки _start:

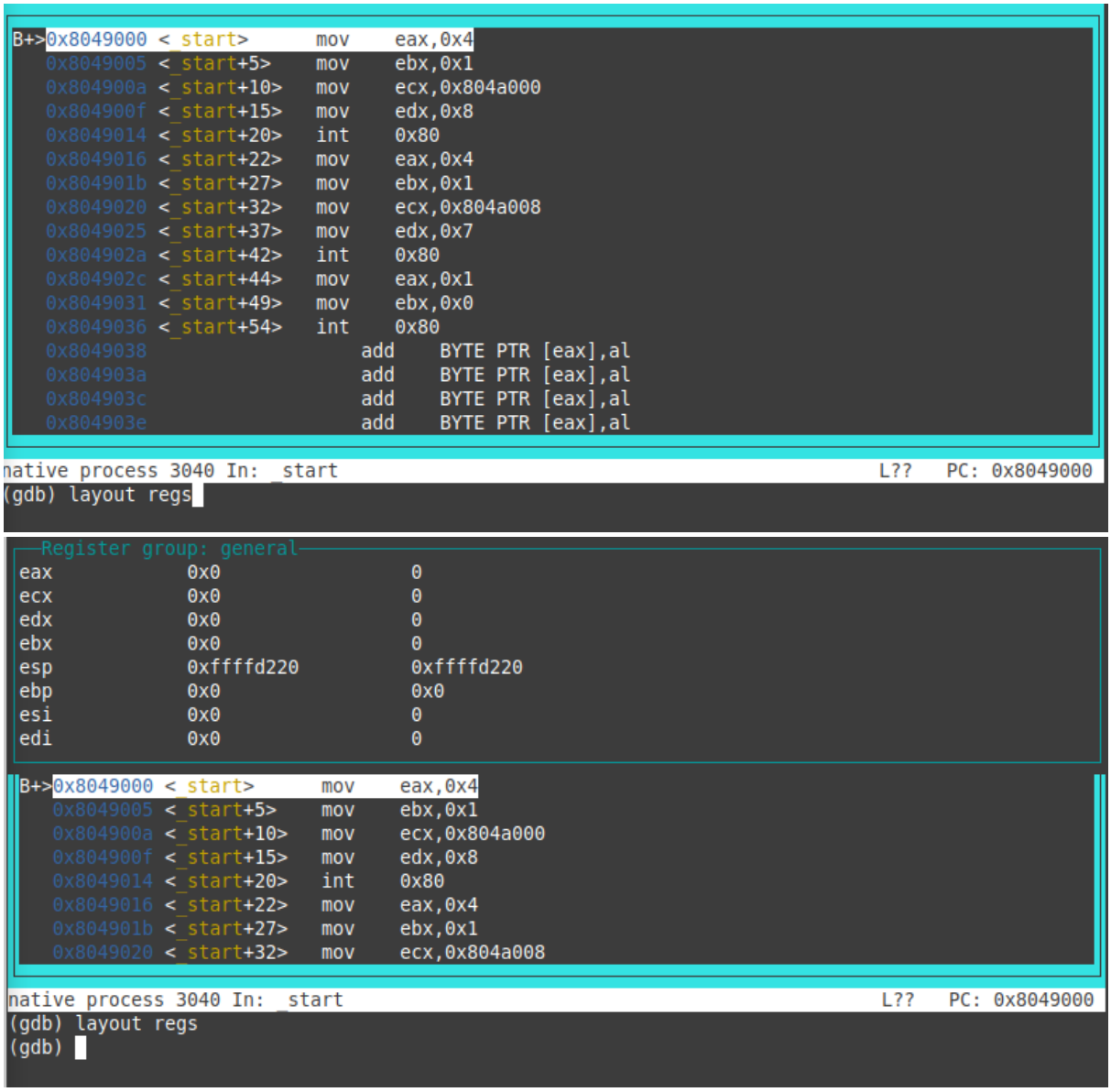
```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:     mov    $0x4,%eax
   0x08049005 <+5>:     mov    $0x1,%ebx
   0x0804900a <+10>:    mov    $0x804a000,%ecx
   0x0804900f <+15>:    mov    $0x8,%edx
   0x08049014 <+20>:    int    $0x80
   0x08049016 <+22>:    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)
```

8. Далее с помощью специальных команд я переключился на intel'овское отображение синтаксиса:

```
(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.
```

Отличие заключается в том, что в диссамилированном отображении в командах используют символы '%' и '$', а в Intel'овском отображении эти символы отсутствуют.

9. Следующим шагом я включил режим псевдографики для более удобного анализа программы:



9.4.2.1. Добавление точек останова(метки)

10. Я посмотрел наличие метки с помощью команды info breakpoints, а также установил еще одну метку по адресу инструкции:

```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08049000 <_start>
        breakpoint already hit 1 time
(gdb) i b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08049000 <_start>
        breakpoint already hit 1 time
2       breakpoint     keep y   0x08049031 <_start+49>
```

9.4.2.2. Работа с данными программы в GDB

11. С помощью команды info registers (i r) я посмотрел значение регистров:

```
native process 3040 In: _start
eax            0x0                 0
ecx            0x0                 0
edx            0x0                 0
ebx            0x0                 0
esp            0xffffd220          0xffffd220
ebp            0x0                 0x0
esi            0x0                 0
edi            0x0                 0
--Type <RET> for more, q to quit, c to continue without paging--
```

12. Далее я посмотрел значение переменной msg1 по имени:

```
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>:       "Hello, "
```

13. Посмотрел значение второй переменной msg2 по адресу:

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:       "world!\n"
```

14. С помощью команды set я изменил первый символ переменной msg1:

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:       "hello, "
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:       "hhllo, "
```

15. Аналогично изменил переменную msg2:

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:       "Lor d!\n"
```

16. Я изменил значение регистра ebx с помощью команды set:

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
```

Команда выводит различные значения т.к. в первый раз мы вносим в регистр значение 2, а во втором случае присваиваем регистру значение 2

17. Я завершил работу с файлом в отладчике с помощью комманд ‘c’, ‘si’ и ‘quit’:

```
(gdb) c
Continuing.
hhllo, Lor d!

Breakpoint 2, 0x08049031 in _start ()
```

9.4.2.3. Обработка аргументов командной строки в GDB

18. Я скопировал 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, а также создал исполняемый файл:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/a
rch-pc/lab09/lab09-3.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
```

19. Далее я загрузил исполняемый файл в отладчик, указав аргументы: gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3' . В отладчике поставил метку на _start и запустил программу:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'ар
гумент 3'
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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:
<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 lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) run
Starting program: /home/philipp916/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, 0x080490e8 in _start ()
(gdb)
```

20. Я проверил адрес вершины стека и убедился, что там хранится 5 элементов:

```
(gdb) x/x $esp
0xffffd1e0:     0x00000005
```

21. Затем я посмотрел все позиции стека:

```
(gdb) x/s *(void**)($esp + 4)
0xffffd390:     "/home/philipp916/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd3bc:     "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd3ce:     "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd3df:     "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd3e1:     "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
```

По первому адресу хранится сам адрес, в остальных адресах, в свою очередь, хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт. Для того чтобы данные сохранялись правильно, компьютер использует новый стек для новой информации.

9.5. Выполнение самостоятельной работы

1. Сначала я создал файл для выполнения первого задания самостоятельной работы под названием lab09-4.asm:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ touch lab09-4.asm
```

2. Затем преобразовал свой код программы из лабораторной работы №8 и реализовал вычисления как подпрограмму:

```
 GNU nano 4.8              /home/philipp916/work/arch-pc/lab09/lab09-4.asm
%include 'in_out.asm'
section .data
Pr db 'f(x)=15x+2',0
O db 'Результат: ',0
section .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,Pr
call sprintLF
next:
cmp ecx,0
jz _end
pop eax
all atoi
call P2
add esi,eax
loop next
_end:
mov eax,O
call sprint
mov eax,esi
call iprintLF
call quit
P2:
mov ebx,15
mul ebx
add eax,2
ret
```

3. Создал исполняемый файл и запустил его, чтобы проверить правильность выполнения программы. Программа работает исправно:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 1 2 3 4
f(x)=15x+2
Результат: 158
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 1 2
f(x)=15x+2
Результат: 49
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-4 2 3 4
f(x)=15x+2
Результат: 141
```

4. Для выполнения второго задания я создал файл lab09-5.asm:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ touch lab09-5.asm
```

5. С помощью листинга 9.3 я написал код программы. Далее создал исполняемый файл и запустил его:

```
 GNU nano 4.8              /home/philipp916/work/arch-pc/lab09/lab09-5.asm              Изменён
%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 ecx,4
mul ecx
add ebx,5
mov edi,ebx

mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
```

6. Далее я выявил ошибку, связанную с неправильным вычислением программы, запустил программу в отладчике:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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:
<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 lab09-5...
(No debugging symbols found in lab09-5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) run
Starting program: /home/philipp916/work/arch-pc/lab09/lab09-5

Breakpoint 1, 0x080490e8 in _start ()
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:     mov    ebx,0x3
   0x080490ed <+5>:     mov    eax,0x2
   0x080490f2 <+10>:    add    ebx,eax
   0x080490f4 <+12>:    mov    ecx,0x4
   0x080490f9 <+17>:    mul    ecx
   0x080490fb <+19>:    add    ebx,0x5
   0x080490fe <+22>:    mov    edi,ebx
   0x08049100 <+24>:    mov    eax,0x804a000
   0x08049105 <+29>:    call   0x804900f <sprint>
   0x0804910a <+34>:    mov    eax,edi
   0x0804910c <+36>:    call   0x8049086 <iprintLF>
   0x08049111 <+41>:    call   0x80490db <quit>
End of assembler dump.
(gdb)
```

7. Я открыл регистры и стал внимательно их изучать. Заметил, что не все регистры стоят на своих местах:

```
─Register group: general─
eax            0x0                 0
ecx            0x0                 0
edx            0x0                 0
ebx            0x0                 0
esp            0xffffd220          0xffffd220
ebp            0x0                 0x0
esi            0x0                 0
edi            0x0                 0

B+>0x80490e8 <_start>       mov    ebx,0x3
   0x80490ed <_start+5>     mov    eax,0x2
   0x80490f2 <_start+10>    add    ebx,eax
   0x80490f4 <_start+12>    mov    ecx,0x4
   0x80490f9 <_start+17>    mul    ecx
   0x80490fb <_start+19>    add    ebx,0x5
   0x80490fe <_start+22>    mov    edi,ebx
   0x8049100 <_start+24>    mov    eax,0x804a000

native process 5214 In: _start                                L??   PC: 0x80490e8
(gdb) layout regs
```

8. После изменения регистров я запустил программу. Программа стала работать корректно и вывела в терминале ответ 25:

```
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
philipp916@philipp916-VirtualBox:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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:
<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 lab09-5...
(gdb) r
Starting program: /home/philipp916/work/arch-pc/lab09/lab09-5
Результат: 25
[Inferior 1 (process 5310) exited normally]
```

Вывод

Вывод: я приобрел навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями.

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/Learningbash-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).