

Лабораторная работа №9

Дисциплина: Архитектура компьютера

Жибицкая Евгения Дмитриевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	14
4	Выводы	16

Список иллюстраций

2.1	Создание каталога и файла	6
-----	-------------------------------------	---

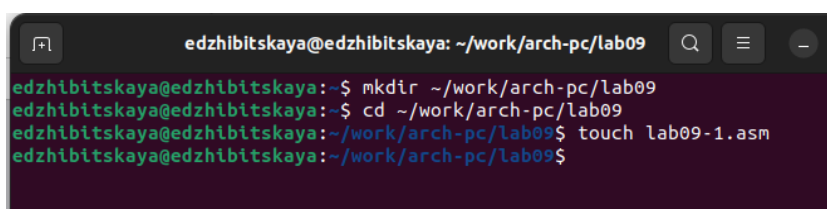
Список таблиц

1 Цель работы

Продолжение изучения языка ассемблера, использование подпрограмм и знакомство с методами отладки при помощи GDB.

2 Выполнение лабораторной работы

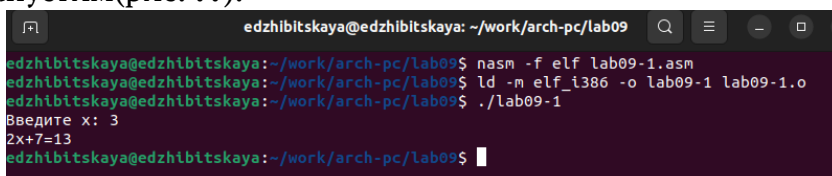
Для начала создадим каталог для 9 лабораторной работы, создадим файл и не забудем скопировать файл `in_out.asm` в этот же каталог(рис. 2.1).



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

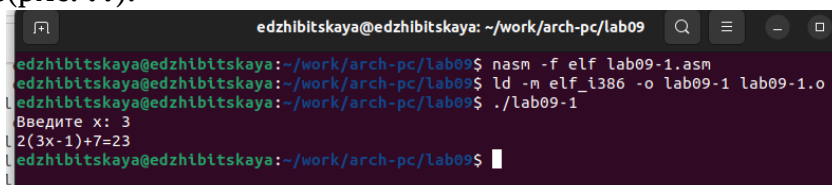
Рис. 2.1: Создание каталога и файла

Затем заполним файл текстом из Листинга 9.1, создадим исполняемый файл и запустим(рис. ??).



```
edzhbitskaya@edzhbitskaya: ~/work/arch-pc/lab09
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
2x+7=13
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$
```

Потом изменим программу, добавив подпрограмму `_subcalcul` и также запустим ее(рис. ??).



```
edzhbitskaya@edzhbitskaya: ~/work/arch-pc/lab09
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
2(3x-1)+7=23
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$
```

Далее добавим файл `lab09-2.asm` с текстом программы из Листинга 9.2, создадим исполняемый файл с ключом `-g` для отладочной информации и загрузим его в отладчик(рис. ??).

```

edzhibitskaya@edzhibitskaya:~$ cd ~/work/arch-pc/lab09
edzhibitskaya@edzhibitskaya:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst
lab09-2.asm
edzhibitskaya@edzhibitskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
edzhibitskaya@edzhibitskaya:~/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)

```

Затем командой `run` запустим программу(рис. ??), также добавим брейкпоинт и еще раз запустим ее(рис. ??).

```

(gdb) run
Starting program: /home/edzhibitskaya/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 7237) exited normally]
(gdb)

edzhibitskaya@edzhibitskaya: ~/work/arch-pc/lab09
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/edzhibitskaya/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Посмотрим дисассимилированный код программы командой `disassemble`(рис. ??), также переключимся на отображение с Intel'овским синтаксисом(рис. ??). Отличаются порядком операндов, записью числовых констант, наличием/отсутствием % перед именем регистра и тд.

```
edzhibitskaya@edzhibitskaya: ~/work/arch-p
(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)

(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)
```

Далее включим режим псевдографики и сразу посмотрим уже установленные ранее точки останова(рис. ??).


```

edzhibitskaya@edzhibitskaya: ~/work/arch-pc/lab09
[ Register Values Unavailable ]

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
0x804903a add BYTE PTR [eax],al
0x804903c add BYTE PTR [eax],al

native process 7653 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 *0x08049031

```

Установим еще один брейкпоинт и еще раз проверим информацию о всех точках останова(рис. ??).

```

edzhibitskaya@edzhibitskaya: ~/work/arch-pc/lab09
[ Register Values Unavailable ]

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
0x804903a add BYTE PTR [eax],al
0x804903c add BYTE PTR [eax],al

native process 7653 In: _start
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb)

```

Чтобы посмотреть содержимое регистров введем info registers(рис. ??).

```

edzhibitskaya@edzhibitskaya: ~/work/arch-pc/lab09
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd230 0xffffd230
lab09-2.asm
B+> 9  mov eax, 4
    10 mov ebx, 1
    11 mov ecx, msg1
    12 mov edx, msg1Len
    13 int 0x80

native process 9609 In:  _start      L9      PC: 0x8049000
Breakpoint 1, _start () at lab09-2.asm:9
(gdb) i r
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd230 0xffffd230
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--

```

Затем посмотрим значение переменной msg1 по имени и msg2 по адресу(рис. ??).

```

edzhibitskaya@edzhibitskaya: ~/work/arch-pc/lab09
Register group: general
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]

    13 int 0x80
    14 mov eax, 4
    15 mov ebx, 1
    16 mov ecx, msg2
    17 mov edx, msg2Len

native process 9609 In:  _start
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>:  "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:  "world!\n\034"
(gdb)

```

После инструкции set изменим символы в переменных msg1 и msg2(рис. ??).

```

(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "World!\n\034"
(gdb)

```

Наконец командой print посмотрим значения регистра(в 16ричном,20ичном и символьном форматах)(рис. ??) и изменим значение регистра ebx(символ и число)(рис. ??).

До полной разрядки!

```

edzhibitskaya@edzhibit
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd230 0xffffd230

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

native process 9933 In: _start
A syntax error in expression, near `2'.
(gdb) p/x $edx
$6 = 0x8
(gdb) p/s $edx
$7 = 8
(gdb) p/t $edx
$8 = 1000
(gdb)

```

```

register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd230 0xffffd230

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

native process 9933 In: _start
$8 = 1000
(gdb) set $ebx='2'
(gdb) p/s $ebx
$9 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$10 = 2
(gdb) █

```

Следующим этапом скопируем файл из предыдущей работы и создадим исполняемый файл(рис. ??).

```

edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/ar
ch-pc/lab09/lab09-3.asm
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
edzhbitskaya@edzhbitskaya:~/work/arch-pc/lab09$ █

```

Загрузим эту программу в отладчик использовав ключ -args, установим точку останова и запустим ее(рис. ??).

```

edzhibitskaya@edzhibitskaya:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент
1 аргумент 2 'аргумент 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 5.
(gdb) run
Starting program: /home/edzhibitskaya/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5   pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)

```

В конце посмотрим, что число аргументов равно 5 - посмотрим остальные позиции стека по адресу(рис. ??).

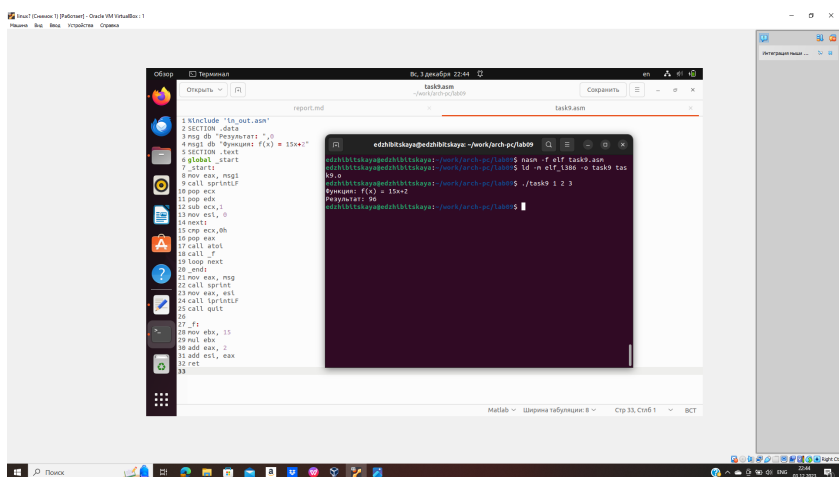
```

(gdb) x/x $esp
0xffffd1e0: 0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xffffd399: "/home/edzhibitskaya/work/arch-pc/lab09/lab09-3"
(gdb)
0xffffd3c8: "аргумент1"
(gdb) x/s *(void**)( $esp + 8)
0xffffd3c8: "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xffffd3da: "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xffffd3eb: "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffd3ed: "аргумент 3"
(gdb) x/s *(void**)( $esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

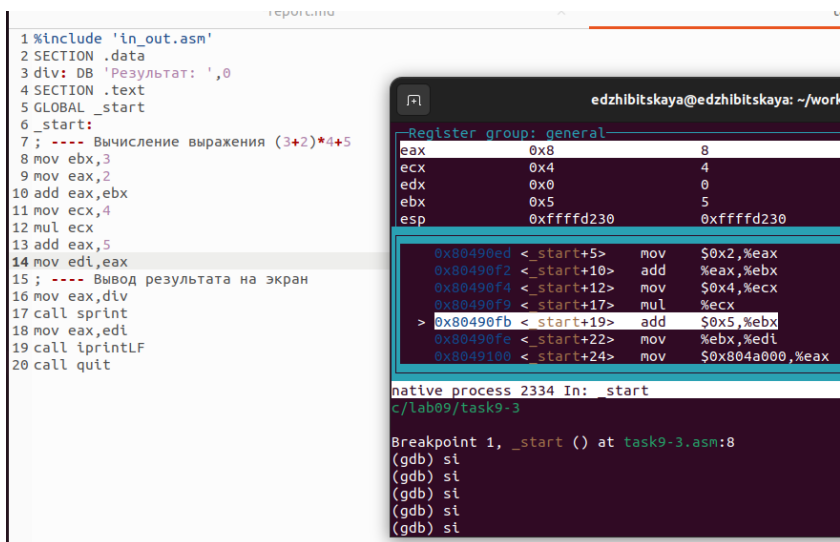
```

3 Задание для самостоятельной работы

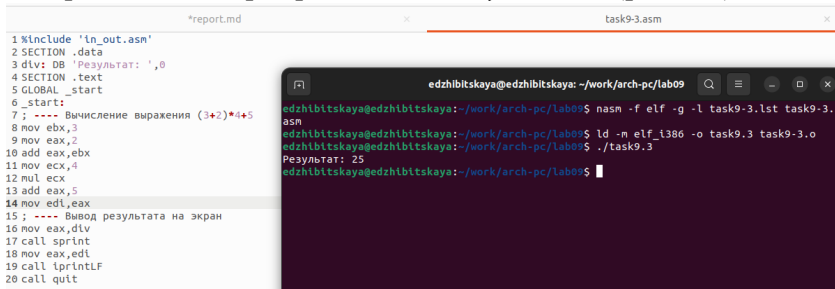
1. Преобразуем программу из 8 лабораторной работы, реализовав вычисление функции как подпрограммы(рис. ??).



2. Определение и исправление ошибки из Лиситнга 9.3 С помощью отладчика gdb, поочередно выполняя команды и следя за значениями регистров, видим, что ошибка возникает из-за неправильной записи результата сложения в регистр ebx, а не eax, так как при последующем умножении, умножается именно регистр eax(рис. ??).



Исправим код программы и запустим ее(рис. ??).



4 Выводы

В ходе работы было произведено знакомство с методами и возможностями отладки программ, применены подпрограммы.