

Отчёт по лабораторной работе №9

Архитектура компьютера

Алехин Давид Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	lab09/lab9-1.asm	8
4.2	Листинг 9.1	8
4.3	Изменённый 9.1	9
4.4	./9.1 изменённый	9
4.5	Отладка 9.2	10
4.6	./9.2 через отладчик	10
4.7	Брежпоинт на метку _start	10
4.8	Дисассимпированный код	11
4.9	Intel'овское отображение	11
4.10	Псевдографика	12
4.11	Смотрю наличие брейкпоинтов	12
4.12	Добавляю метку	12
4.13	Просмотр регистров	13
4.14	Просмотр значения переменной	13
4.15	Значение переменной msg2	13
4.16	Изменение значения переменной	13
4.17	Изменение msg2	14
4.18	Значение регистров ехх и еах	14
4.19	Меняю значение регистров ебх	14
4.20	Запуск файла в отладчике	15
4.21	Запуск файла lab10-3 через метку	15
4.22	Адрес вершины стека	15
4.23	Все позиции стека	16
4.24	Текст программы	16
4.25	Запуск программы	16
4.26	Запуск программы	17
4.27	Запуск программы в отладчике	17
4.28	Анализ регистров	18
4.29	Запуск	18

Список таблиц

1 Цель работы

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

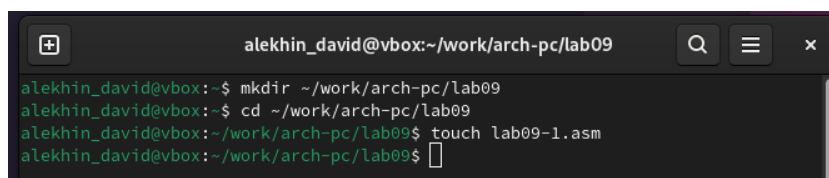
2 Задание

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга. Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

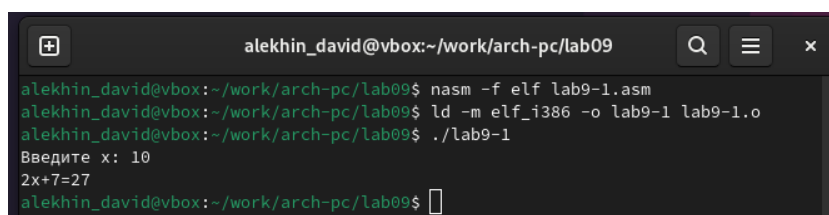
Создаю lab09/lab9-1.asm. (рис. 4.1).

A terminal window titled 'alekhin_david@vbox:~/work/arch-pc/lab09' with search, menu, and close icons. The command history shows: 'mkdir ~/work/arch-pc/lab09', 'cd ~/work/arch-pc/lab09', and 'touch lab09-1.asm'. The prompt is now 'alekhin_david@vbox:~/work/arch-pc/lab09\$' with a cursor.

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

Рис. 4.1: lab09/lab9-1.asm

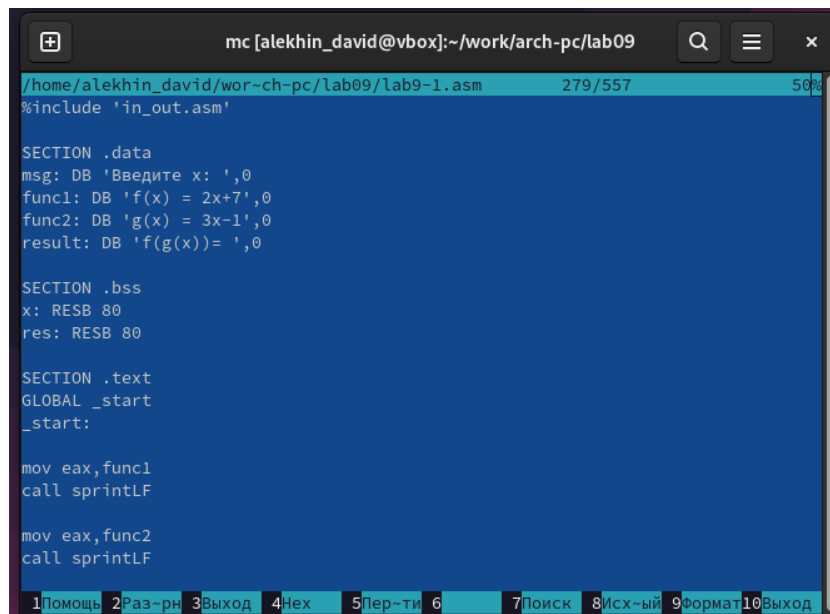
Вставляю туда Листинг 9.1, компаную и запускаю. (рис. 4.2).

A terminal window titled 'alekhin_david@vbox:~/work/arch-pc/lab09' with search, menu, and close icons. The command history shows: 'nasm -f elf lab9-1.asm', 'ld -m elf_i386 -o lab9-1 lab9-1.o', and './lab9-1'. The program output is 'Введите x: 10' and '2x+7=27'. The prompt is now 'alekhin_david@vbox:~/work/arch-pc/lab09\$' with a cursor.

```
alekhin_david@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
alekhin_david@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
alekhin_david@vbox:~/work/arch-pc/lab09$
```

Рис. 4.2: Листинг 9.1

Изменяю текст программы, чтобы она решала выражение $f(g(x))$ и запускаю. (рис. 4.3), (рис. 4.4).



```
mc [alekhin_david@vbox]:~/work/arch-pc/lab09
/home/alekhin_david/work/arch-pc/lab09/lab9-1.asm 279/557 50%
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
func1: DB 'f(x) = 2x+7',0
func2: DB 'g(x) = 3x-1',0
result: DB 'f(g(x))= ',0

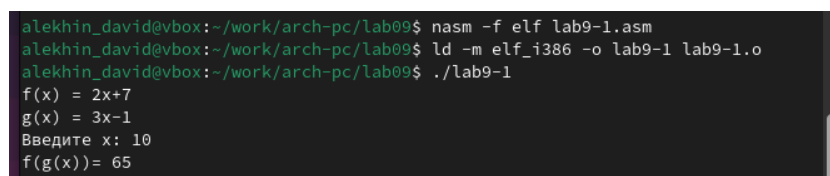
SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,func1
call sprintf

mov eax,func2
call sprintf
```

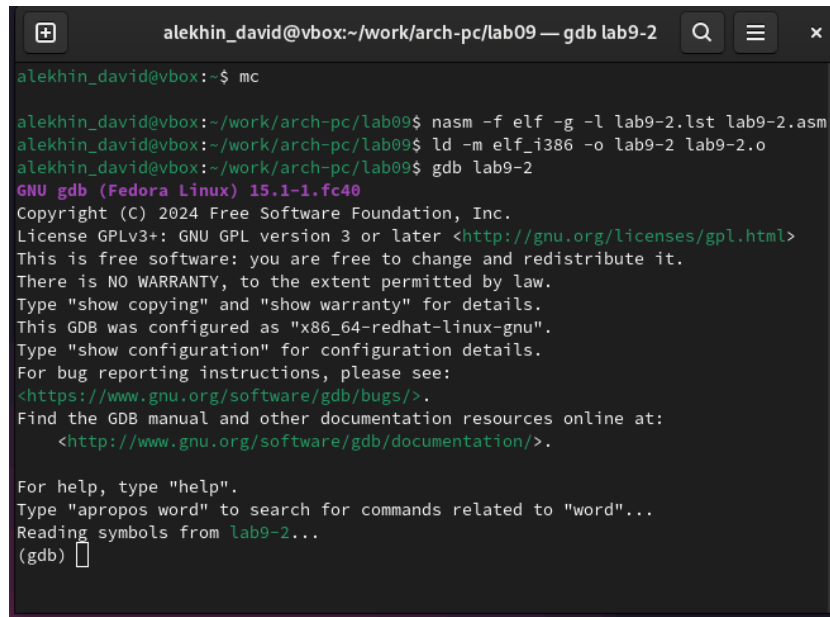
Рис. 4.3: Изменённый 9.1



```
alekhin_david@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
alekhin_david@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 10
f(g(x))= 65
```

Рис. 4.4: ./9.1 изменённый

Создаю файл вписываю туда текст Листинг 9.2, компаную и запускаю через отладчик. (рис. 4.5).

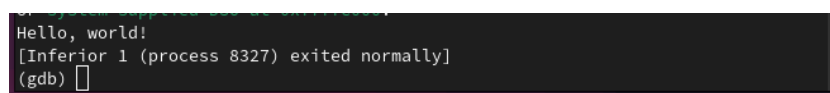


```
alekhin_david@vbox:~/work/arch-pc/lab09 — gdb lab9-2
alekhin_david@vbox:~$ mc
alekhin_david@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
alekhin_david@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
alekhin_david@vbox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.1-1.fc40
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-redhat-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 lab9-2...
(gdb)
```

Рис. 4.5: Отладка 9.2

Запускаю 9.2 через отладчик. (рис. 4.6).

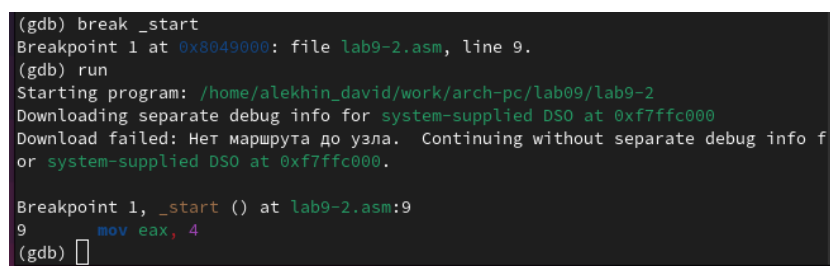


```

Hello, world!
[Inferior 1 (process 8327) exited normally]
(gdb)
```

Рис. 4.6: ./9.2 через отладчик

Ставлю брекпоинт на метку `_start` и запустил программу. (рис. 4.7).



```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/alekhin_david/work/arch-pc/lab09/lab9-2
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Download failed: Нет маршрута до узла. Continuing without separate debug info f
or system-supplied DSO at 0xf7ffc000.

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 4.7: Брекпоинт на метку `_start`

Смотрю дисассимплированный код программы начиная с метки. (рис. 4.8).

```
(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. 4.8: Дисассимплированный код

С помощью команды я переключился на intel'овское отображение синтаксиса. Отличие заключается в командах, в дисассимилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. (рис. 4.9).

```
(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. 4.9: Intel'овское отображение

Включаю режим псевдографики. (рис. 4.10).

```

alekhin_david@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eflags    0x202    [ IF ]
cs        0x23    35
ss        0x2b    43
ds        0x2b    43
es        0x2b    43
fs        0x0     0
gs        0x0     0

0x80492c4    add    BYTE PTR [eax],al
0x80492c6    add    BYTE PTR [eax],al
0x80492c8    add    BYTE PTR [eax],al
0x80492ca    add    BYTE PTR [eax],al
0x80492cc    add    BYTE PTR [eax],al
0x80492ce    add    BYTE PTR [eax],al
0x80492d0    add    BYTE PTR [eax],al

native process 8597 (asm) In: _start    L9    PC: 0x8049000

```

Рис. 4.10: Псевдографика

Смотрю наличие брейкпоинтов. (рис. 4.11).

```

(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint      keep y   0x08049000  lab9-2.asm:9
breakpoint already hit 1 time
(gdb)

```

Рис. 4.11: Смотрю наличие брейкпоинтов

Добавляю метку на предпоследнюю строку. (рис. 4.12).

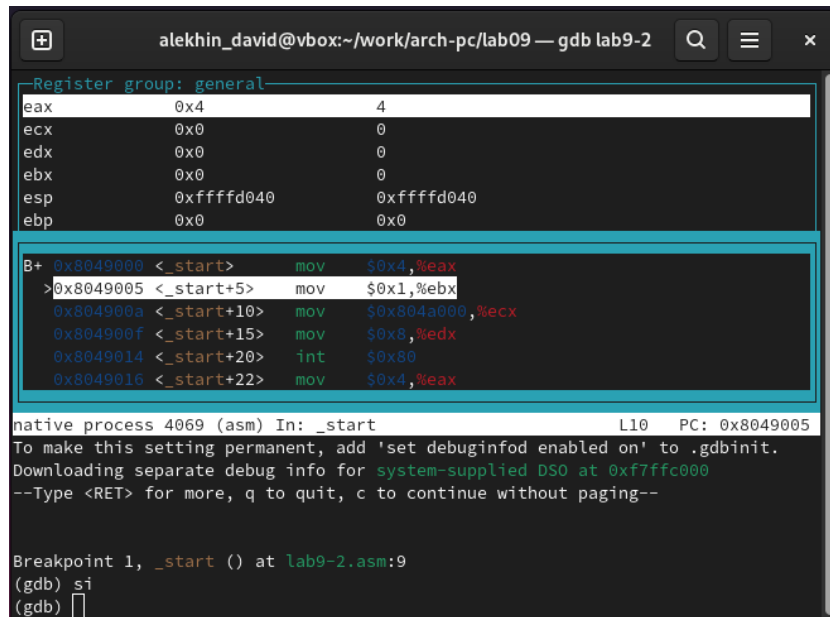
```

(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num      Type             Disp Enb Address      What
1        breakpoint      keep y   0x08049000  lab9-2.asm:9
2        breakpoint      keep y   0x08049031  lab9-2.asm:20
(gdb)

```

Рис. 4.12: Добавляю метку

С помощью команды si просматриваю регистры. (рис. 4.13).



```
alekhin_david@vbox:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd040 0xffffd040
ebp      0x0      0x0

B+ 0x8049000 <_start>  mov $0x4,%eax
>0x8049005 <_start+5>  mov $0x1,%ebx
0x804900a <_start+10>  mov $0x804a000,%ecx
0x804900f <_start+15>  mov $0x8,%edx
0x8049014 <_start+20>  int $0x80
0x8049016 <_start+22>  mov $0x4,%eax

native process 4069 (asm) In: _start      L10  PC: 0x8049005
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
--Type <RET> for more, q to quit, c to continue without paging--

Breakpoint 1, _start () at lab9-2.asm:9
(gdb) si
(gdb)
```

Рис. 4.13: Просмотр регистров

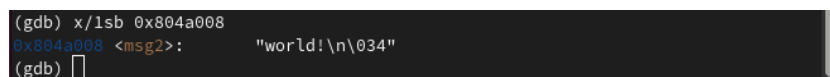
С помощью команды смотрю значение переменной msg1. (рис. 4.14).



```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb)
```

Рис. 4.14: Просмотр значения переменной

Просматриваю значение второй переменной msg2. (рис. 4.15).



```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb)
```

Рис. 4.15: Значение переменной msg2

С помощью команды set меняю значение переменной msg1. (рис. 4.16).



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

Рис. 4.16: Изменение значения переменной

Я изменяю переменную msg2. (рис. 4.17).

```
(gdb) x/1sb 0x804927e
0x804927e: "K"
(gdb) set {char} 0x804a008='K'
(gdb) set {char} 0x804a00b='3'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "Kor3d!\n\034"
(gdb) 
```

Рис. 4.17: Изменение msg2

Вывожу значение регистров ecx и eax. (рис. 4.18).

```
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb) 
```

Рис. 4.18: Значение регистров ecx и eax

Меняю значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум. (рис. 4.19).

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

Рис. 4.19: Меняю значение регистров ebx

Копирую файл lab8-2.asm и запускаю в отладчике. (рис. 4.20).

```

alekhin_david@vbox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
alekhin_david@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
alekhin_david@vbox:~/work/arch-pc/lab09$ gdb --args lab9-3 аргумент1 аргумент 2
'аргумент 3'
GNU gdb (Fedora Linux) 15.1-1.fc40
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-redhat-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) 

```

Рис. 4.20: Запуск файла в отладчике

Ставлю метку на `_start` и запускаю файл. (рис. 4.21).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/alekhin_david/work/arch-pc/lab09/lab9-3 аргумент1 аргуме
нт 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.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) 

```

Рис. 4.21: Запуск файла lab10-3 через метку

Проверяю адрес вершины стека. (рис. 4.22).

```

(gdb) x/x $esp
0xffffcfff: 0x00000005
(gdb) 

```

Рис. 4.22: Адрес вершины стека

Я смотрю все позиции стека. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации. (рис. 4.23).

```
(gdb) x/s *(void**)(esp + 4)
0xffffd1bb:  "/home/alekhin_david/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd1e0:  "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd1fb:  "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd20c:  "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd20e:  "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb) 
```

Рис. 4.23: Все позиции стека

Перехожу к самостоятельной работе. Преобразую программу из лабораторной работы №8 и реализую вычисления как подпрограмму. Запускаю. (рис. 4.24), (рис. 4.25).

```
lab9-1s.asm  [-M--]  0 L: [ 21+23  44/ 44] *(378 / 378b) <EOF>  [*][X]
cmp  ecx,0
jz  _end

pop  eax
call atoi
call fir
add  esi,eax

loop next

_end:
mov  eax,otv
call sprint
mov  eax,esi
call iprintLF
call quit

fir:
mov  ebx,2
mul  ebx
add  eax,15
ret

1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Пер-ть 7 Поиск 8 Уда-ть 9 МенюМС 10 Выход
```

Рис. 4.24: Текст программы

```
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-1s 1 12 13
f(x)=2x+15
Результат: 97
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-1s 11 12 13
f(x)=2x+15
Результат: 117
alekhin_david@vbox:~/work/arch-pc/lab09$ 
```

Рис. 4.25: Запуск программы

Копирую программу и запускаю, получается арифметическая ошибка.(рис. 4.26).


```

alekhin_david@vbox:~/work/arch-pc/lab09$ nasm -f elf lab9-2s.asm
alekhin_david@vbox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2s lab9-2s.o
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-2s
Результат: 10

```

Рис. 4.26: Запуск программы

Запускаю программу в отладчике. (рис. 4.27).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /home/alekhin_david/work/arch-pc/lab09/lab9-2s

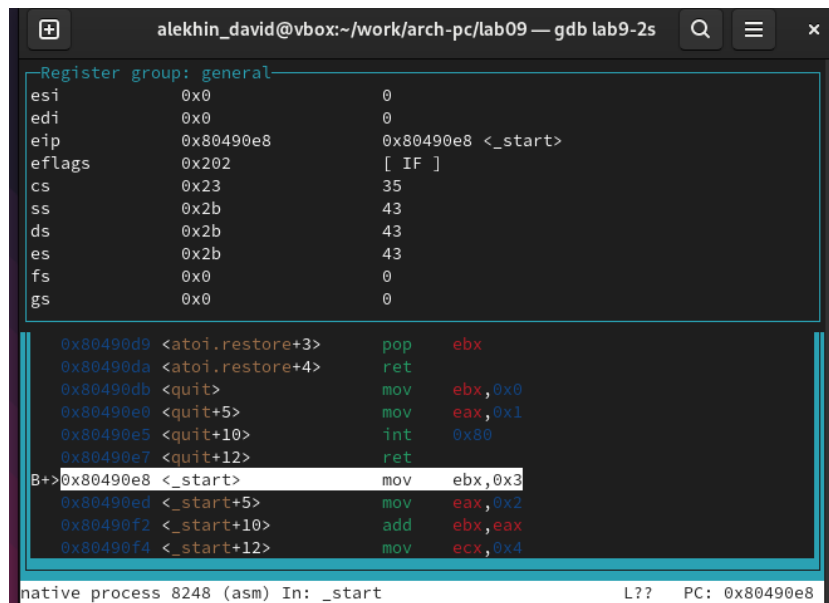
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, 0x80490e8 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)

```

Рис. 4.27: Запуск программы в отладчике

Анализирую регистры, некоторые из них стоят не на своих местах. (рис. 4.28).



The screenshot shows the GDB interface with the title bar 'alekhin_david@vbox:~/work/arch-pc/lab09 — gdb lab9-2s'. The 'Register group: general' section displays the following values:

Register	Value	Comment
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
gs	0x0	0

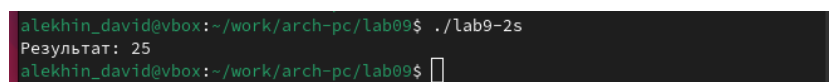
The assembly window shows the following code:

```
0x80490d9 <atoi.restore+3> pop    ebx
0x80490da <atoi.restore+4> ret
0x80490db <quit>          mov     ebx, 0x0
0x80490e0 <quit+5>         mov     eax, 0x1
0x80490e5 <quit+10>        int     0x80
0x80490e7 <quit+12>        ret
B>0x80490e8 <_start>       mov     ebx, 0x3
0x80490ed <_start+5>        mov     eax, 0x2
0x80490f2 <_start+10>       add     ebx, eax
0x80490f4 <_start+12>       mov     ecx, 0x4
```

The status bar at the bottom indicates 'native process 8248 (asm) In: _start' and 'PC: 0x80490e8'.

Рис. 4.28: Анализ регистров

Исправляю ошибки и запускаю программу, получаем верный результат. (рис. 4.29).



The screenshot shows a terminal window with the following output:

```
alekhin_david@vbox:~/work/arch-pc/lab09$ ./lab9-2s
Результат: 25
alekhin_david@vbox:~/work/arch-pc/lab09$
```

Рис. 4.29: Запуск

5 Выводы

Я приобрел навыки написания программ использованием подпрограмм. Обучился отладке программ.

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,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
 17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
 18. — 1120 с. — (Классика Computer Science).