

Отчёт по лабораторной работе 9

Архитектура компьютеров и операционные системы

Алексей Белов НПИбд-01-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	21
3	Выводы	28

Список иллюстраций

2.1	Программа в файле lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	8
2.3	Программа в файле lab9-1.asm	9
2.4	Запуск программы lab9-1.asm	10
2.5	Программа в файле lab9-2.asm	11
2.6	Запуск программы lab9-2.asm в отладчике	12
2.7	Дизассемблированный код	13
2.8	Дизассемблированный код в режиме интел	14
2.9	Точка остановки	15
2.10	Изменение регистров	16
2.11	Изменение регистров	17
2.12	Изменение значения переменной	18
2.13	Вывод значения регистра	19
2.14	Вывод значения регистра	20
2.15	Вывод значения регистра	21
2.16	Программа в файле lab9-4.asm	22
2.17	Запуск программы lab9-4.asm	23
2.18	Код с ошибкой	24
2.19	Отладка	25
2.20	Код исправлен	26
2.21	Проверка работы	27

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Я создал каталог для выполнения лабораторной работы № 9 и перешел в него. Затем я создал файл lab9-1.asm.

В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.(рис. [2.1]) (рис. [2.2])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
/home/adb~ab9-1.asm [----] 21 L:[ 1+29 30/ 30] *(450 / 462b) 1075 0
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80

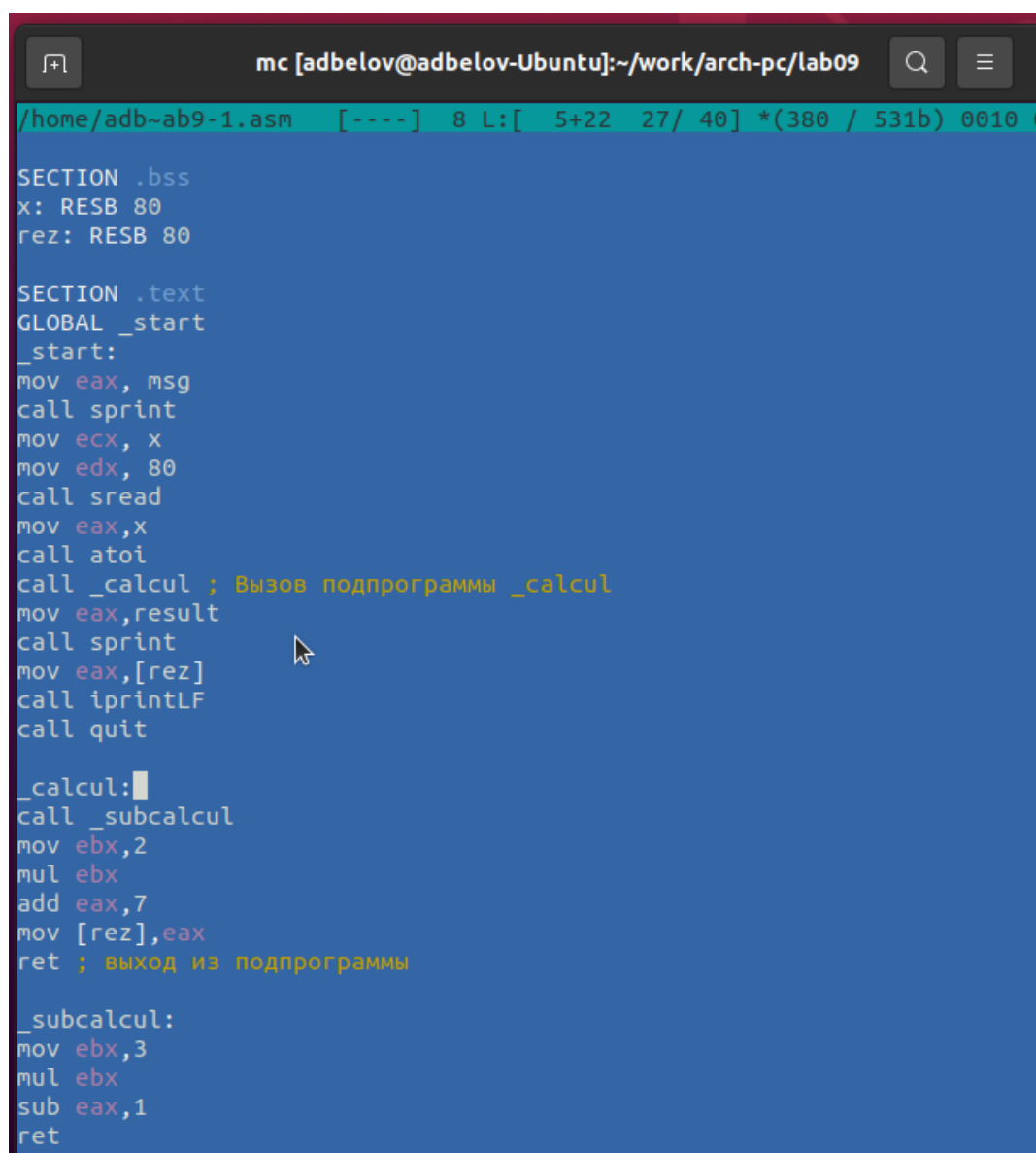
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

Рис. 2.1: Программа в файле lab9-1.asm

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

Изменил текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. (рис. [2.3]) (рис. [2.4])



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/lab09
/home/adbelov/lab9-1.asm [----] 8 L: [ 5+22 27/ 40] *(380 / 531b) 0010
SECTION .bss
x: RESB 80
rez: 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 ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

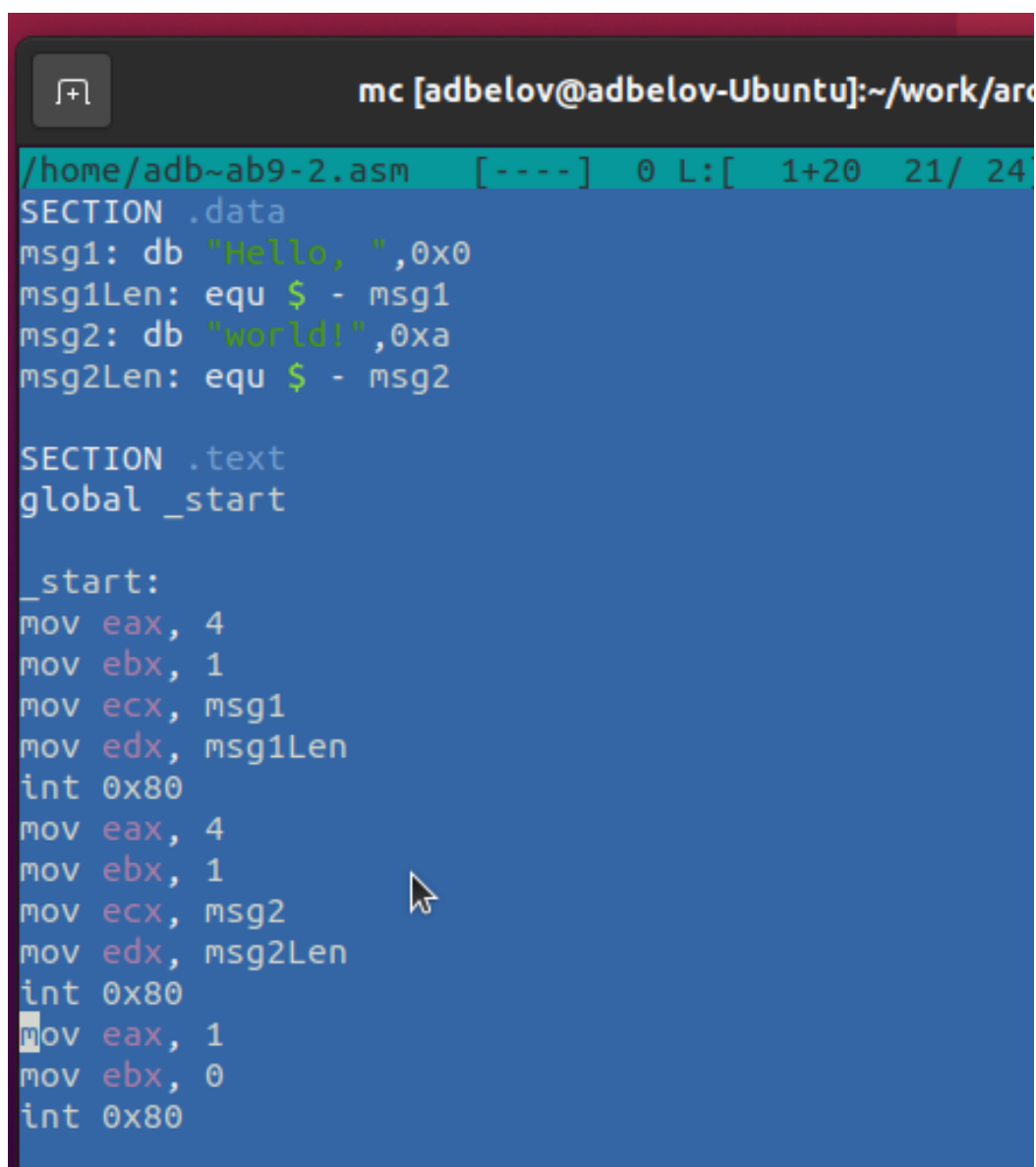
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 2.3: Программа в файле lab9-1.asm

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2(3x-1)+7=47
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ █
```

Рис. 2.4: Запуск программы lab9-1.asm

Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!). (рис. [2.5])



```
mc [adbelov@adbelov-Ubuntu]:~/work/arc  
/home/adbelov/lab9-2.asm [----] 0 L: [ 1+20 21/ 24  
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. 2.5: Программа в файле lab9-2.asm

Получил исполняемый файл и добавил отладочную информацию с помощью ключа '-g' для работы с GDB.

Загрузил исполняемый файл в отладчик GDB и проверил работу программы, запустив ее с помощью команды 'run' (сокращенно 'r'). (рис. [2.6])

```

adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 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 lab9-2...
(gdb) run
Starting program: /home/adbelov/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 6813) exited normally]
(gdb) █

```

Рис. 2.6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы, установил точку остановки на метке 'start', с которой начинается выполнение любой ассемблерной программы, и запустил ее. Затем просмотрел дизассемблированный код программы.(рис. [2.7]) (рис. [2.8])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
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 lab9-2...
(gdb) run
Starting program: /home/adbelov/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 6813) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000
(gdb) run
Starting program: /home/adbelov/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 in _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)
```

Рис. 2.7: Дизассемблированный код

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
Starting program: /home/adbelov/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 in _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) 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. 2.8: Дизассемблированный код в режиме интел

Для проверки точки остановки по имени метки '_start', использовал команду 'info breakpoints' (сокращенно 'i b'). Затем установил еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции 'mov ebx, 0x0'. (рис. [2.9])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd1e0 0xffffd1e0  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      gs       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

native process 6817 In: _start L?? PC: 0x8049000
(gdb)
(gdb)
(gdb) b *0x8049031Breakpoint 2 at 0x8049031
(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>
(gdb) |
```

Рис. 2.9: Точка остановки

В отладчике GDB можно просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды 'stepi' (сокращенно 'si') и отследил изменение значений регистров. (рис. [2.10]) (рис. [2.11])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd1e0 0xffffd1e0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>  eflags    0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       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

native process 6817 In: _start L?? PC: 0x8049005
eip      0x8049000 0x8049000 <_start>
eflags    0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
0x08049005 in _start ()
(gdb) 
```

Рис. 2.10: Изменение регистров


```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1e0  0xffffd1e0  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
fs       0x0      0      gs       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

native process 6817 In: _start L?? PC: 0x8049016
gs      0x0      0
(gdb) si
0x08049005 in _start ()
(gdb) si
0x0804900a in _start ()
(gdb) si
0x0804900f in _start ()
(gdb) si
0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) 
```

Рис. 2.11: Изменение регистров

Просмотрел значение переменной msg1 по имени и получил нужные данные.
Просмотрел значение переменной msg1 по имени и получил нужные данные.
Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1. (рис. [2.12])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

--Register group: general--
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1e0      0xffffd1e0      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
fs       0x0      0      gs       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

native process 6817 In: _start      L??      PC: 0x8049016
0x8049016 in _start ()
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "Hello, "
(gdb)
(gdb) x/1sb 0x804a0080x804a008 <msg2>:      "world!\n"
(gdb)
(gdb)
(gdb) x/1sb &msg10x804a000 <msg1>:      "hello, "
(gdb)
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n"
(gdb)
```

Рис. 2.12: Изменение значения переменной

Для изменения значения регистра или ячейки памяти использовал команду set, указав имя регистра или адрес в качестве аргумента. Изменил первый символ переменной msg1.(рис. [2.13])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd1e0 0xffffd1e0  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
fs       0x0      0      gs       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

native process 6817 In: _start L?? PC: 0x8049016
(gdb) p/t $eax$2 = 1000
(gdb)
(gdb) p/s $ecx$3 = 134520832
(gdb)
(gdb) p/x $ecx$4 = 0x804a000
(gdb)
(gdb) p/s $edx$5 = 8
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx на нужное значение.
(рис. [2.14])

```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09

Register group: general
eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x2      2
esp      0xffffd1e0  0xffffd1e0  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
fs       0x0      0      gs       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

native process 6817 In: start L?? PC: 0x8049016
(gdb) p/s $edx$5 = 8
(gdb)
(gdb) p/t $edx$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
(gdb)
(gdb) p/s $ebx$8 = 50
(gdb)
(gdb) p/s $ebx
$9 = 2
(gdb)
```

Рис. 2.14: Вывод значения регистра

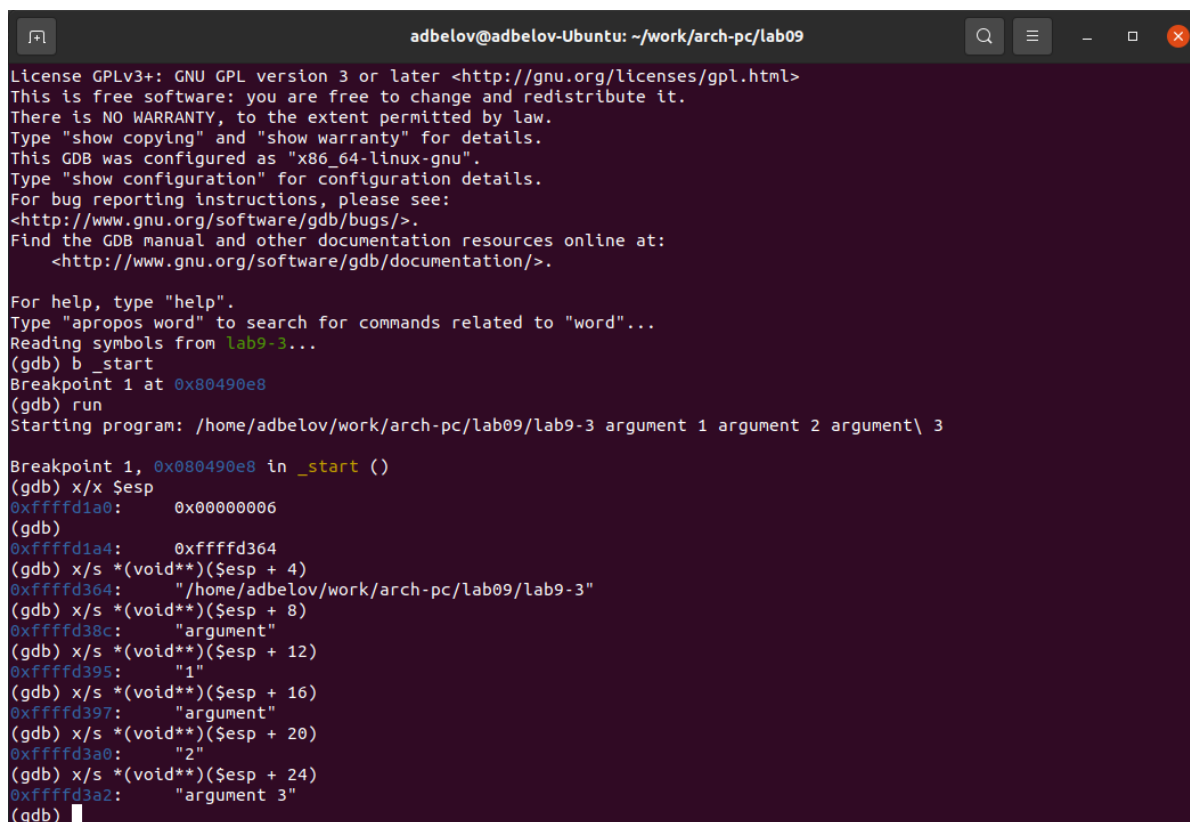
Скопировал файл lab8-2.asm, созданный во время выполнения лабораторной работы №8, который содержит программу для вывода аргументов командной строки. Создал исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в gdb использовал ключ `-args` и загрузил исполняемый файл в отладчик с указанными аргументами.

Установил точку останова перед первой инструкцией программы и запустил ее.

Адрес вершины стека, содержащий количество аргументов командной строки (включая имя программы), хранится в регистре `esp`. По этому адресу находится число, указывающее количество аргументов. В данном случае видно, что количество аргументов равно 5, включая имя программы `lab9-3` и сами аргументы: `аргумент1`, `аргумент2` и `'аргумент 3'`.

Просмотрел остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] - второго и так далее. (рис. [2.15])



```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x080490e8
(gdb) run
Starting program: /home/adbelov/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xffffd1a0: 0x00000006
(gdb)
0xffffd1a4: 0xffffd364
(gdb) x/s *(void**)(esp + 4)
0xffffd364: "/home/adbelov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd38c: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd395: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd397: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3a0: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3a2: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]).

2.1 Самостоятельное задание

Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму. (рис. [2.16]) (рис. [2.17])

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc
/home/adbelov/lab9-4.asm [----] 0 L:[ 3+18 21/ 38] *(
msg db "Результат: ",0
fx: db 'f(x)= 7 + 2x',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintf
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
call funk
add esi,eax

loop next

_end:
mov eax, msg
call sprintf
mov eax, esi
call iprintLF
call quit

funk:
mov ebx,2
mul ebx
add eax,7
ret
```

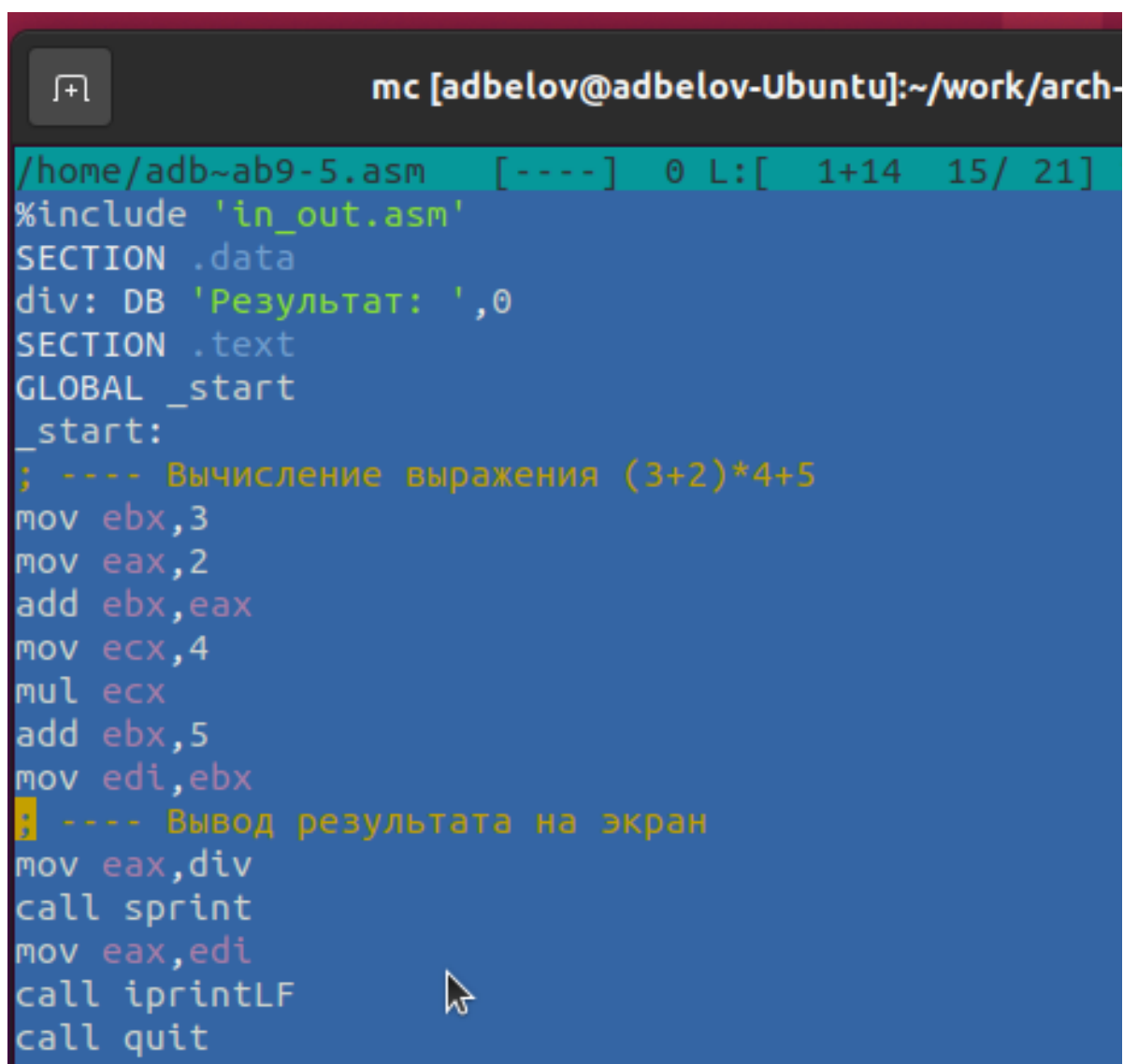
Рис. 2.16: Программа в файле lab9-4.asm

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-4 lab9-4.o
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ./lab9-4 1
f(x)= 7 + 2x
Результат: 9
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$ ./lab9-4 1 3 6 4 7 8 9
f(x)= 7 + 2x
Результат: 125
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab09$
```

Рис. 2.17: Запуск программы lab9-4.asm

В листинге приведена программа вычисления выражения $(3 + 2) * 4 + 5$. При запуске данная программа дает неверный результат. Проверил это, анализируя изменения значений регистров с помощью отладчика GDB.

Определил ошибку - перепутан порядок аргументов у инструкции `add`. Также обнаружил, что по окончании работы в `edi` отправляется `ebx` вместо `eax`. (рис. [2.18])



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-  
/home/adbelov~ab9-5.asm [----] 0 L:[ 1+14 15/ 21]  
%include 'in_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL _start  
_start:  
; ---- Вычисление выражения (3+2)*4+5  
mov ebx,3  
mov eax,2  
add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рис. 2.18: Код с ошибкой


```

adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
eax      0x804a000      134520832      ecx      0x4          4
edx      0x0           0              ebx      0xa         10
esp      0xffffd1e0    0xffffd1e0    ebp      0x0         0x0
esi      0x0           0              edi      0xa         10
eip      0x8049105     0x8049105 < _start+29>  eflags    0x206        [ PF IF ]
cs       0x23         35            ss       0x2b         43
ds       0x2b         43            es       0x2b         43
fs       0x0           0              gs       0x0         0

B+ 0x80490e8 < _start>      mov     ebx,0x3
B+ 0x80490e8 < _start>5>    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,0x5
0x80490fb < _start>19>     add     ebx,0x5
0x80490fe < _start>22>     mov     edi,ebx04a000
>0x8049100 < _start>24>     mov     eax,0x804a000<rint>
0x8049105 < _start>29>     call    0x804900f <sprint>
0x804910a < _start>34>     mov     eax,edi86 <iprintLF>
0x804910c < _start>36>     call    0x8049086 <iprintLF>

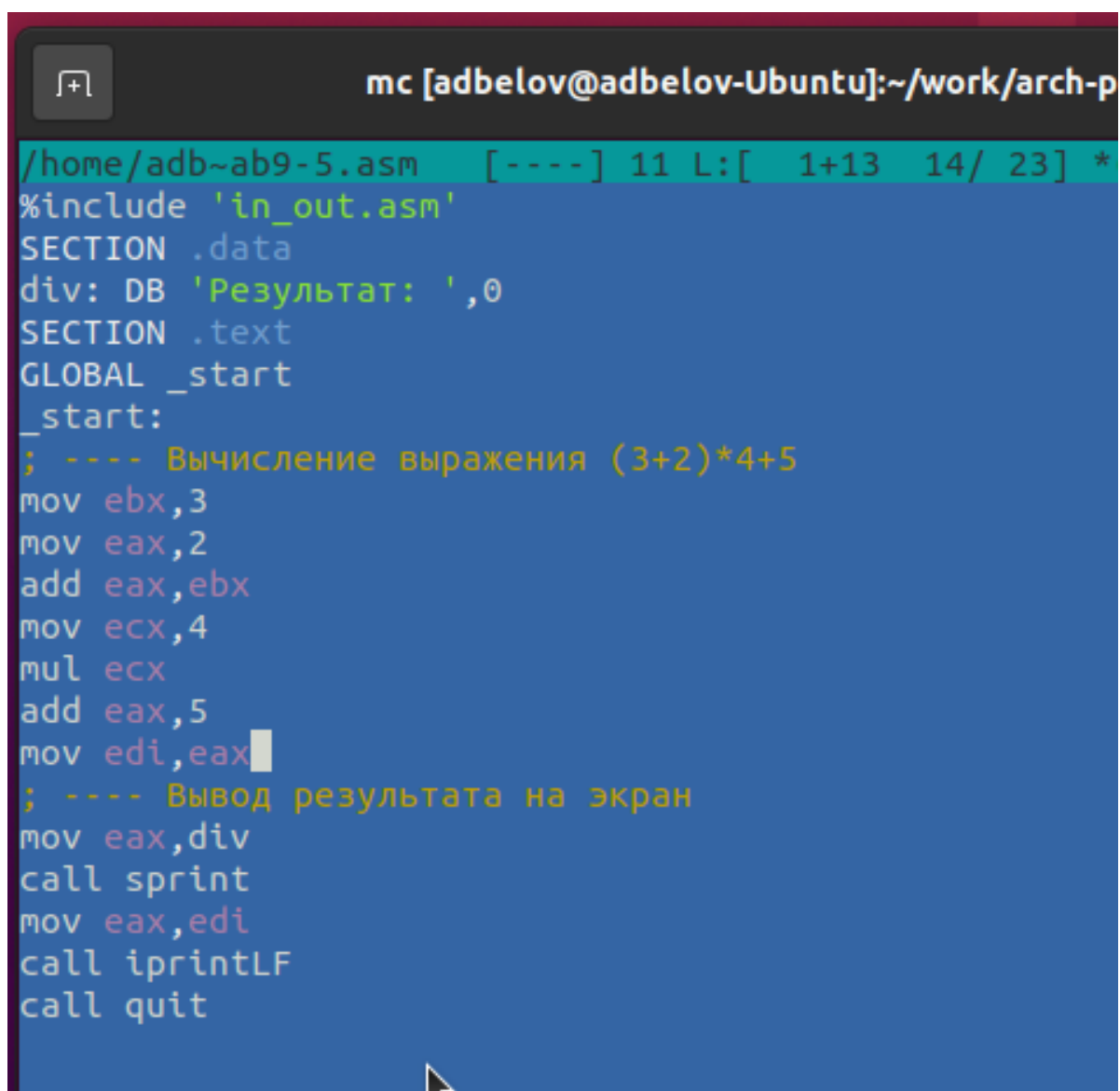
native process 6916 In: _start      L??  PC: 0x8049105
0x08049105 No process In:          L??  PC: ??
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) si
0x08049100 in _start ()
(gdb) si
0x08049105 in _start ()
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 6916) exited normally]
(gdb)

```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. [2.19])

Исправленный код программы (рис. [2.20]) (рис. [2.21])



The image shows a terminal window with a dark background. At the top, a title bar indicates the user is 'mc [adbelov@adbelov-Ubuntu]:~/work/arch-p'. Below this, a status bar shows the file path '/home/adbelov~ab9-5.asm', line numbers '[---] 11 L:[1+13 14/ 23]', and a '*' symbol. The main area contains assembly code for a program that calculates (3+2)*4+5 and prints the result. The code is color-coded: comments are yellow, section names are green, and instructions are white. The code includes a data section for a string 'Результат: ', a text section for the main logic, and a global _start function. The logic involves moving 3 to ebx, 2 to eax, adding ebx to eax, multiplying by 4 in ecx, adding 5 to eax, and storing the result in edi. Finally, it prints the string and the result using sprintf and iprintLF, then calls quit.

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-p
/home/adbelov~ab9-5.asm [ --- ] 11 L:[ 1+13 14/ 23 ] *
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprintf
mov eax,edi
call iprintLF
call quit
```

Рис. 2.20: Код исправлен

```

adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab09
eax 0x804a000 134520832 ecx 0x4 4
edx 0x0 0 ebx 0x3 3
esp 0xffffd1e0 0xffffd1e0 ebp 0x0 0x0
esi 0x0 0 edi 0x19 25
eip 0x8049105 0x8049105 < start+29> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

B+ 0x80490e8 <_start> mov ebx,0x3
B+ 0x80490e8 <_start>5> mov ebx,0x3
0x80490ed <_start>5> mov eax,0x2
0x80490f2 <_start>10> add eax,ebx
0x80490f4 <_start>12> mov ecx,0x4
0x80490f9 <_start>17> mul ecx,0x5
0x80490fb <_start>19> add eax,0x5
0x80490fe <_start>22> mov edi,eax04a000
>0x8049100 <_start>24> mov eax,0x804a000frint>
0x8049105 <_start>29> call 0x804900f <sprint>
0x804910a <_start>34> mov eax,edi86 <iprintLF>
0x804910c <_start>36> call 0x8049086 <iprintLF>

native process 6952 In: _start L?? PC: 0x8049105
0x08049105 No process In: L?? PC: ??
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) si
0x08049100 in _start ()
(gdb) si
0x08049105 in _start ()
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 6952) exited normally]
(gdb)

```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.