

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

Дисциплина: Архитектура компьютера

Обрезкова Анастасия Владимировна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Задания для самостоятельной работы	13
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создание, переход в lab07	8
4.2	Результат программы	8
4.3	Изменения текста	9
4.4	Результат изменений	9
4.5	Изменила программу	10
4.6	Вывела результат	10
4.7	Создание файла	11
4.8	Ввела нужный текст	11
4.9	Вывод результата	12
4.10	Создание файла	12
4.11	Открытие файла	12
4.12	Открытие файла после удаления	13
4.13	Добавления в листинге	13
4.14	Текст программы	14
4.15	Результат	14
4.16	Текст программы	15
4.17	Текст программы	16
4.18	Результат	17

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

Изученить команды переходов. Приобрести навыки написания программ с использованием переходов. Ознакомится с назначением и структурой файла листинга.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Описание инструкции `cmp`

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения.

Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания:

`cmp ,`

Команда `cmp`, так же как и команда вычитания, выполняет вычитание -, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

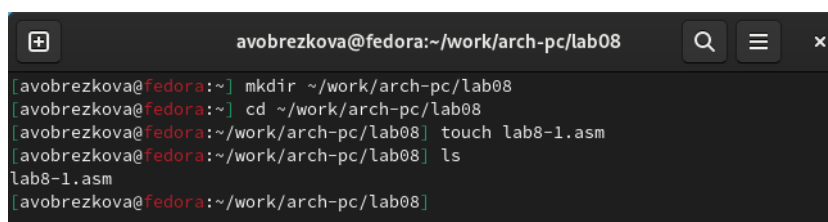
Команда условного перехода имеет вид

`j label`

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

4 Выполнение лабораторной работы

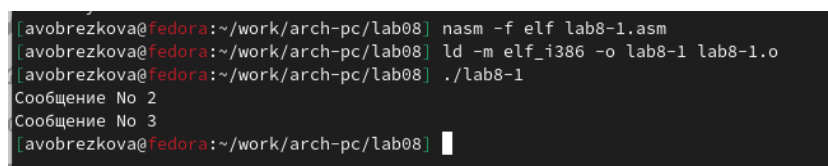
1. Создала каталог для программ лабораторной работы №8, перешла в него и создала файл lab8-1.asm. (рис. 4.1)



```
avobrezkova@fedora:~/work/arch-pc/lab08
avobrezkova@fedora:~$ mkdir ~/work/arch-pc/lab08
avobrezkova@fedora:~$ cd ~/work/arch-pc/lab08
avobrezkova@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
avobrezkova@fedora:~/work/arch-pc/lab08$ ls
lab8-1.asm
avobrezkova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание, переход в lab07

2. Ввела в файл lab8-1 нужный текст программы из листинга 8.1., создала исполняемый файл и вывела результат. (рис. 4.2)



```
avobrezkova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
avobrezkova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
avobrezkova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Сообщение No 2
Сообщение No 3
avobrezkova@fedora:~/work/arch-pc/lab08$
```

Рис. 4.2: Результат программы

3. Изменила текст программы в соответствии с листингом 8.2. (рис. 4.3; рис. 4.4)



```
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label2

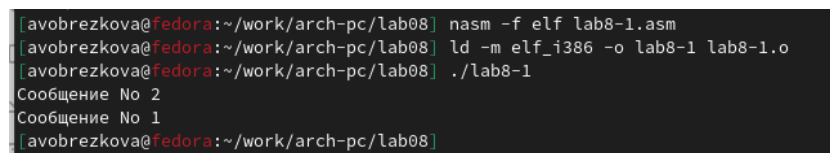
_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

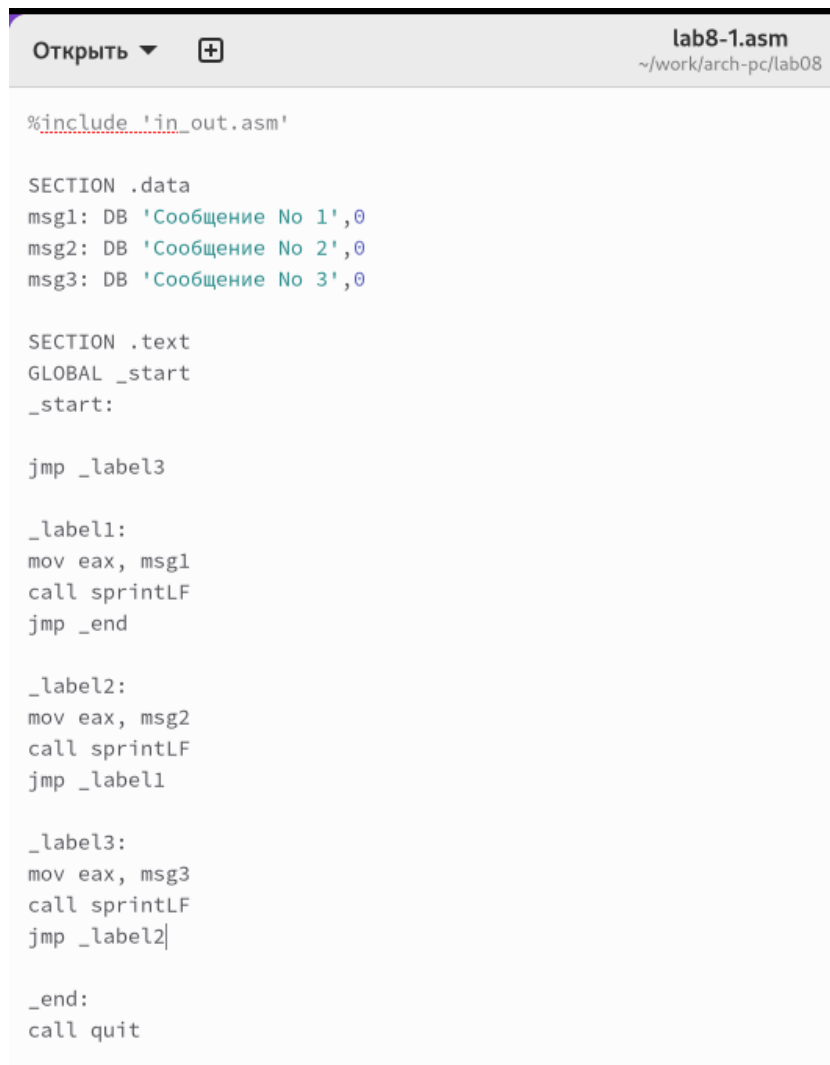
Рис. 4.3: Изменения текста



```
[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf lab8-1.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o lab8-1 lab8-1.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./lab8-1
Сообщение No 2
Сообщение No 1
[avobrezkova@fedora:~/work/arch-pc/lab08]
```

Рис. 4.4: Результат изменений

4. Изменила текст программы, чтобы программа выводила определенный результат. (рис. 4.6; рис. 4.7)



```
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

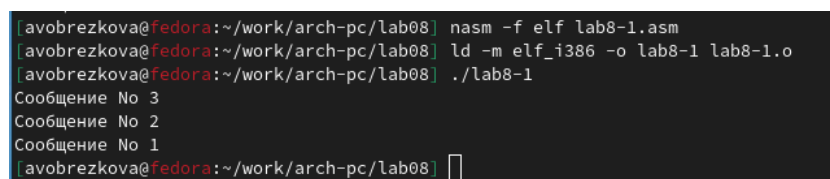
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 4.5: Изменила программу



```
[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf lab8-1.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o lab8-1 lab8-1.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[avobrezkova@fedora:~/work/arch-pc/lab08] □
```

Рис. 4.6: Вывела результат

5. Создала файл lab8-2.asm в нужном каталоге, ввела нужный текст и вывела результат. (рис. 4.7; рис. 4.8; рис. 4.9)

```

[avobrezkova@fedora:~/work/arch-pc/lab08] touch lab8-2.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
[avobrezkova@fedora:~/work/arch-pc/lab08]

```

Рис. 4.7: Создание файла

Открыть ▾

+

lab8-2.asm
~/work/arch-pc/lab08

```

#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10

section .text

global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

check_B:
mov eax,max
call atoi
mov [max],eax

mov ecx,[max]

```

Рис. 4.8: Ввела нужный текст

```

[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf lab8-2.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o lab8-2 lab8-2.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./lab8-2
Введите B: 30
Наибольшее число: 50
[avobrezkova@fedora:~/work/arch-pc/lab08] ./lab8-2
Введите B: 12
Наибольшее число: 50
[avobrezkova@fedora:~/work/arch-pc/lab08] ./lab8-2
Введите B: 60
Наибольшее число: 60
[avobrezkova@fedora:~/work/arch-pc/lab08]

```

Рис. 4.9: Вывод результата

6. Создала файл листинга для программы из файла lab8-2.asm, открыла созданный файл с помощью команды, ознакомилась с его форматом и содержанием. (рис. 4.10; рис. 4.11)

```

[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf -l lab8-2.lst lab8-2.asm

```

Рис. 4.10: Создание файла

```

lab8-2.lst  [----]  0 L: [ 1+ 0 1/227] *(0 /13361b) 0032 0x020 [*][X]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:.....
5          00000000 53          <1>      push    ebx.....
6          00000001 89C3       <1>      mov     ebx, eax.....
7          <1>.....
8          <1> nextchar:.....
9          00000003 803800     <1>      cmp     byte [eax], 0...
10         00000006 7403       <1>      jz      finished.....
11         00000008 40        <1>      inc     eax.....
12         00000009 EBF8       <1>      jmp     nextchar.....
13         <1>.....
14         <1> finished:
15         0000000B 29D8       <1>      sub     eax, ebx
16         0000000D 5B        <1>      pop     ebx.....
17         0000000E C3        <1>      ret.....
18         <1>..
19         <1>..
20         <1> ;----- sprint -----
21         <1> ; Функция печати сообщения
22         <1> ; входные данные: mov eax,<message>

```

Рис. 4.11: Открытие файла

7. Открыла файл с программой lab8-2.asm в инструкции с двумя операндами и удалила один операнд, выполнила трансляцию полученного файла листинга. (рис. 4.12; рис. 4.13)

```

[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:19: error: invalid combination of opcode and operands
[avobrezkova@fedora:~/work/arch-pc/lab08]

```

Рис. 4.12: Открытие файла после удаления

```

lab8-2.lst  [--O]  0 L:[187+ 0 187/228] *(11379/13450b) 0032 0x020[*][X]
12.....
13.....          global _start
14.....          _start:
15.....
16 000000E8 B8[00000000]      mov eax,msg1
17 000000ED E81DFFFFFF      call sprint
18.....
19.....          mov ecx
19.....          *****      error: invalid combination of opcode an
20 000000F2 BA0A000000      mov edx,10
21 000000F7 E847FFFFFF      call sread
22.....
23 000000FC B8[0A000000]      mov eax,B
24 00000101 E896FFFFFF      call atoi.
25 00000106 A3[0A000000]      mov [B],eax.
26.....
27 0000010B 8B0D[35000000]      mov ecx,[A]
28 00000111 890D[00000000]      mov [max],ecx.
29.....
30 00000117 3B0D[39000000]      cmp ecx,[C].
31 0000011D 7F0C          jg check_B.
32 0000011F 8B0D[39000000]      mov ecx,[C].

```

Рис. 4.13: Добавления в листинге

Никаких выходных файлов не создается, так как постоянно возникает ошибка. Ошибка возникает из-за того, что в программе всегда должно быть два операнда, а так как мы удаляем один операнд, программа не работает. В листинге добавляется строка с надписью error.

4.1 Задания для самостоятельной работы

1. Я написала программу, которая находит наименьшее из 3 целочисленных переменных. (рис. 4.14; рис. 4.15)

```

Открыть ▾ + • sr1.asm
~/work/arch-pc/lab08

%include 'in_out.asm'
section .data
msg1 db 'Введите B', 0h
msg2 db 'Наименьшее число', 0h
A dd '79'
C dd '41'

section .bss
min resb 10
B resb 10
section .text

global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jnb check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'

```

Рис. 4.14: Текст программы

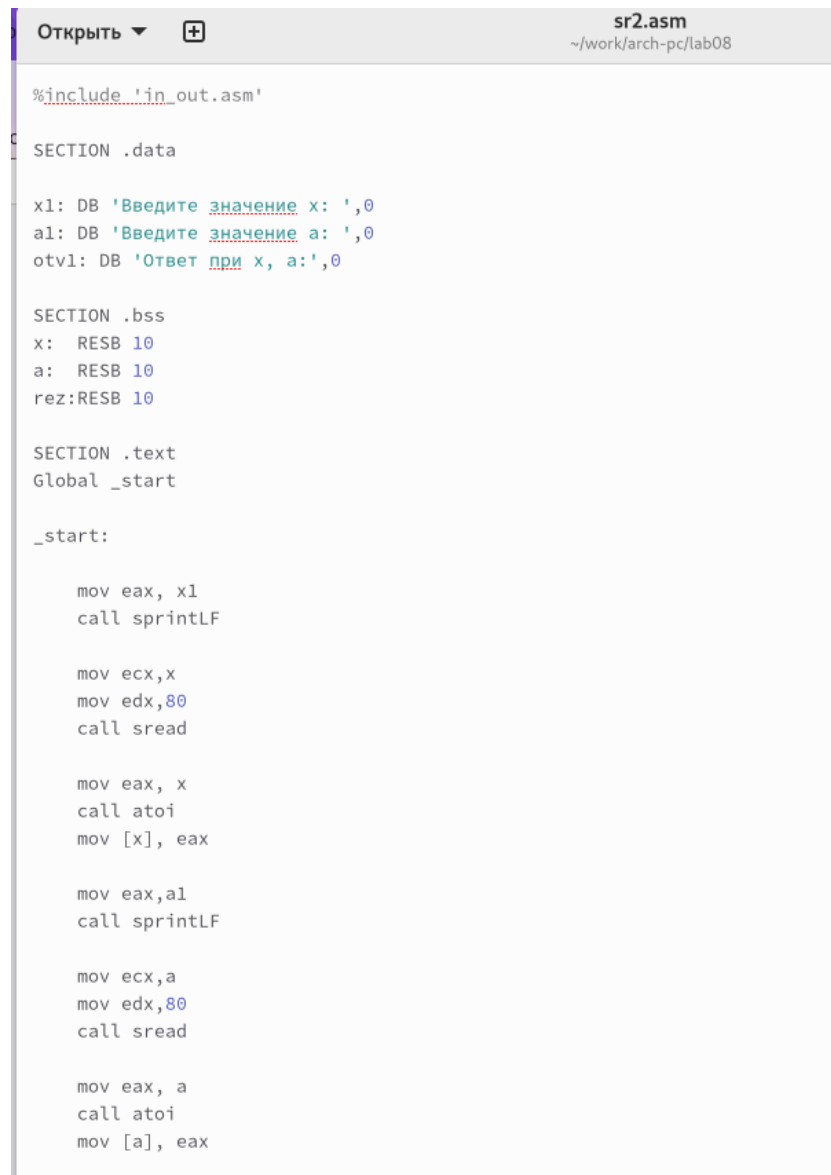
```

avobrezkova@fedora:~/work/arch-pc/lab08
[avobrezkova@fedora:~] nasm -f elf sr1.asm
nasm: fatal: unable to open input file 'sr1.asm' No such file or directory
[avobrezkova@fedora:~] cd ~/work/arch-pc/lab08
[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf sr1.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o sr1 sr1.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./sr1
Введите B83
Наименьшее число41
[avobrezkova@fedora:~/work/arch-pc/lab08]

```

Рис. 4.15: Результат

2. Я написала программу, которая для введенных с клавиатуры значений вычислит значение заданной функции №6. (рис. 4.16; рис. 4.17; рис. 4.18)



```
Открыть ▾ + sr2.asm
~/.work/arch-pc/lab08

%include 'in_out.asm'

SECTION .data

x1: DB 'Введите значение x: ',0
a1: DB 'Введите значение a: ',0
otv1: DB 'Ответ при x, a:',0

SECTION .bss
x: RESB 10
a: RESB 10
rez: RESB 10

SECTION .text
Global _start

_start:

    mov eax, x1
    call sprintLF

    mov ecx, x
    mov edx, 80
    call sread


    mov eax, x
    call atoi
    mov [x], eax

    mov eax, a1
    call sprintLF

    mov ecx, a
    mov edx, 80
    call sread

    mov eax, a
    call atoi
    mov [a], eax
```

Рис. 4.16: Текст программы

Открыть ▾ 

sr2.asm
~/work/arch-pc/lab08

```
mov eax, x1
call sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi
mov [x], eax

mov eax, a1
call sprintLF

mov ecx, a
mov edx, 80
call sread

mov eax, a
call atoi
mov [a], eax

mov ebx, [x]
cmp ebx, [a] ; compare
jg konets

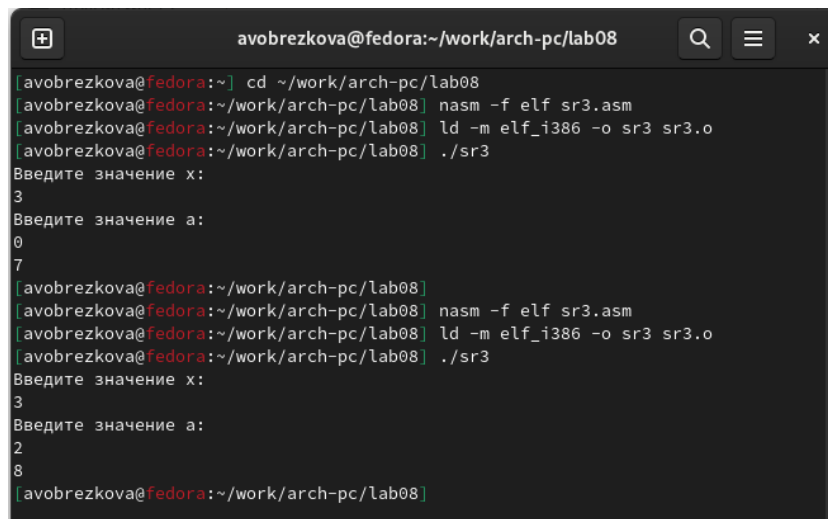
mov eax, [x]
add eax, [a]

jmp end

konets:
mov eax, [x]
mov ebx, 5
mul ebx

end:
call iprintLF
call quit
```

Рис. 4.17: Текст программы



```
avobrezkova@fedora:~/work/arch-pc/lab08
[avobrezkova@fedora:~] cd ~/work/arch-pc/lab08
[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf sr3.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o sr3 sr3.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./sr3
Введите значение x:
3
Введите значение a:
0
7
[avobrezkova@fedora:~/work/arch-pc/lab08]
[avobrezkova@fedora:~/work/arch-pc/lab08] nasm -f elf sr3.asm
[avobrezkova@fedora:~/work/arch-pc/lab08] ld -m elf_i386 -o sr3 sr3.o
[avobrezkova@fedora:~/work/arch-pc/lab08] ./sr3
Введите значение x:
3
Введите значение a:
2
8
[avobrezkova@fedora:~/work/arch-pc/lab08]
```

Рис. 4.18: Результат

Данные изменения можно проверить по ссылке: https://github.com/avobrezkova/study_2022-2023_arh-pc/tree/master/labs/lab08

5 Выводы

Изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга..

Список литературы

1. https://esystem.rudn.ru/pluginfile.php/1584392/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%968.pdf