

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Лобанов Владислав Олегович

Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Выполнение лабораторной работы	8
Реализация переходов в NASM	8
Изучение структуры файла листинга	13
Выполнение заданий для самостоятельной работы (вар. 20)	15
Выводы	19
Список литературы	20

Список иллюстраций

1	Создание каталога и файла в ней	8
2	Редактирование файла	9
3	Исполнение программы	9
4	Редактирование файла	10
5	Исполнение программы	10
6	Редактирование файла	11
7	Исполнение программы	11
8	Создание файла	12
9	Редактирование файла	12
10	Исполнение программы для разных значений В	13
11	Название рисунка	14
12	Редактирование файла	14
13	Открытие файла листинга	15
14	Исполнение программы	16
15	Исполнение программы	18

Список таблиц

Цель работы

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

Задание

1. Реализация переходов в NASM
2. Изучение структуры файла листинга
3. Задание для самостоятельной работы

Теоретическое введение

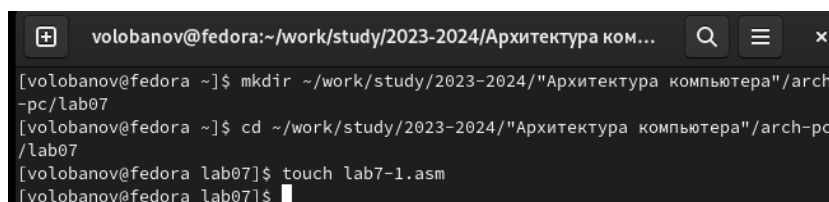
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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.

Выполнение лабораторной работы

Реализация переходов в NASM

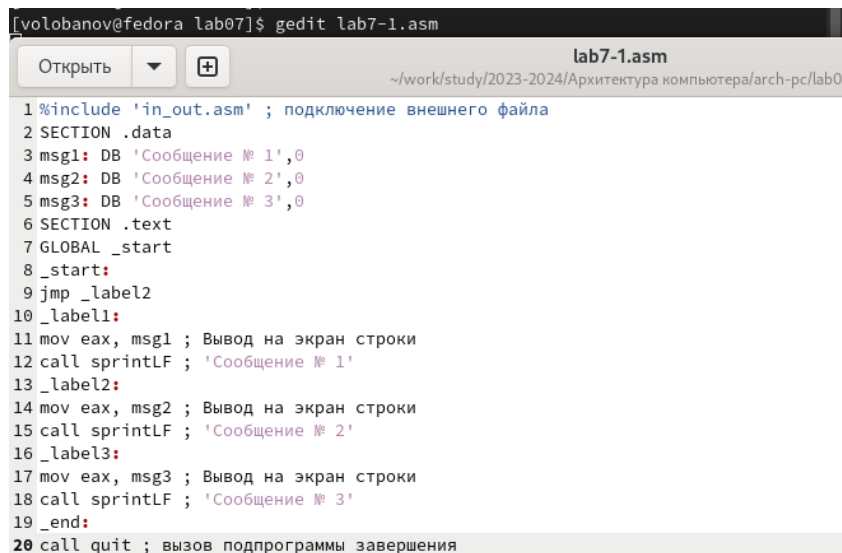
Создаю каталог для программ для лабораторной работе №7, перехожу в него и создаю файл lab7-1.asm (рис. @fig:001).

A terminal window with a dark background. The title bar shows 'volobanov@fedora:~/work/study/2023-2024/Архитектура ком...'. The terminal content shows the following commands and their outputs:

```
[volobanov@fedora ~]$ mkdir ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab07
[volobanov@fedora ~]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab07
[volobanov@fedora lab07]$ touch lab7-1.asm
[volobanov@fedora lab07]$
```

Рис. 1: Создание каталога и файла в ней

Ввожу в файл lab7-1.asm текст программы с использованием функции jmp (рис. @fig:002).



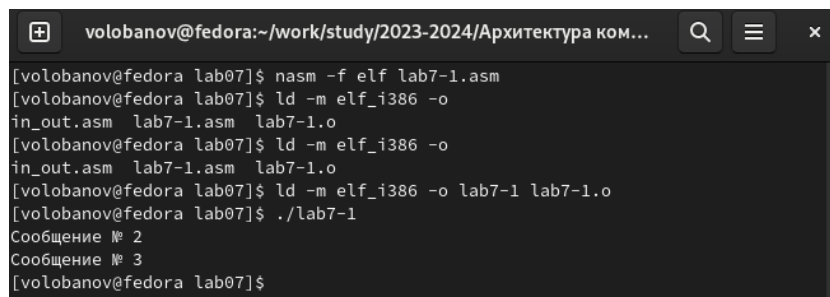
```
[volobanov@fedora lab07]$ gedit lab7-1.asm

lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 2: Редактирование файла

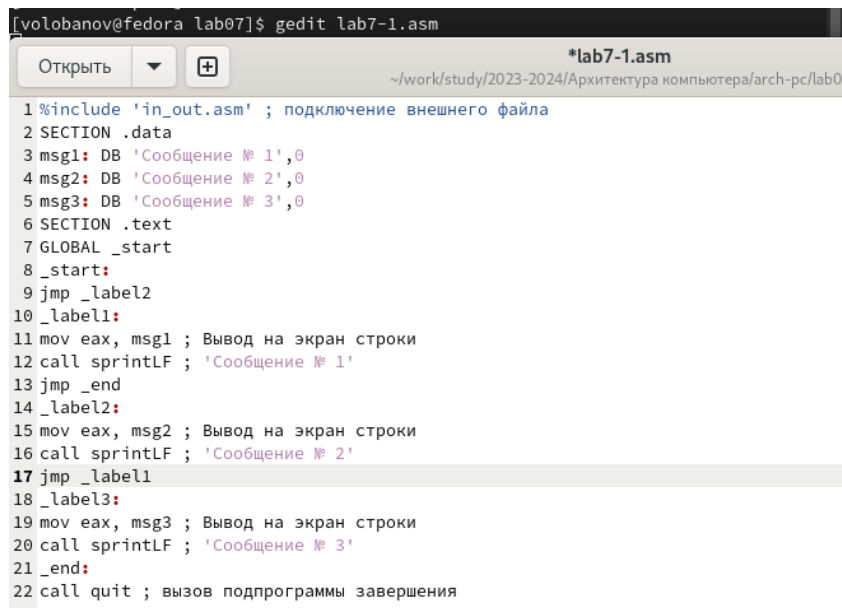
Создаю исполняемый файл и запускаю его (рис. @fig:003).



```
volobanov@fedora:~/work/study/2023-2024/Архитектура ком...
[volobanov@fedora lab07]$ nasm -f elf lab7-1.asm
[volobanov@fedora lab07]$ ld -m elf_i386 -o
in_out.asm lab7-1.asm lab7-1.o
[volobanov@fedora lab07]$ ld -m elf_i386 -o
in_out.asm lab7-1.asm lab7-1.o
[volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[volobanov@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[volobanov@fedora lab07]$
```

Рис. 3: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу (рис. @fig:004).



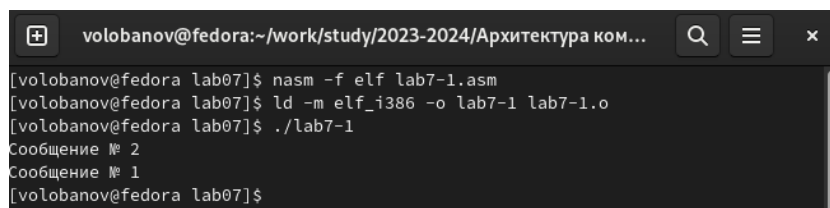
```
[volobanov@fedora lab07]$ gedit lab7-1.asm

*lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4: Редактирование файла

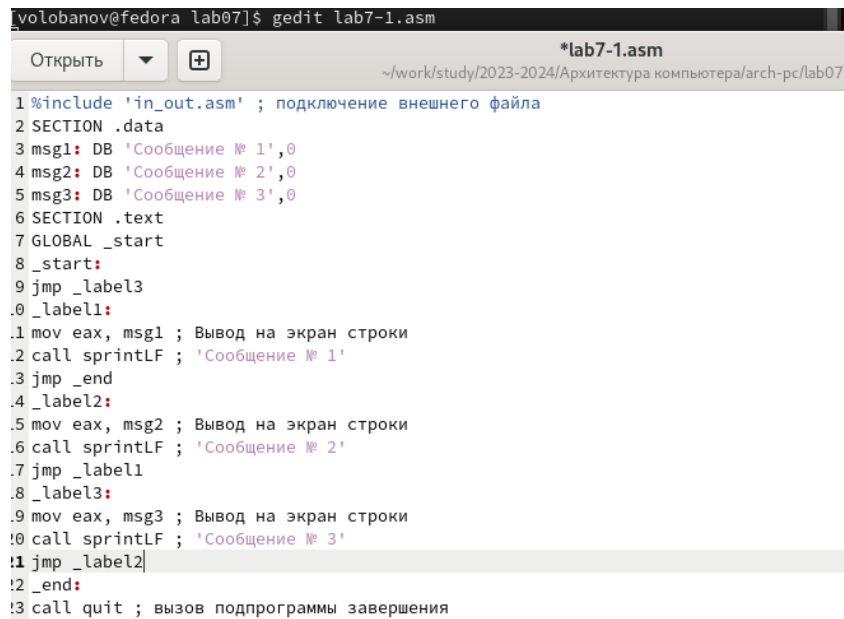
Создаю исполняемый файл и проверяю его работу (рис. @fig:005).



```
volobanov@fedora:~/work/study/2023-2024/Архитектура ком...
[volobanov@fedora lab07]$ nasm -f elf lab7-1.asm
[volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[volobanov@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[volobanov@fedora lab07]$
```

Рис. 5: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 3’, потом ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу (рис. @fig:006).



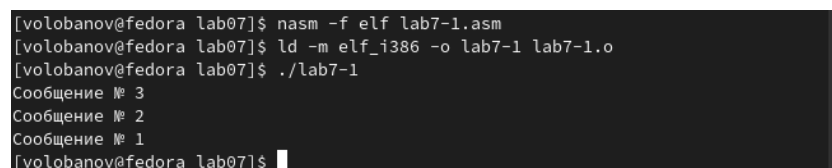
```
volobanov@fedora lab07]$ gedit lab7-1.asm

*lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 6: Редактирование файла

Создаю исполняемый файл и проверяю корректность работы программы (рис. @fig:007). Программа отработала корректно.



```
[volobanov@fedora lab07]$ nasm -f elf lab7-1.asm
[volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[volobanov@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[volobanov@fedora lab07]$
```

Рис. 7: Исполнение программы

Создаю файл lab7-2.asm (рис. @fig:008).

```
[volobanov@fedora lab07]$ gedit lab7-2.asm

lab7-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
```

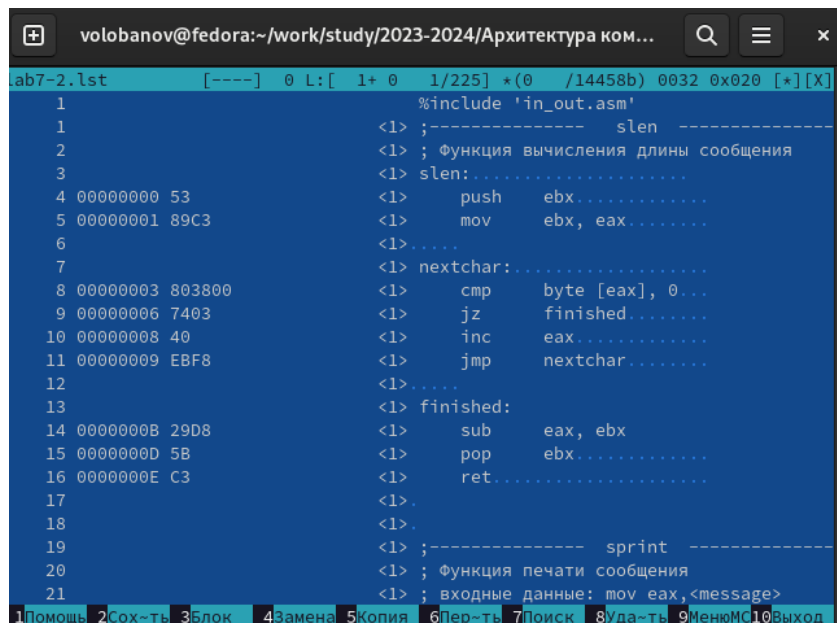
Рис. 8: Создание файла

Ввожу в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (рис. @fig:009).

```
[volobanov@fedora lab07]$ nasm -f elf lab7-2.asm
[volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[volobanov@fedora lab07]$ ./lab7-2
Введите B: 20
Наибольшее число: 50
[volobanov@fedora lab07]$ ./lab7-2
Введите B: 70
Наибольшее число: 70
[volobanov@fedora lab07]$ ./lab7-2
Введите B: 40
Наибольшее число: 50
[volobanov@fedora lab07]$
```

Рис. 9: Редактирование файла

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. @fig:010). Программа сработала корректно.



```
lab7-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*][X]
1                                     %include 'in_out.asm'
1                                     <1> ;----- slen -----
2                                     <1> ; функция вычисления длины сообщения
3                                     <1> slen:.....
4 00000000 53                         <1> push ebx.....
5 00000001 89C3                       <1> mov ebx, eax.....
6                                     <1> .....
7                                     <1> nextchar:.....
8 00000003 803800                     <1> cmp byte [eax], 0...
9 00000006 7403                       <1> jz finished.....
10 00000008 40                        <1> inc eax.....
11 00000009 EBF8                     <1> jmp nextchar.....
12                                     <1> .....
13                                     <1> finished:
14 0000000B 29D8                     <1> sub eax, ebx
15 0000000D 5B                        <1> pop ebx.....
16 0000000E C3                       <1> ret.....
17                                     <1> .
18                                     <1> .
19                                     <1> ;----- sprint -----
20                                     <1> ; функция печати сообщения
21                                     <1> ; входные данные: mov eax,<message>
```

Рис. 10: Исполнение программы для разных значений В

Изучение структуры файла листинга

Создание файла листинга и его просмотр в текстовом редакторе gedit (рис. @fig:011).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'

```

Рис. 11: Название рисунка

1. В строке 5 содержится собственно номер строки[5], адрес[00000001], машинный код[89C3] и содержимое строки кода[`mov ebx, eax`].
2. В строке 11 содержится собственно номер строки[11], адрес[00000009], машинный код[EBF8] и содержимое строки кода[`jmp nextchar`].
3. В строке 14 содержится собственно номер строки[14], адрес[0000000B], машинный код[29D8] и содержимое строки кода[`sub eax, ebx`].

Открываю файл lab7-2.asm и удаляю в инструкции `mov` второй операнд (рис. @fig:012).

```

volobanov@fedora lab07]$ gedit lab7-2.asm
volobanov@fedora lab07]$
volobanov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands

```

Рис. 12: Редактирование файла

Открытие файла листинга после трансляции (рис. @fig:013). Если в коде появляется ошибка, то её описание появится в файле листинга.

```

volobanov@fedora lab07]$ nasm -f elf lab7-3.asm
volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
volobanov@fedora lab07]$ ./lab7-3
Наибольшее число: 87
volobanov@fedora lab07]$

```

Рис. 13: Открытие файла листинга

Выполнение заданий для самостоятельной работы (вар. 20)

Создаю файл lab7-3.asm, пишу в нём программу для нахождения наименьшей из трёх целочисленных переменных a, b и c.

Текст программы в файле lab7-3.asm:

```
%include 'in_out.asm' section .data msg2 db "Наибольшее число:",0h A dd '54' B
dd '62' C dd '87'
```

```
section .bss max resb 10
```

```
section .text
```

```
global _start _start: mov eax,B call atoi ; Вызов подпрограммы перевода символа
в число mov [B],eax ; запись преобразованного числа в 'B' ; ——— Записываем
'A' в переменную 'max' mov ecx,[A] ; 'ecx = A' mov [max],ecx ; 'max = A' ; ———
Сравниваем 'A' и 'C' (как символы) cmp ecx,[C] ; Сравниваем 'A' и 'C' jg check_B ;
если 'A>C', то переход на метку 'check_B', mov ecx,[C] ; иначе 'ecx = C' mov [max],ecx
; 'max = C'
```

```
; ——— Преобразование 'max(A,C)' из символа в число check_B: mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число mov [max],eax ; запись
преобразованного числа в max ; ——— Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B' jg fin ; если 'max(A,C)>B',
то переход на 'fin', mov ecx,[B] ; иначе 'ecx = B'
```

```
mov [max],ecx
```

```
; ——— Вывод результата fin: mov eax, msg2 call sprint ; Вывод сообщения
'Наибольшее число:' mov eax,[max] call iprintLF ; Вывод 'max(A,B,C)' call quit ;
Выход
```

Создаю исполняемый файл и проверяю его работу (рис. @fig:014). Программа отработала корректно.

```
volobanov@fedora lab07]$ nasm -f elf lab7-4.asm
volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
volobanov@fedora lab07]$ ./lab7-4
x = 2
a = 1

volobanov@fedora lab07]$ ./lab7-4
x = 3
a = 2

volobanov@fedora lab07]$
```

Рис. 14: Исполнение программы

Создаю файл lab7-4.asm, пишу в нём программу, которая для введённых с клавиатуры значений x и a вычисляет значение функции $f(x)$, которая равна $x-a$ при $x \geq a$, и 5, когда $x < a$ и выводит результат вычислений.

Текст программы в файле lab7-4.asm:

```
%include 'in_out.asm'
```

```
section .data
```

```
msgX db "x = ",0h
```

```
msgA db "a = ",0h
```

```
section .bss
```

```
x resb 10
```

```
a resb 10
```

```
f resb 10
```

```
section .text
```

```
global _start
```

```
_start:
```



```

; ----- Ввод 'X'
mov eax, msgX
call sprint
mov ecx, x
mov edx, 10
call sread

; ----- Ввод 'A'
mov eax, msgA
call sprint
mov ecx, a
mov edx, 10
call sread

; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi
mov [x], eax

; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi
mov [a], eax

mov ecx, [x]
cmp ecx, [a]

jge func

```

```
mov eax, 5
jmp fin
```

func:

```
mov eax, [x]
mov ecx, [a]
sub eax, ecx
```

fin:

```
call iprintLF
call quit
```

Создаю исполняемый файл и проверяю его работу для пар x и a (1,2) и (2,1) (рис. @fig:015). Программа отработала верно.

```
volobanov@fedora lab07]$ nasm -f elf lab7-4.asm
volobanov@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
volobanov@fedora lab07]$ ./lab7-4
x = 2
a = 1

volobanov@fedora lab07]$ ./lab7-4
x = 3
a = 2

volobanov@fedora lab07]$
```

Рис. 15: Исполнение программы

Выводы

В ходе выполнения лабораторной работы я освоил принципы условного и безусловного перехода в NASM.

Список литературы

1. Лабораторная работа №6