

Очёт по лабораторной работе № 8

Архитектура Компьютера

Миразим Азимов

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	12
3.3	Задание для самостоятельной работы	14
4	Выводы	17

Список иллюстраций

3.1	Название рисунка	6
3.2	Название рисунка	7
3.3	Название рисунка	7
3.4	Название рисунка	8
3.5	Название рисунка	9
3.6	Название рисунка	9
3.7	Название рисунка	10
3.8	Название рисунка	10
3.9	Название рисунка	11
3.10	Название рисунка	11
3.11	Название рисунка	11
3.12	Название рисунка	12
3.13	Название рисунка	12
3.14	Название рисунка	13
3.15	Название рисунка	13
3.16	Название рисунка	14
3.17	Название рисунка	14
3.18	Название рисунка	15
3.19	Название рисунка	15
3.20	Название рисунка	16
3.21	Название рисунка	16

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыков написания программ с использованием переходов. Ознакомиться с назначением и структурой файла листинга.

2 Задание

1. Реализовать переходы в NASM
2. Изучить структуру файлов листинга
3. Выполнить задание для самостоятельной работы

3 Выполнение лабораторной работы

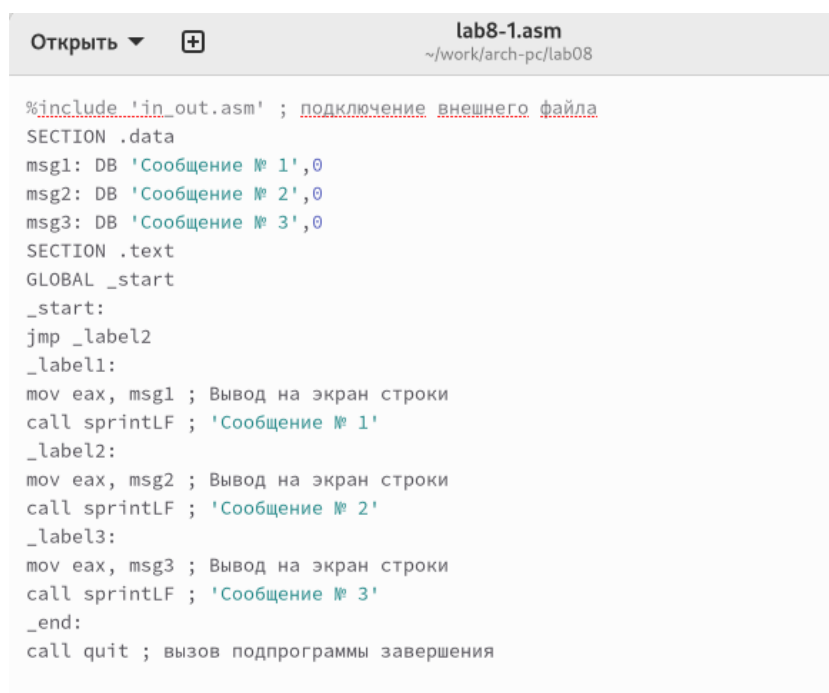
3.1 Реализация переходов в NASM

1. Создали каталог для программ лабораторной работы № 8, перешли в него и создали файл lab8-1.asm: (рис. 3.1)

```
[mazimov@fedora lab07]$ mkdir ~/work/arch-pc/lab08  
[mazimov@fedora lab07]$ cd ~/work/arch-pc/lab08  
[mazimov@fedora lab08]$ touch lab8-1.asm  
[mazimov@fedora lab08]$
```

Рис. 3.1: Название рисунка

2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрели пример программы с использованием инструкции jmp. Ввели в файл lab8-1.asm текст программы из листинга 8.1. (рис. 3.2)



```

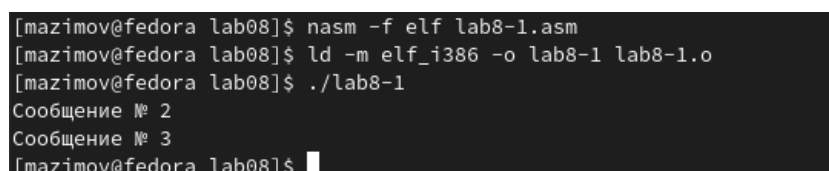
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.2: Название рисунка

Создали исполняемый файл и запустили его. Результат работы данной программы следующий: (рис. 3.3)



```

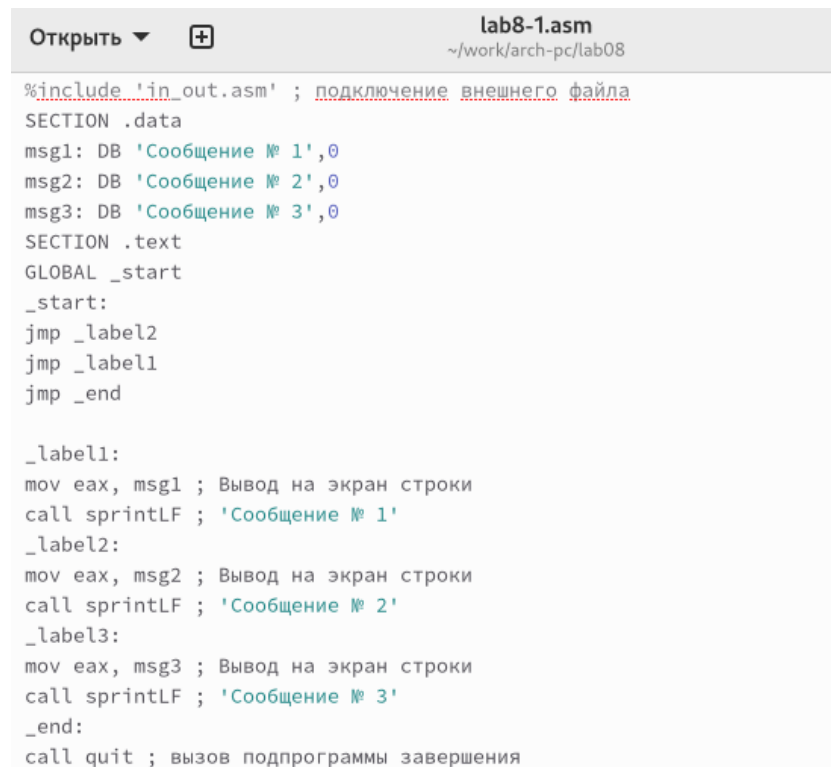
[mazimov@fedora lab08]$ nasm -f elf lab8-1.asm
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[mazimov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[mazimov@fedora lab08]$


```

Рис. 3.3: Название рисунка

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменили программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавили инструкцию `jmp` с меткой `_end`

(т.е. переход к инструкции call quit). Изменили текст программы в соответствии с листингом 8.2 (рис. 3.4), (рис. 3.5), (рис. 3.6)

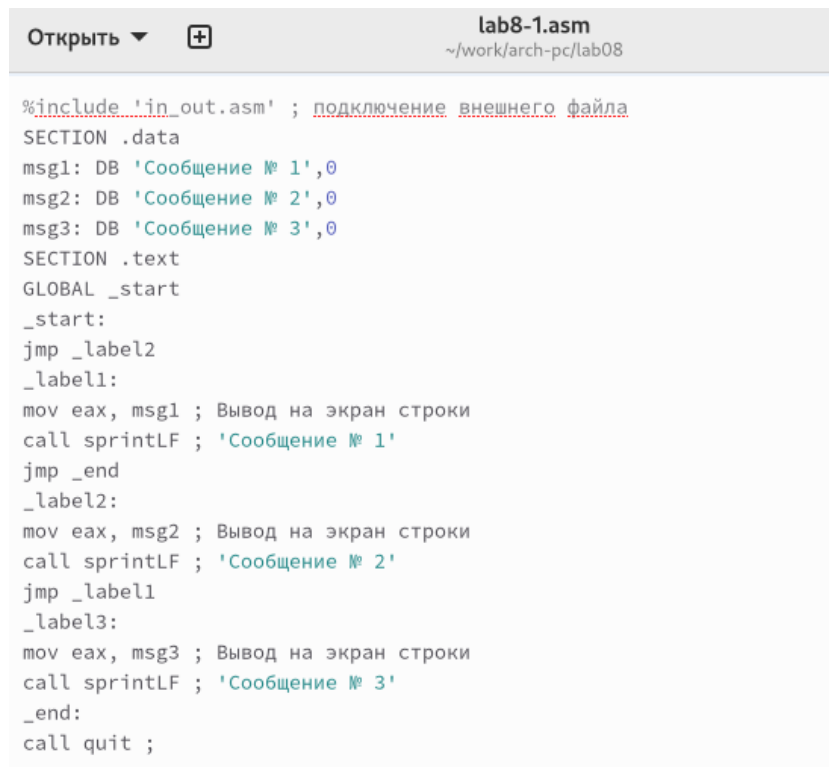


```
Открыть ▾  lab8-1.asm
~/work/arch-pc/lab08

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
jmp _label1
jmp _end

_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Название рисунка




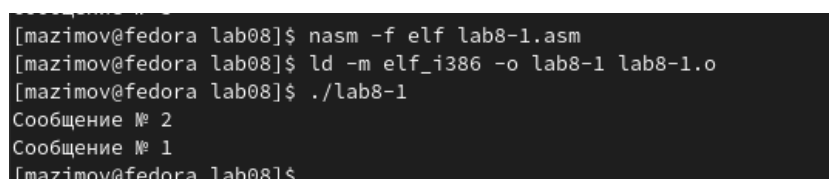
```
Открыть ▾  lab8-1.asm  
~/work/arch-pc/lab08  
  
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
_start:  
jmp _label2  
_label1:  
mov eax, msg1 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 1'  
jmp _end  
_label2:  
mov eax, msg2 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 2'  
jmp _label1  
_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 3'  
_end:  
call quit ;
```

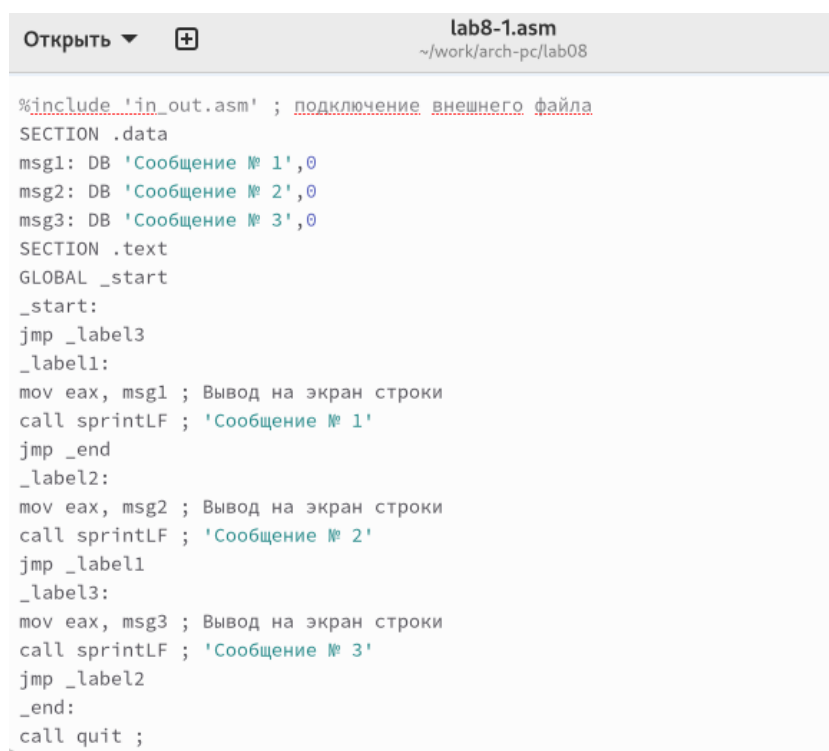
Рис. 3.5: Название рисунка



```
[mazimov@fedora lab08]$ nasm -f elf lab8-1.asm  
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[mazimov@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 1  
[mazimov@fedora lab08]$
```

Рис. 3.6: Название рисунка

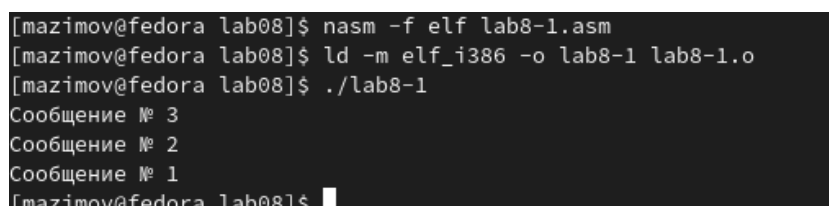
Измените текст программы добавив или изменив инструкции `jmp`. (рис. 3.7),
(рис. 3.8)



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

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ;
```

Рис. 3.7: Название рисунка



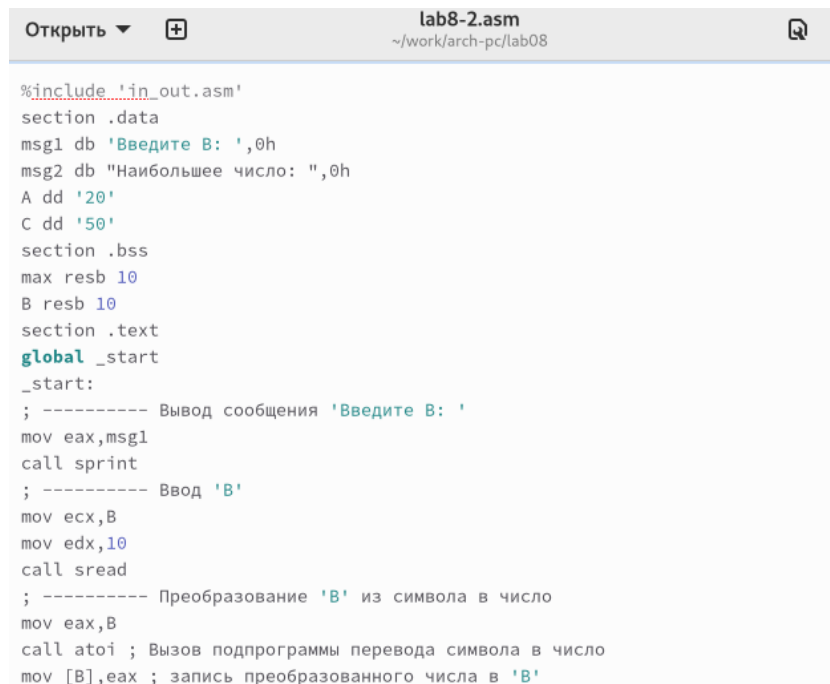
```
[mazimov@fedora lab08]$ nasm -f elf lab8-1.asm
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[mazimov@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[mazimov@fedora lab08]$
```

Рис. 3.8: Название рисунка

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создали файл `lab8-2.asm` в каталоге `~/work/arch-pc/lab08`. (рис. 3.9) Внимательно изучили текст программы из листинга 8.3 и ввели в `lab8-2.asm`. (рис. 3.10)

```
[mazimov@fedora lab08]$ touch lab8-2.asm
```

Рис. 3.9: Название рисунка



```
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:
; ----- 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'
```

Рис. 3.10: Название рисунка

Создали исполняемый файл и проверили его работу для разных значений B.
(рис. 3.11)

```
[mazimov@fedora lab08]$ nasm -f elf lab8-2.asm
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[mazimov@fedora lab08]$ ./lab8-2
Введите B: 3
Наибольшее число: 50
[mazimov@fedora lab08]$ ./lab8-2
Введите B: 60
Наибольшее число: 60
[mazimov@fedora lab08]$
```


Рис. 3.11: Название рисунка

Обратили внимание, в данном примере переменные A и C сравниваются как символы, а переменная B и максимум из A и C как числа (для этого используется функция `atoi` преобразования символа в число). Это сделано для демонстрации

того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию `atoi`). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

3.2 Изучение структуры файлы листинга

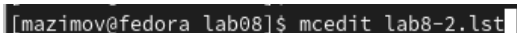
4. Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке. Создали файл листинга для программы из файла `lab8-2.asm`. (рис. 3.12)



```
[mazimov@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

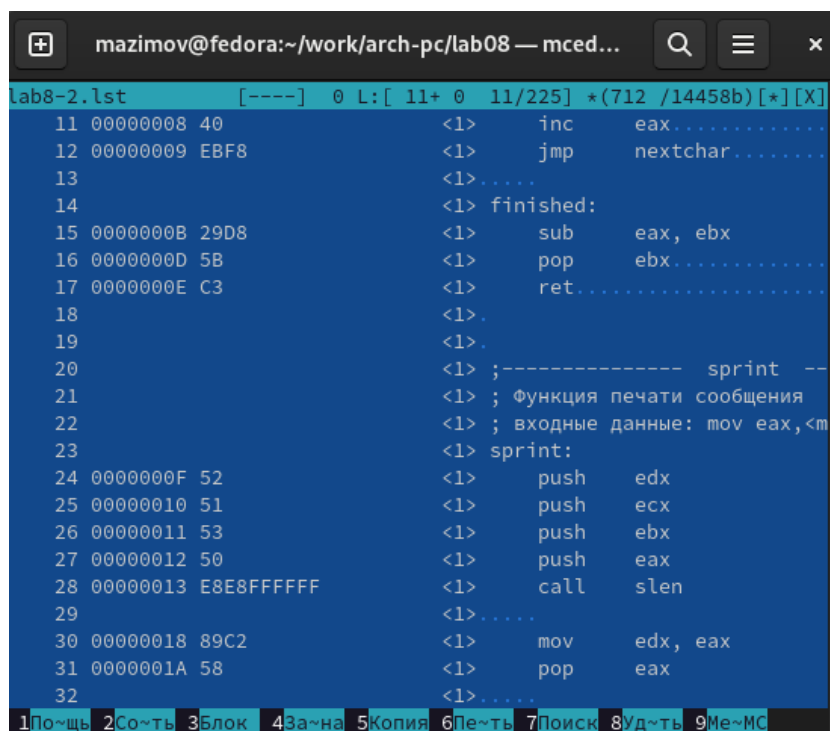
Рис. 3.12: Название рисунка

Открыли файл листинга `lab8-2.lst` с помощью текстового редактора `mcedit`: (рис. 3.13), (рис. 3.14)



```
[mazimov@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.13: Название рисунка

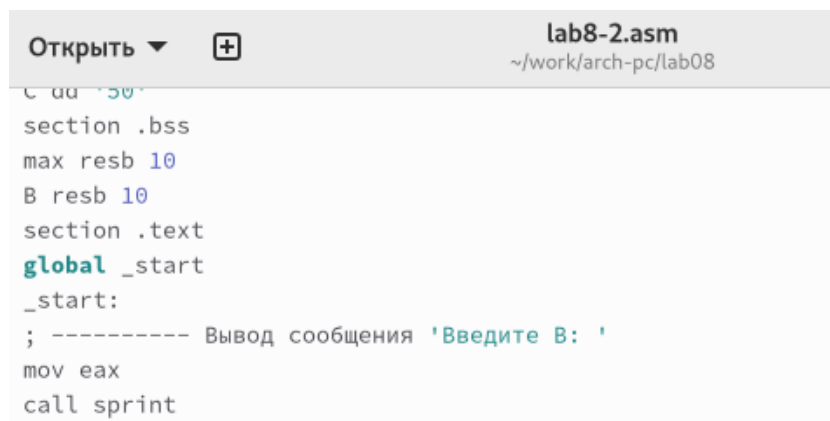


```
lab8-2.lst      [----]  0 L: [ 11+ 0  11/225] *(712 /14458b) [*] [X]
11 00000008 40          <1>    inc     eax.....
12 00000009 EBF8       <1>    jmp     nextchar.....
13                      <1>.....
14                      <1> finished:
15 0000000B 29D8       <1>    sub     eax, ebx
16 0000000D 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,<m
23                      <1> sprint:
24 0000000F 52          <1>    push    edx
25 00000010 51          <1>    push    ecx
26 00000011 53          <1>    push    ebx
27 00000012 50          <1>    push    eax
28 00000013 E8E8FFFFFF <1>    call    slen
29                      <1>.....
30 00000018 89C2       <1>    mov     edx, eax
31 0000001A 58          <1>    pop     eax
32                      <1>.....
1По~щ  2Со~ть  3Блок  4За~на  5Копия  6Пе~ть  7Поиск  8Уд~ть  9Ме~МС
```

Рис. 3.14: Название рисунка

Внимательно ознакомились с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

Открыли файл с программой lab8-2.asm и в инструкции mov с двумя операндами удалить один операнд. (рис. 3.15) Выполните трансляцию с получением файла листинга: (рис. 3.16), (рис. 3.17)



```
lab8-2.asm
~/work/arch-pc/lab08
Открыть ▾  +
C:\dd\50
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax
call sprint
```

Рис. 3.15: Название рисунка

```
[mazimov@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

Рис. 3.16: Название рисунка

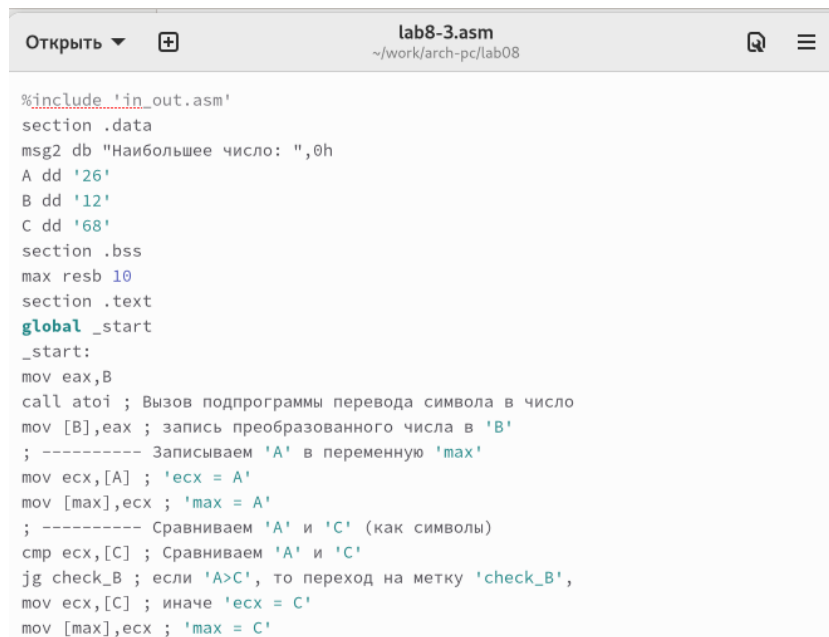
```
lab8-2.lst [----] 0 L: [ 1+ 0 1/226] *(0 /14544b) [*] [X]
1                               %include 'in_out.asm'
2                               <1> ;----- slen ---
3                               <1> ; Функция вычисления длины с
4                               <1> slen:.....
5 00000000 53                   <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,<m
1Поиск 2Сохранить 3Блок 4Замена 5Копия 6Печать 7Поиск 8Удалить 9Меню
```

Рис. 3.17: Название рисунка

Какие выходные файлы создаются в этом случае? Что добавляется в листинге?

3.3 Задание для самостоятельной работы

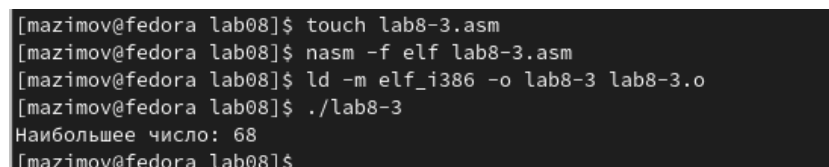
1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. (рис. 3.18) Значения переменных выбрали из таблицы в соответствии с 17 вариантом, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу. (рис. 3.19)



```
Открыть + lab8-3.asm ~/work/arch-pc/lab08

%include 'in_out.asm'
section .data
msg2 db "Наибольшее число: ",0h
A dd '26'
B dd '12'
C dd '68'
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'
```

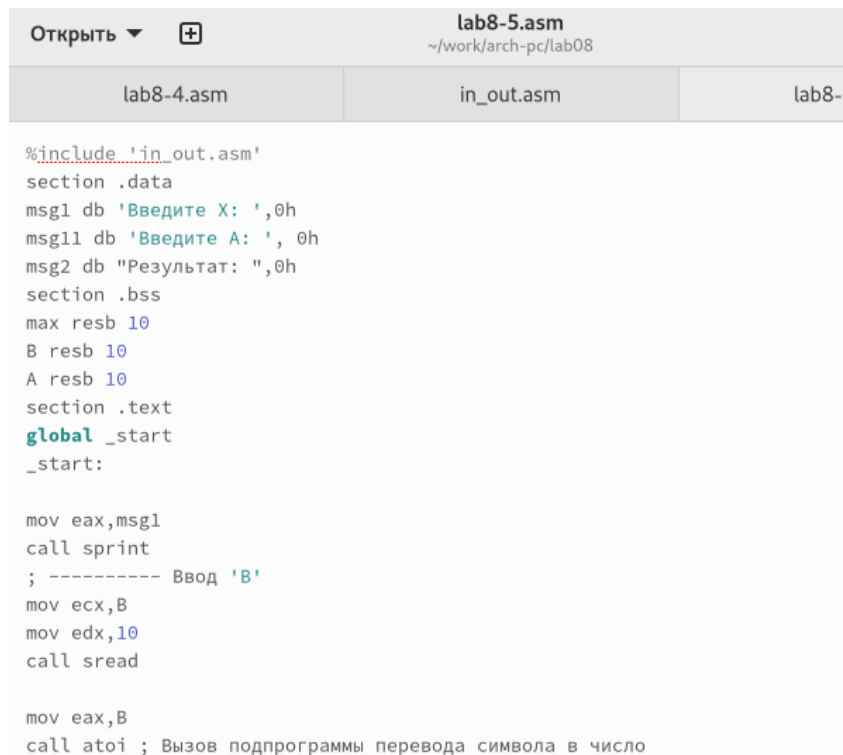
Рис. 3.18: Название рисунка



```
[mazimov@fedora lab08]$ touch lab8-3.asm
[mazimov@fedora lab08]$ nasm -f elf lab8-3.asm
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[mazimov@fedora lab08]$ ./lab8-3
Наибольшее число: 68
[mazimov@fedora lab08]$
```

Рис. 3.19: Название рисунка

2. Написали программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. (рис. 3.20) Вид функции $f(x)$ выбрали из таблицы вариантов заданий в соответствии с вариантом 17, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу для значений x и a . (рис. 3.21)



```
lab8-5.asm
~/work/arch-pc/lab08

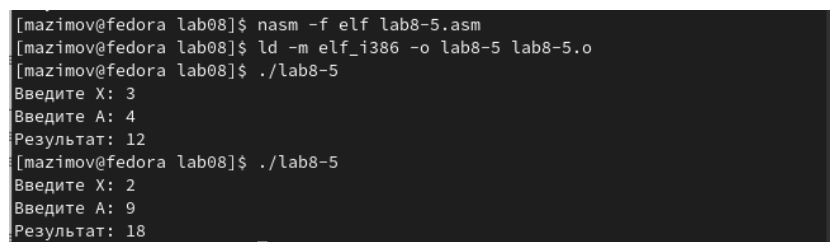
lab8-4.asm  in_out.asm  lab8-5.asm

#include 'in_out.asm'
section .data
msg1 db 'Введите X: ',0h
msg11 db 'Введите A: ', 0h
msg2 db "Результат: ",0h
section .bss
max resb 10
B resb 10
A resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 3.20: Название рисунка



```
[mazimov@fedora lab08]$ nasm -f elf lab8-5.asm
[mazimov@fedora lab08]$ ld -m elf_i386 -o lab8-5 lab8-5.o
[mazimov@fedora lab08]$ ./lab8-5
Введите X: 3
Введите A: 4
Результат: 12
[mazimov@fedora lab08]$ ./lab8-5
Введите X: 2
Введите A: 9
Результат: 18
```

Рис. 3.21: Название рисунка

4 Выводы

В ходе выполнения лабораторной работы были изучены команды условного и безусловного переходов. Были приобретены навыки написания программ с использованием переходов. Ознакомились с назначением и структурой файла листинга.