

Лабораторная работа №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Акунаева Антонина Эрдниевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Реализация переходов в NASM	7
Изучение структуры файлы листинга	14
Описание результатов выполнения заданий для самостоятельной работы	18
Выводы	23

Список иллюстраций

1	Использование команд mkdir и touch	7
2	mcedit: lab7-1.asm	8
3	Трансляция, компоновка и запуск исполняемого файла lab7-1	8
4	mcedit: изменённый lab7-1.asm	9
5	lab7-1: изменение вывода сообщений	10
6	mcedit: новые изменения lab7-1.asm	11
7	lab7-1: изменение вывода сообщений: все три	11
8	Использование touch: lab7-2.asm	12
9	mcedit: lab7-1.asm. часть 1	13
10	mcedit: lab7-1.asm. часть 2	14
11	Трансляция, компоновка и запуск исполняемого файла lab7-2	14
12	Создание листинга lab7-2.asm	15
13	mcedit: листинг lab7-2.asm	15
14	mcedit: файл lab7-2.asm. Удаление операнда	16
15	Неудачная трансляция файла lab7-2.asm	17
16	mcedit: листинг lab7-2.lst с ошибкой	17
1	touch lab7-3.asm	18
2	mcedit: lab7-3.asm	19
3	Трансляция, компоновка и запуск исполняемого файла lab7-3	20
4	Вариант 13	20
5	touch lab7-4.asm	20
6	mcedit: lab7-4.asm. part 1	21
7	mcedit: lab7-4.asm. part 2	22
8	Трансляция, компоновка и запуск исполняемого файла lab7-4	22

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

Научиться реализовывать переходы в NASM.

Изучить структуры файлы листинга.

Выполнение лабораторной работы

Реализация переходов в NASM

3.1.1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm.

```
aeakunaeva@fedora:~$ mkdir ~/work/arch-pc/lab07
aeakunaeva@fedora:~$ cd ~/work/arch-pc/lab07
aeakunaeva@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
lab7-1.asm
```

Рис. 1: Использование команд mkdir и touch

Создадим каталог lab07 в рабочем каталоге при помощи mkdir, перейдём в него с cd. В новом каталоге создадим NASM-файл lab7-1.asm при помощи touch.

3.1.2. Введите в файл lab7-1.asm текст программы из листинга 7.1. Создайте исполняемый файл и запустите его. Результат работы данной программы будет следующим:

```
user@dk4n31:~$ ./lab7-1
```

Сообщение № 2

Сообщение № 3

```
user@dk4n31:~$
```

```
lab7-1.asm [-M--] 49 L:[ 1+26 27/ 27] *[*][X]
%include<-----> 'in_out.asm'<-> ; подключение внешнего

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
```

1По~щ~ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск

Рис. 2: mcedit: lab7-1.asm

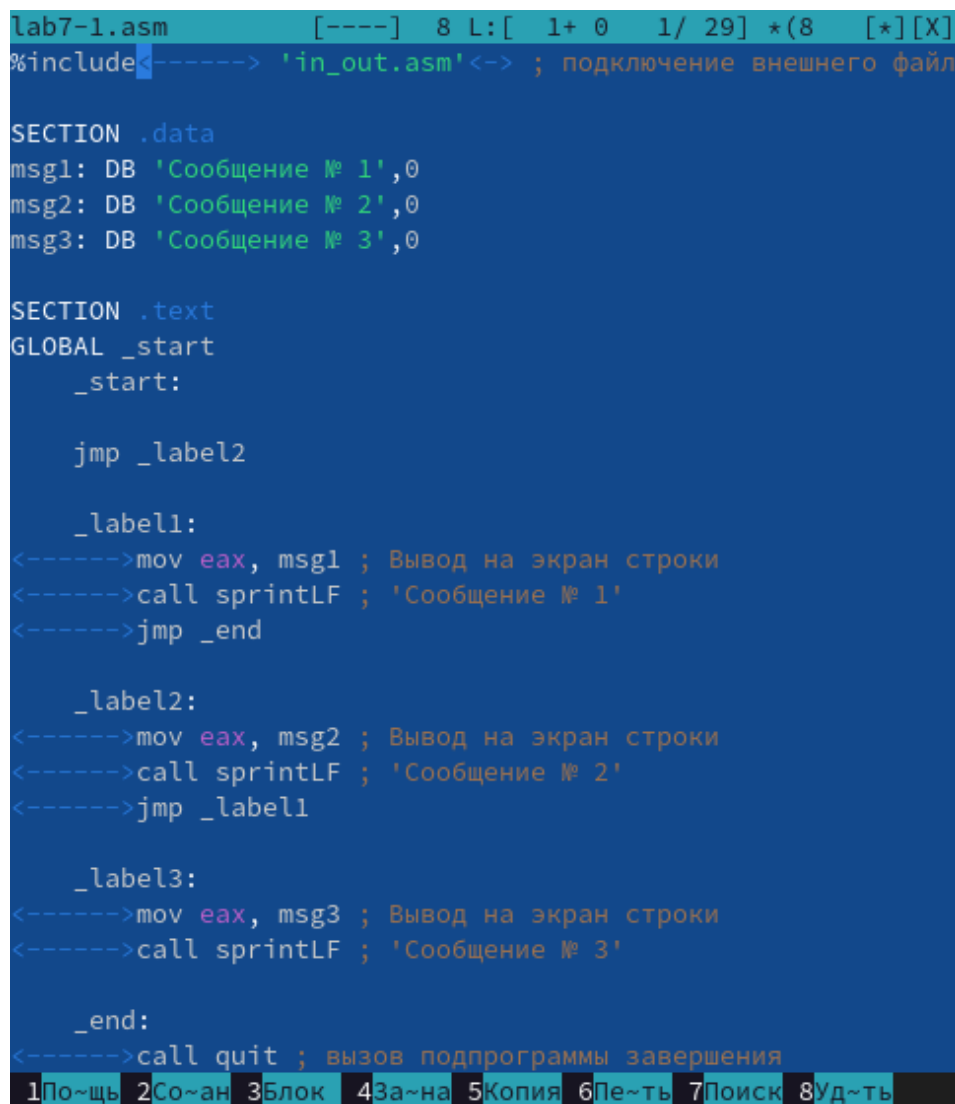
```
aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aeakunaeva@fedora:~/work/arch-pc/lab07$
```

Рис. 3: Трансляция, компоновка и запуск исполняемого файла lab7-1

Откроем файл lab7-1.asm в mcedit. Скопируем текст листинга 7.1 в файл и сохраним. Затем оттранслируем, скомпонуем и запустим исполняемый файл lab7-1. Результат совпадает с

предложенным в примере, потому что, несмотря на наличие в листинге сообщения 1, команда `jmp _label2` пропускает первое и переходит сразу ко второму сообщению.

Измените текст программы в соответствии с листингом 7.2. Создайте исполняемый файл и проверьте его работу.



```
lab7-1.asm      [----]  8 L:[  1+ 0  1/ 29] *(8  [*] [X])
%include<-----> 'in_out.asm'<-> ; подключение внешнего файл

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'
<----->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'

_end:
<----->call quit ; вызов подпрограммы завершения
1По~щ~ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть
```

Рис. 4: mcedit: изменённый lab7-1.asm

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
aeakunaeva@fedora:~/work/arch-pc/lab07$ █

```

Рис. 5: lab7-1: изменение вывода сообщений

Скопируем текст листинга 7.2 в файл и сохраним. Затем оттранслируем, скомпилируем и запустим исполняемый файл lab7-1. Как и до этого, программа начинает с сообщения 2 из-за команды `jmp _label2`, но в секции с сообщением 2 есть команда `jmp _label1`, потому выводится первое сообщение, а затем осуществляется переход в `jmp _end`, и программа завершает работу.

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:

```

user@dk4n31:~$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@dk4n31:~$

```

```

lab7-1.asm      [-M--] 49 L:[ 1+29 30/ 30] *(7[*][X]
%include<-----> 'in_out.asm'<-> ; подключение внешнего ф

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 ; вызов подпрограммы завершения
1По~щ~ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть

```

Рис. 6: mcedit: новые изменения lab7-1.asm

```

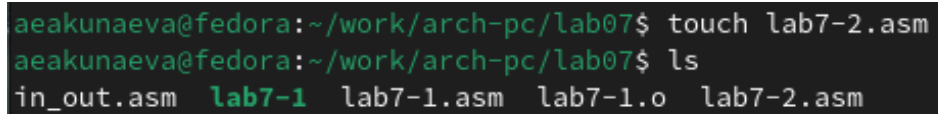
aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 7: lab7-1: изменение вывода сообщений: все три

Изменим первый `jmp` на `jmp _label3`, тогда начинаем с третьего сообщения, в нём запишем команду `jmp _label2`, выводится второе, оставляем остальное, как есть, тогда следующим выводится сообщение 1 и программа завершает работу.

3.1.3. Создайте файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Внимательно изучите текст программы из листинга 7.3 и введите в `lab7-2.asm`. Создайте исполняемый файл и проверьте его работу для разных значений `B`.



```
aeakunaeva@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
```

Рис. 8: Использование `touch`: `lab7-2.asm`

Создадим файл `lab7-2.asm` при помощи `touch` в текущей директории.

```

lab7-2.asm      [-M--] 13 L:[ 1+ 9 10/ 50] *(194 /188[*])[X]
#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'
; ----- 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'
1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9Ме~МС

```

Рис. 9: mcedit: lab7-1.asm. часть 1

```

    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
1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9Ме~МС

```

Рис. 10: mcedit: lab7-1.asm. часть 2

Скопируем текст из листинга 7.3 в файл lab7-2.asm, открыв его в mcedit.

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 12
Наибольшее число: 50
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 100
Наибольшее число: 100
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 11: Трансляция, компоновка и запуск исполняемого файла lab7-2

Запустим исполняемый файл lab7-2 несколько раз и проверим разные значения B. Так как по умолчанию A = 20, C = 50, то первое максимальное значение будет 50 (при B = 12), второе - 100 (B = 100), третье - 50 (B = 30).

Изучение структуры файлы листинга

3.2.1. Создайте файл листинга для программы из файла lab7-2.asm. Откройте файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit. Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst
lab7-1 lab7-1.o lab7-2.asm lab7-2.o
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 12: Создание листинга lab7-2.asm

Создадим листинг для lab7-2.asm, указав ключ -l и задав название файла листинга lab7-2.lst.

```

lab7-2.lst [----] 25 L: [ 99+ 0 99/226] *(6106/14635b) 0032 0[*][X]
 98 0000007C 83F900 <1> cmp ecx, 0...
 99 0000007F 75F2 <1> jnz printLoop..
100 <1> .
101 00000081 5E <1> pop esi...
102 00000082 5A <1> pop edx...
103 00000083 59 <1> pop ecx...
104 00000084 58 <1> pop eax.....
105 00000085 C3 <1> ret
106 <1> .
107 <1> .
108 <1> ;----- iprintLF -----
109 <1> ; Функция вывода на экран чисел в ф
110 <1> ; входные данные: mov eax,<int>
111 <1> iprintLF:
112 00000086 E8C9FFFFFF <1> call iprint.....
113 <1> .
114 0000008B 50 <1> push eax.....
115 0000008C B80A000000 <1> mov eax, 0Ah.....
116 00000091 50 <1> push eax.....
117 00000092 89E0 <1> mov eax, esp.....
118 00000094 E876FFFFFF <1> call sprint.....
119 00000099 58 <1> pop eax.....
120 0000009A 58 <1> pop eax.....
121 0000009B C3 <1> ret
122 <1> .
123 <1> ;----- atoi -----
124 <1> ; Функция преобразования ascii-код
125 <1> ; входные данные: mov eax,<int>
126 <1> atoi:
127 0000009C 53 <1> push ebx.....
128 0000009D 51 <1> push ecx.....
129 0000009E 52 <1> push edx.....
1 По-щ 2 Со-ан 3 Блок 4 Замена 5 Копия 6 Пе-ть 7 Поиск 8 Уда-ть 9 МенюМС 10 Выход

```

Рис. 13: mcedit: листинг lab7-2.asm

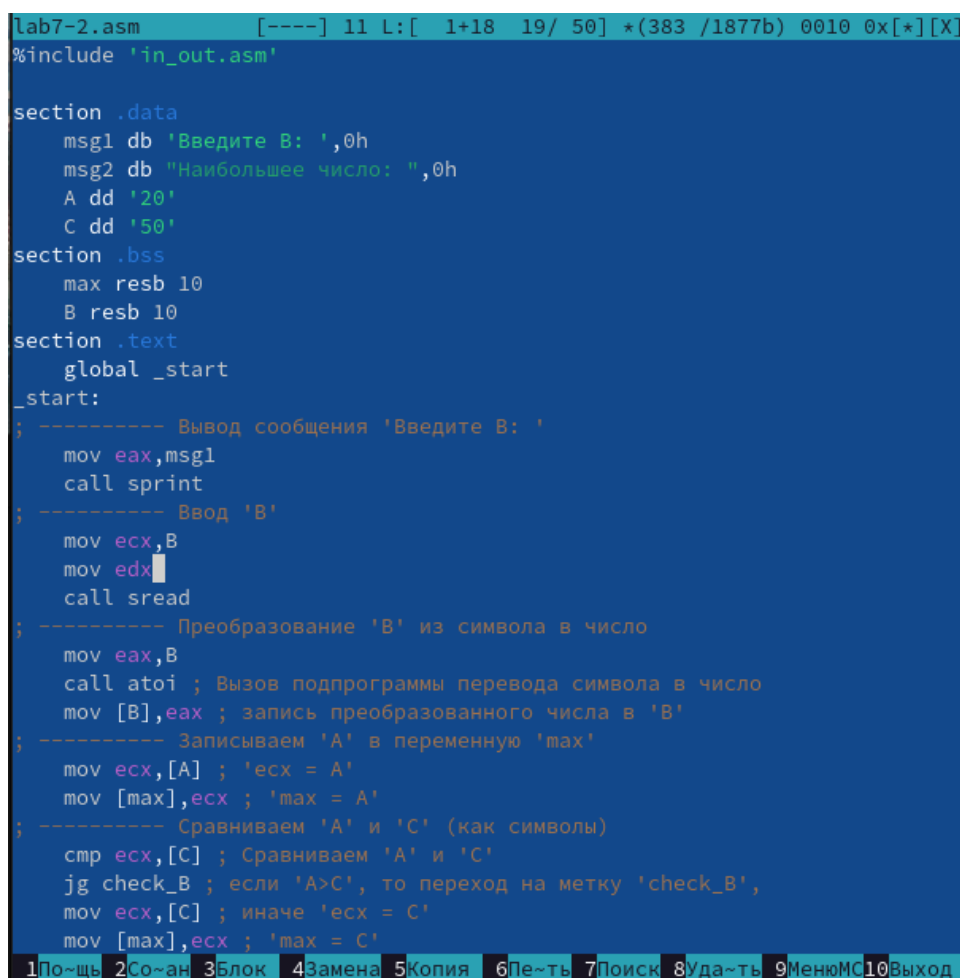
Откроем листинг в текстовом редакторе при помощи mcedit. Изучим содержимое: в листинге указан наш код программы и пояснение слева в машинном коде.

Строка 115: адрес в сегменте кода 0000008C, машинный код B80A000000, команда из текста файла .asm mov eax, 0Ah - присвоение eax значения 0Ah по таблице ASCII.

Строка 116: адрес в сегменте кода 00000091, машинный код 50, команда из текста файла .asm push eax - позволяет сохранить начальные данные регистров при старте программы с последующим восстановлением.

Строка 117: адрес в сегменте кода 00000092, машинный код 89E0, команда из текста файла .asm mov eax,esp - присвоение переменной eax значения из esp.

Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга. Какие выходные файлы создаются в этом случае? Что добавляется в листинге?



```
lab7-2.asm [----] 11 L: [ 1+18 19/ 50] *(383 /1877b) 0010 0x[*][X]
#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
    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 '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'

1По~щ 2Со~ан 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюМС10Выход
```

Рис. 14: mscedit: файл lab7-2.asm. Удаление операнда


```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:19: error: invalid combination of opcode and operands
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 15: Неудачная трансляция файла lab7-2.asm

```

16 000000ED E81DFFFFFF      call sprint
17                                ; ----- Ввод 'B'
18 000000F2 B9[0A000000]      mov ecx,B
19                                mov edx
19                                ***** error: invalid combination of opcode and operands
20 000000F7 E847FFFFFF      call sread
21                                ; ----- Преобразование 'B' из символа в число
22 000000FC B8[0A000000]      mov eax,B
23 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в
24 00000106 A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
25                                ; ----- Записываем 'A' в переменную 'max'
26 0000010B 8B0D[35000000]      mov ecx,[A] ; 'ecx = A'

```

Рис. 16: mcedit: листинг lab7-2.lst с ошибкой

Удалим операнд 10 в `mov edx, 10`. Попробуем оттранслировать файл, но отобразится ошибка, т.к. не хватает операнда. Тем не менее листинг создан, откроем его и посмотрим. В строке 19 листинга появляется текст ошибки, которая выводится на экран.

Описание результатов выполнения заданий для самостоятельной работы

4.1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Создайте исполняемый файл и проверьте его работу.

Вариант 13 из предыдущей лабораторной. Значения 84, 32, 77.

```
aeakunaeva@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst
lab7-1      lab7-1.o    lab7-2.asm  lab7-3.asm
```

Рис. 1: touch lab7-3.asm

```
lab7-3.asm [----] 13 L:[ 1+32 33/ 33] *[*][X]
#include 'in_out.asm'
section .data
    msg1 db "Наименьшее число: ",0h
    A dd '84'
    C dd '32'
    B dd '77'
section .bss
    min resb 10
section .text
    global _start
_start:
    mov ecx,[A]
    mov [min],ecx
    cmp ecx,[C]
    jl check_B
    mov ecx,[C]
    mov [min],ecx
check_B:
    mov eax,min
    call atoi
    mov [min],eax

    mov ecx,[min]
    cmp ecx,[B]
    jl fin
    mov ecx,[B]
    mov [min],ecx
fin:
    mov eax, msg1
    call sprint
    mov eax,[min]
    call iprintLF
    call quit
```

1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск

Рис. 2: mcedit: lab7-3.asm

Напишем программу нахождения минимального значения из 3-ёх переменных.

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab
7-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386
-o lab7-3 lab7-3.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 32
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 3: Трансляция, компоновка и запуск исполняемого файла lab7-3

Создадим и запустим исполняемый файл. Программа успешно выводит наименьшее значение 32.

4.2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Вариант 13 из предыдущей лабораторной.

$$13 \quad \begin{cases} a - 7, & a \geq 7 \\ ax, & a < 7 \end{cases} \quad (3;9) \quad (6;4)$$

Рис. 4: Вариант 13

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ touch lab7-4.as
m
aeakunaeva@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.o    lab7-2.lst  lab7-3.o
lab7-1      lab7-2      lab7-3      lab7-4.asm
lab7-1.asm  lab7-2.asm  lab7-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 5: touch lab7-4.asm

```

lab7-4.asm [----] 13 L: [ 1+22 23/ 49] *[*][X]
#include 'in_out.asm'
section .data
    msg1 db 'Введите x: ',0h
    msg2 db 'Введите a: ',0h
    msg3 db "f(x) = ",0h
section .bss
    x: resb 80
    a: resb 80
    res: resb 80
section .text
    global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    mov [x],eax

    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,80
    call sread
    mov eax,a
    call atoi
    mov [a],eax

    mov ecx,[a]
    mov [res],ecx

```

Рис. 6: mcedit: lab7-4.asm. part 1

```

    cmp ecx,7
    jl check_A
    sub ecx,7
    mov [res],ecx
    jmp fin
check_A:
    mov eax,[a]
    mov ecx,[x]
    mul ecx
    mov [res],eax
    jmp fin
fin:
    mov eax,msg3
    call sprint
    mov eax,[res]
    call iprintLF

    call quit

```

1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск

Рис. 7: mcedit: lab7-4.asm. part 2

Напишем программу нахождения значения функции для вводимых с клавиатуры переменных.

```

aeakunaeva@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aeakunaeva@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 9
f(x) = 2
aeakunaeva@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 6
Введите a: 4
f(x) = 24
aeakunaeva@fedora:~/work/arch-pc/lab07$

```

Рис. 8: Трансляция, компоновка и запуск исполняемого файла lab7-4

Создадим и запустим исполняемый файл. Введём значения переменных с клавиатуры. Программа успешно выводит значение функции для заданных x и a.

Выводы

Я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.