

# **Очёт по лабораторной работе № 8**

**Архитектура Компьютера**

Чепелевич Владислав Олегович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация переходов в NASM . . . . .	6
3.2	Изучение структуры файлы листинга . . . . .	10
3.3	Задание для самостоятельной работы . . . . .	13
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	lab8-1.asm . . . . .	6
3.2	Текст программы . . . . .	6
3.3	Результат работы . . . . .	7
3.4	Использование инструкций . . . . .	7
3.5	Текст программы . . . . .	8
3.6	Инструкции jmp . . . . .	8
3.7	Исполняемый файл . . . . .	8
3.8	lab8-2.asm . . . . .	9
3.9	Текст программы . . . . .	9
3.10	Исполняемый файл . . . . .	10
3.11	Ключ -l . . . . .	10
3.12	mcedit . . . . .	11
3.13	lab8-2.lst . . . . .	11
3.14	lab8-2.asm . . . . .	12
3.15	mcedit . . . . .	12
3.16	lab8-2.lst . . . . .	13
3.17	lab8-3.asm . . . . .	14
3.18	Исполняемый файл . . . . .	14
3.19	lab8-4.asm . . . . .	15
3.20	Исполняемый файл . . . . .	15

# 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

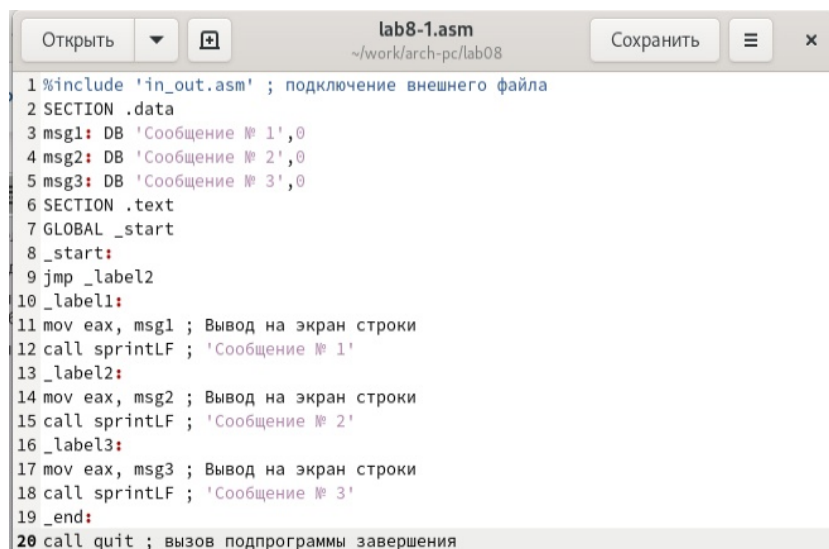
### 3.1 Реализация переходов в NASM

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

```
[CepelevichV0@fedora ~]$ mkdir ~/work/arch-pc/lab08
[CepelevichV0@fedora ~]$ cd ~/work/arch-pc/lab08
[CepelevichV0@fedora lab08]$ touch lab8-1.asm
[CepelevichV0@fedora lab08]$
```

Рис. 3.1: lab8-1.asm

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)



```
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. 3.2: Текст программы

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

```
[CepelevichV0@fedora lab08]$ nasm -f elf lab8-1.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[CepelevichV0@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Результат работы

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)

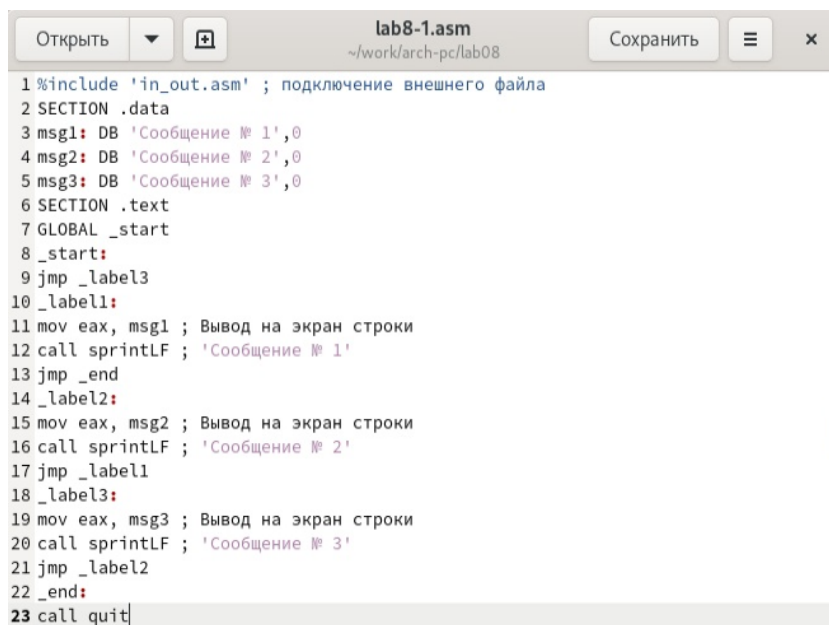
```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран стр
12 _label2:
13 mov eax, msg2 ; Вывод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

Рис. 3.4: Использование инструкций

```
[CepelevichV0@fedora lab08]$ nasm -f elf lab8-1.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[CepelevichV0@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Текст программы

Измените текст программы добавив или изменив инструкции `jmp`. (рис. 3.6), (рис. 3.7)



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

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
```

Рис. 3.6: Инструкции `jmp`

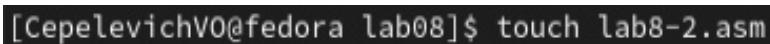
```
[CepelevichV0@fedora lab08]$ nasm -f elf lab8-1.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[CepelevichV0@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.7: Исполняемый файл

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

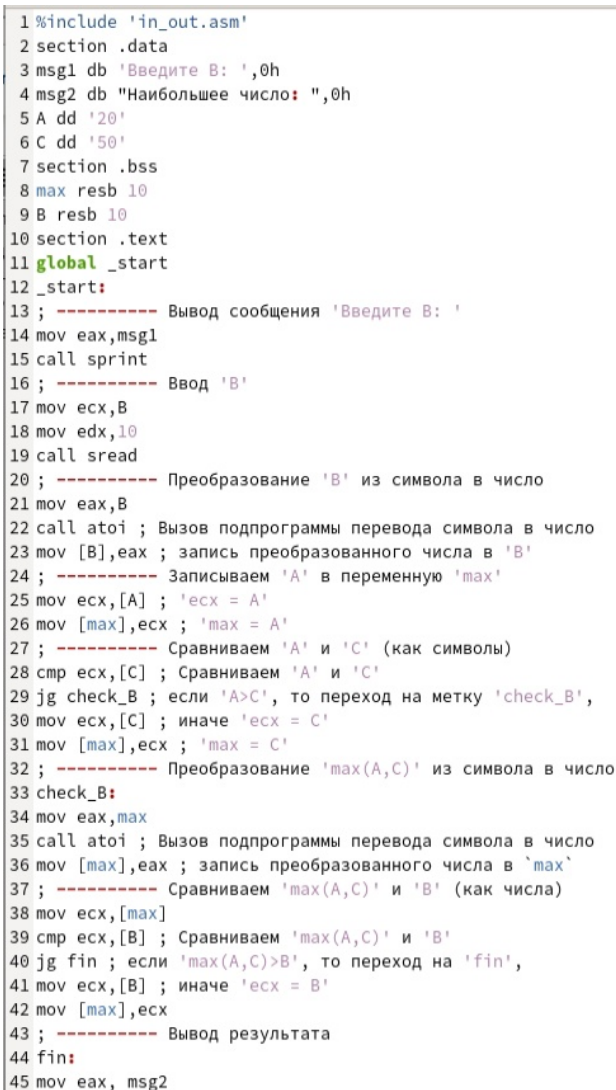


примера рассмотрели программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры. Создали файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. (рис. 3.8) Внимательно изучили текст программы из листинга 8.3 и ввели в lab8-2.asm. (рис. 3.9)



```
[CepelevichV0@fedora lab08]$ touch lab8-2.asm
```

Рис. 3.8: lab8-2.asm



```
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'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
```

Рис. 3.9: Текст программы

Создали исполняемый файл и проверили его работу для разных значений В.  
(рис. 3.10)

```
[CepelevichV0@fedora ~]$ cd ~/work/arch-pc/lab08
[CepelevichV0@fedora lab08]$ nasm -f elf lab8-2.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[CepelevichV0@fedora lab08]$ ./lab8-2
Введите В: 55
Наибольшее число: 55
[CepelevichV0@fedora lab08]$ ./lab8-2
Введите В: 5
Наибольшее число: 50
```

Рис. 3.10: Исполняемый файл

Обратили внимание, в данном примере переменные А и С сравниваются как символы, а переменная В и максимум из А и С 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. 3.11)

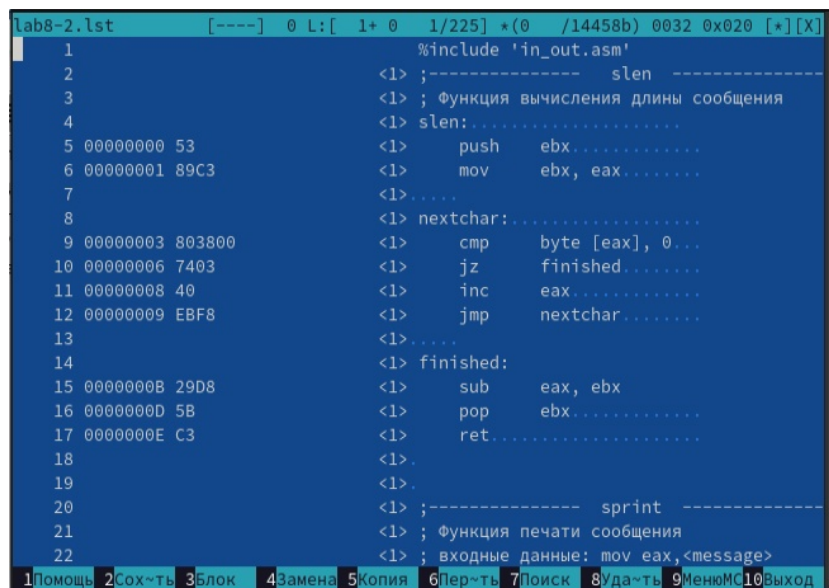
```
[CepelevichV0@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

Рис. 3.11: Ключ `-l`

Открыли файл листинга `lab8-2.lst` с помощью текстового редактора `mcedit`:  
(рис. 3.12), (рис. 3.13)

```
[CepelevichV0@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.12: mcedit



```
lab8-2.lst  [----]  0 L:  1+ 0  1/225] *(0  /14458b) 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
8          <1> .....
9          <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
14          <1> .....
14          <1> finished:
15 0000000B 29D8        <1> sub     eax, ebx
16 0000000D 5B         <1> pop     ebx.....
17 0000000E C3         <1> ret.....
18
19          <1> .....
20          <1> .....
20          <1> ;----- sprint -----
21          <1> ; Функция печати сообщения
22          <1> ; входные данные: mov eax,<message>
```

Рис. 3.13: lab8-2.lst

Внимательно ознакомились с его форматом и содержимым. Содержимое трёх строк файла листинга: 1)45 00000154 B8[13000000] mov eax, msg2 - строка 45, адрес 00000154, B8[13000000] - машинный код, mov eax, msg2 - исходный текст программы 2)46 00000159 E8B1FEFFFF call sprint - строка 46, адрес 00000159, E8B1FEFFFF - машинный код, call sprint - исходный текст программы 3)47 0000015E A1[00000000] mov eax,[max] - строка 47, адрес 0000015E, A1[00000000] - машинный код, mov eax,[max] - исходный текст программы

Открыли файл с программой lab8-2.asm и в инструкции mov с двумя операндами удалить один операнд. (рис. 3.14) Выполните трансляцию с получением файла листинга: (рис. 3.15), (рис. 3.16)

```

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,|
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'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2

```

Рис. 3.14: lab8-2.asm

```
[CepelevichV0@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

Рис. 3.15: mcedit

```

lab8-2.lst      [----]  0 L:[179+ 0 179/226] *(10993/14548b) 0032[*][X]
 4 0000002E D0BBD0BE3A2000.....
 5 00000035 32300000                A dd '20'
 6 00000039 35300000                C dd '50'
 7                                section .bss
 8 00000000 <res Ah>                max resb 10
 9 0000000A <res Ah>                B resb 10
10                                section .text
11                                global _start
12                                _start:
13                                ; ----- Вывод сообщения 'Введ
14 000000E8 B8[00000000]            mov eax,msg1
15 000000ED E81DFFFFFF            call sprint
16                                ; ----- Ввод 'В'
17                                mov ecx,
17                                ***** error: invalid combination of opc
18 000000F2 BA0A000000            mov edx,10
19 000000F7 E847FFFFFF            call sread
20                                ; ----- Преобразование 'В' из
21 000000FC B8[0A000000]            mov eax,B
22 00000101 E896FFFFFF            call atoi ; Вызов подпрограммы пер
1Поиск 2Со-ть 3Блок 4Замена 5Копия 6Пе-ть 7Поиск 8Уд-ть 9МенюMC 10Выход

```

Рис. 3.16: lab8-2.lst

Создаётся выходной файл lst. В листинге добавляется сообщение об ошибке.

### 3.3 Задание для самостоятельной работы

1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. (рис. 3.17) Значения переменных выбрали из таблицы в соответствии с 2 вариантом, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу. (рис. 3.18)

```

1 %include 'in_out.asm'
2 section .data
3 msg2 db "Наибольшее число: ",0h
4 A dd '82'
5 B dd '59'
6 C dd '61'
7 section .bss
8 max resb 10
9 section .text
10 global _start
11 _start:
12 mov eax,B
13 call atoi ; Вызов подпрограммы перевода символа в число
14 mov [B],eax ; запись преобразованного числа в 'B'
15 ; ----- Записываем 'A' в переменную 'max'
16 mov ecx,[A] ; 'ecx = A'
17 mov [max],ecx ; 'max = A'
18 ; ----- Сравниваем 'A' и 'C' (как символы)
19 cmp ecx,[C] ; Сравниваем 'A' и 'C'
20 jg check_B ; если 'A>C', то переход на метку 'check_B',
21 mov ecx,[C] ; иначе 'ecx = C'
22 mov [max],ecx ; 'max = C'
23 ; ----- Преобразование 'max(A,C)' из символа в число
24 check_B:
25 mov eax,max
26 call atoi ; Вызов подпрограммы перевода символа в число
27 mov [max],eax ; запись преобразованного числа в 'max'
28 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
29 mov ecx,[max]
30 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
31 jg fin ; если 'max(A,C)>B', то переход на 'fin',
32 mov ecx,[B] ; иначе 'ecx = B'
33 mov [max],ecx
34 ; ----- Вывод результата
35 fin:
36 mov eax, msg2
37 call sprint ; Вывод сообщения 'Наибольшее число: '
38 mov eax,[max]
39 call iprintLF ; Вывод 'max(A,B,C)'
40 call quit ; Выход

```

Рис. 3.17: lab8-3.asm

```

[CepelevichV0@fedora lab08]$ nasm -f elf lab8-3.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[CepelevichV0@fedora lab08]$ ./lab8-3
Наибольшее число: 82
[CepelevichV0@fedora lab08]$

```

Рис. 3.18: Исполняемый файл

2. Написали программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. (рис. 3.19) Вид функции  $f(x)$  выбрали из таблицы вариантов заданий в соответствии с вариантом 2, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу для значений  $x$  и  $a$ . (рис. 3.20)

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите X: ',0h
4 msg11 db 'Введите A: ', 0h
5 msg2 db "Результат: ",0h
6 section .bss
7 max resb 10
8 X resb 10
9 A resb 10
10 section .text
11 global _start
12 _start:
13
14 mov eax,msg1
15 call sprint
16
17 mov ecx,X
18 mov edx,10
19 call sread
20
21 mov eax,X
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [X],eax ; запись преобразованного числа
24
25 mov eax,msg11
26 call sprint
27
28 mov ecx,A
29 mov edx,10
30 call sread
31
32 mov eax,A
33 call atoi ; Вызов подпрограммы перевода символа в число
34 mov [A],eax ; запись преобразованного числа
35
36 mov eax, [X]
37 cmp eax, [A]
38 jl yes
39 jge no
40
41 no:
42 sub eax, 1
43 mov esi, eax
44 jmp fin
45
46 mov

```

Matlab ▾ Ширина табуляции: 8 ▾ Стр 57, Стлб 18 ▾ ВСТ

Рис. 3.19: lab8-4.asm

```

[CepelevichV0@fedora lab08]$ nasm -f elf lab8-4.asm
[CepelevichV0@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[CepelevichV0@fedora lab08]$ ./lab8-4
Введите X: 5
Введите A: 7
Результат: 6
[CepelevichV0@fedora lab08]$ ./lab8-4
Введите X: 6
Введите A: 4
Результат: 5
[CepelevichV0@fedora lab08]$

```

Рис. 3.20: Исполняемый файл

## 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.