

Отчёт по лабораторной работе №8

Тарутина Кристина Олеговна

Содержание

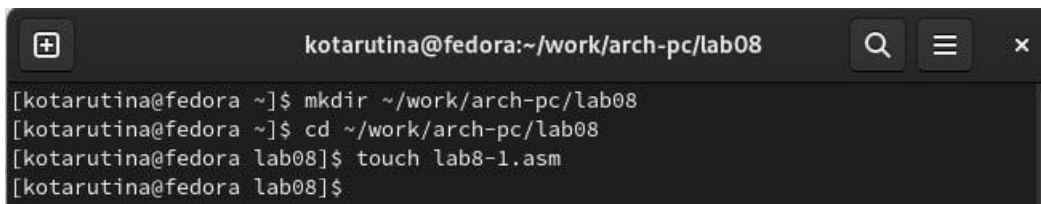
| | | |
|---|--|---|
| 1 | Цель работы | 1 |
| 2 | Выполнение лабораторной работы..... | 1 |
| 3 | Выполнение самостоятельной работы..... | 6 |
| 4 | Выводы | 7 |

1 Цель работы

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

2 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm(рис. 1)



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

Рис. 1: 1

Ввожу в файл lab8-1.asm текст программы из листинга 8.1(рис. 2)

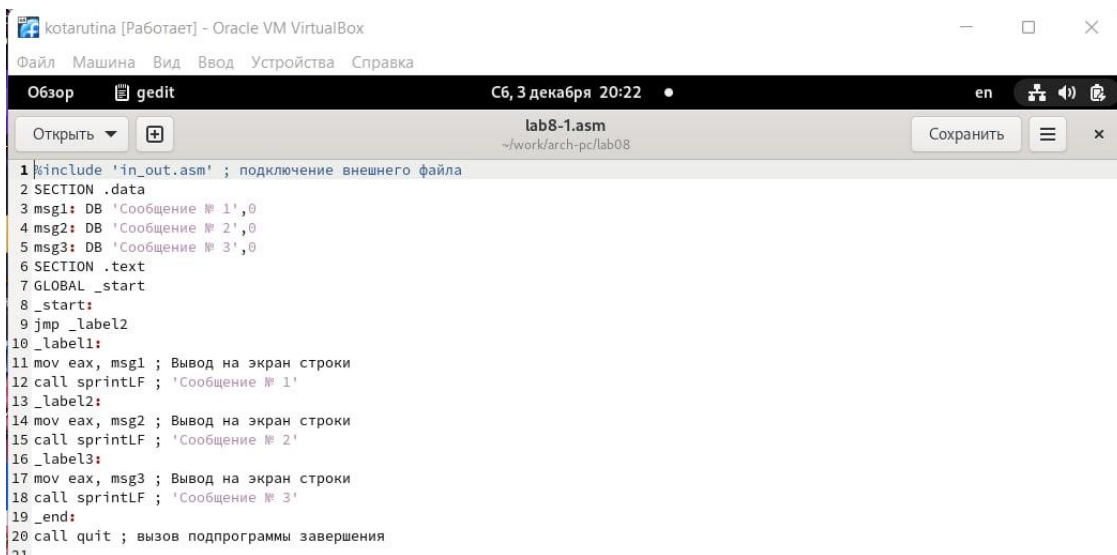


Рис. 2: 2

Создаю исполняемый файл и запускаю его.(рис. 3)

```

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

```

Рис. 3: 3

Изменяю текст программы в соответствии с листингом 8.2(рис. 4)

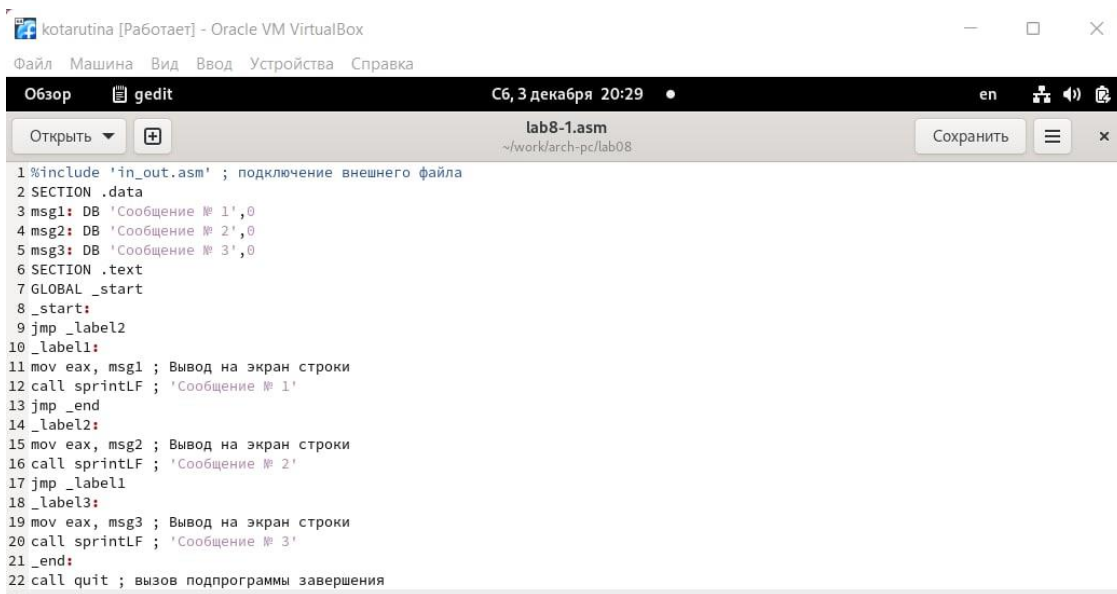


Рис. 4: 4

Создаю исполняемый файл и проверяю его работу(рис. 5)

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

Рис. 5: 5

Изменяю текст программы(рис. 6) чтобы вывод был следующим (рис. 7)



Рис. 6: 6

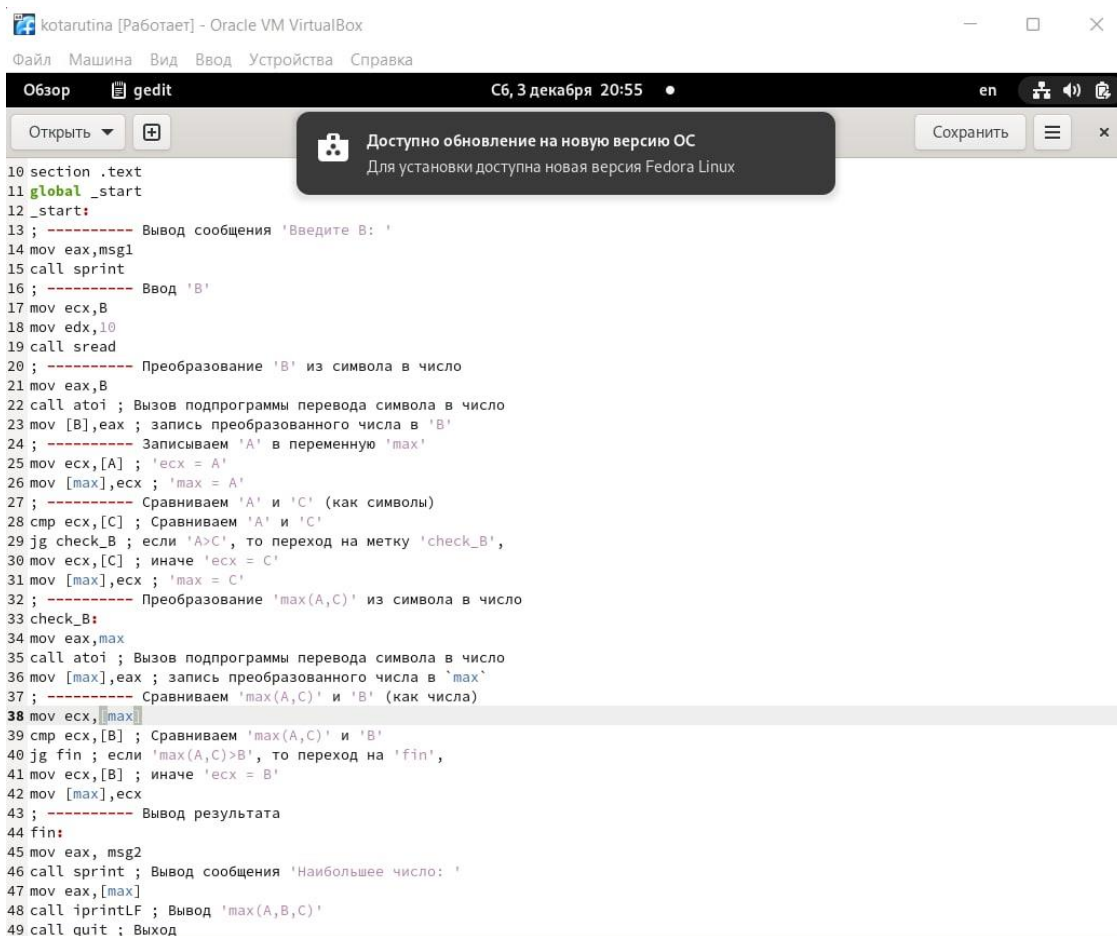
```
[kotarutina@fedora lab08]$ nasm -f elf lab8-1.asm
[kotarutina@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[kotarutina@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 7: 7

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучаю текст программы из листинга 8.3 и ввожу в lab8-2.asm(рис. 8 - 9)

```
Сообщение № 1
[kotarutina@fedora lab08]$ touch lab8-2.asm
[kotarutina@fedora lab08]$ gedit lab8-2.asm
```

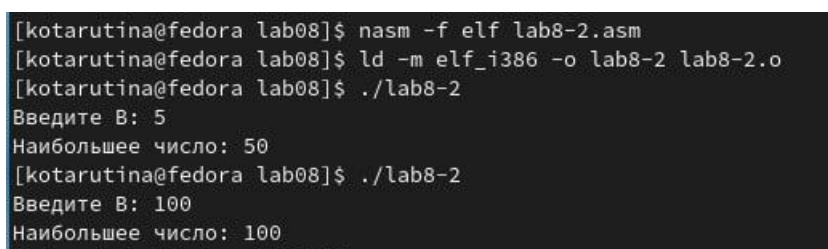
Рис. 8: 8



```
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
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
```

Рис. 9: 9

Создаю исполняемый файл и проверяю его работу для разных значений B(рис. 10)



```
[kotarutina@fedora lab08]$ nasm -f elf lab8-2.asm
[kotarutina@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[kotarutina@fedora lab08]$ ./lab8-2
Введите B: 5
Наибольшее число: 50
[kotarutina@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
```

Рис. 10: 10

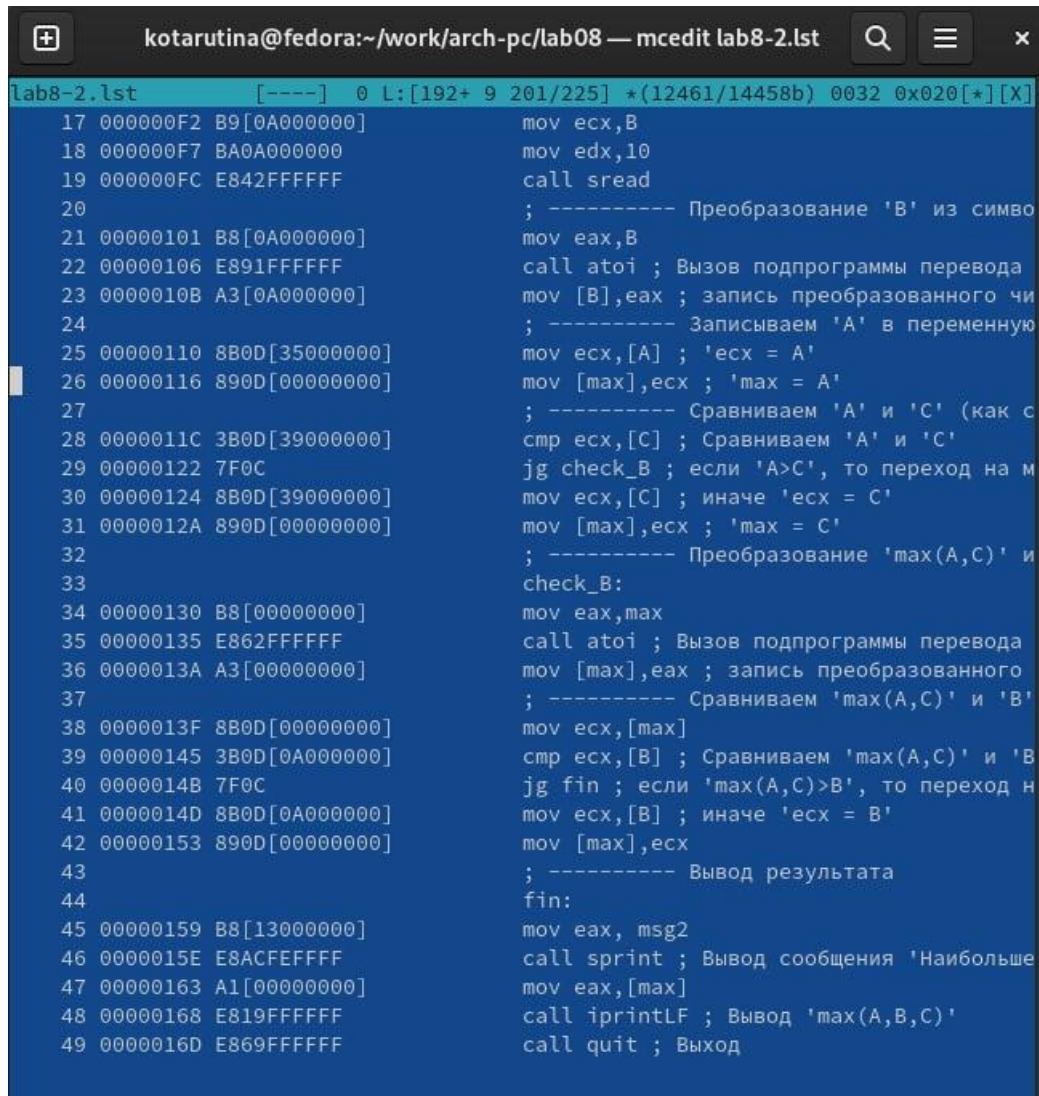
Открываю файл листинга lab8-2.lst с помощью любого текстового редактора mcedit. Внимательно ознакомливаюсь с его форматом и содержимым(рис. 11 - 12)

Рассмотрим строку под номером 34(в листинге, а не в исходном тексте программы). В ней инструкция `mov eax, max` начинается по смещению 00000130 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `mov eax, max` ассемблируется в `B8[00000000]`. В конце же строки представлена непосредственно инструкция.

Также рассмотрим строку под номером 40(в листинге, а не в исходном тексте программы). В ней инструкция `jg fin` начинается по смещению 0000014B в сегменте

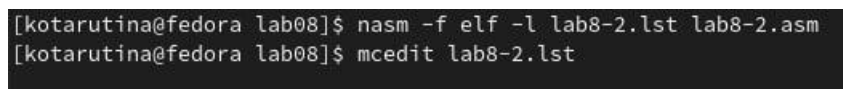
кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `jg fin` ассемблируется в `7F0C`. В конце же строки представлена непосредственно инструкция.

Рассмотрим 49 строку листинга, где 49 - номер строки, `0000016D` - смещение команды в сегменте кода, `E869FFFFFF` - машинный код ассемблерной команды `call quit`



```
lab8-2.lst      [-----]  0 L:[192+ 9 201/225] *(12461/14458b) 0032 0x020[*][X]
17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread
20                               ; ----- Преобразование 'B' из симво
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi ; Вызов подпрограммы перевода
23 0000010B A3[0A000000]      mov [B],eax ; запись преобразованного чи
24                               ; ----- Записываем 'A' в переменную
25 00000110 8B0D[35000000]      mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000]      mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как с
28 0000011C 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C              jg check_B ; если 'A>C', то переход на м
30 00000124 8B0D[39000000]      mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]      mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' и
33                               check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF      call atoi ; Вызов подпрограммы перевода
36 0000013A A3[00000000]      mov [max],eax ; запись преобразованного
37                               ; ----- Сравниваем 'max(A,C)' и 'B'
38 0000013F 8B0D[00000000]      mov ecx,[max]
39 00000145 3B0D[0A000000]      cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 0000014B 7F0C              jg fin ; если 'max(A,C)>B', то переход н
41 0000014D 8B0D[0A000000]      mov ecx,[B] ; иначе 'ecx = B'
42 00000153 890D[00000000]      mov [max],ecx
43                               ; ----- Вывод результата
44                               fin:
45 00000159 B8[13000000]      mov eax, msg2
46 0000015E E8ACFFFFFF      call sprint ; Вывод сообщения 'Наибольше
47 00000163 A1[00000000]      mov eax,[max]
48 00000168 E819FFFFFF      call iprintLF ; Вывод 'max(A,B,C)'
49 0000016D E869FFFFFF      call quit ; Выход
```

Рис. 11: 11



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

Рис. 12: 12

При попытке редактирования инструкции с двумя операндами и удаления одной из них выводится следующая ошибка(рис. 13)


```
[kotarutina@fedora lab08]$ gedit lab8-2.asm
[kotarutina@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:31: error: invalid combination of opcode and operands
```

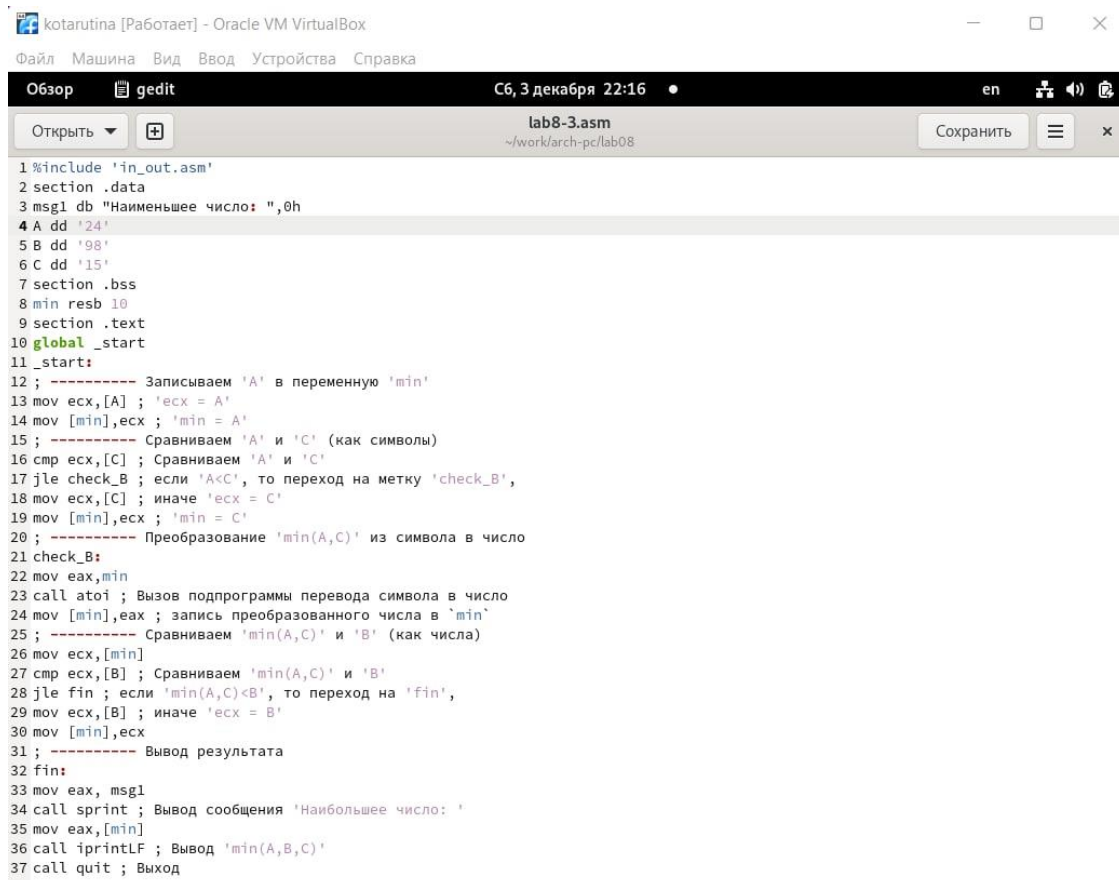
Рис. 13: 13

3 Выполнение самостоятельной работы

Создаю программу нахождения наименьшего числа среди целочисленных a, b и c (рис. 14 - 15) Вариант 9

```
[kotarutina@fedora lab08]$ nasm -f elf -l lab8-3.lst lab8-3.asm
[kotarutina@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[kotarutina@fedora lab08]$ ./lab8-3
Наименьшее число: 15
```

Рис. 14: 14



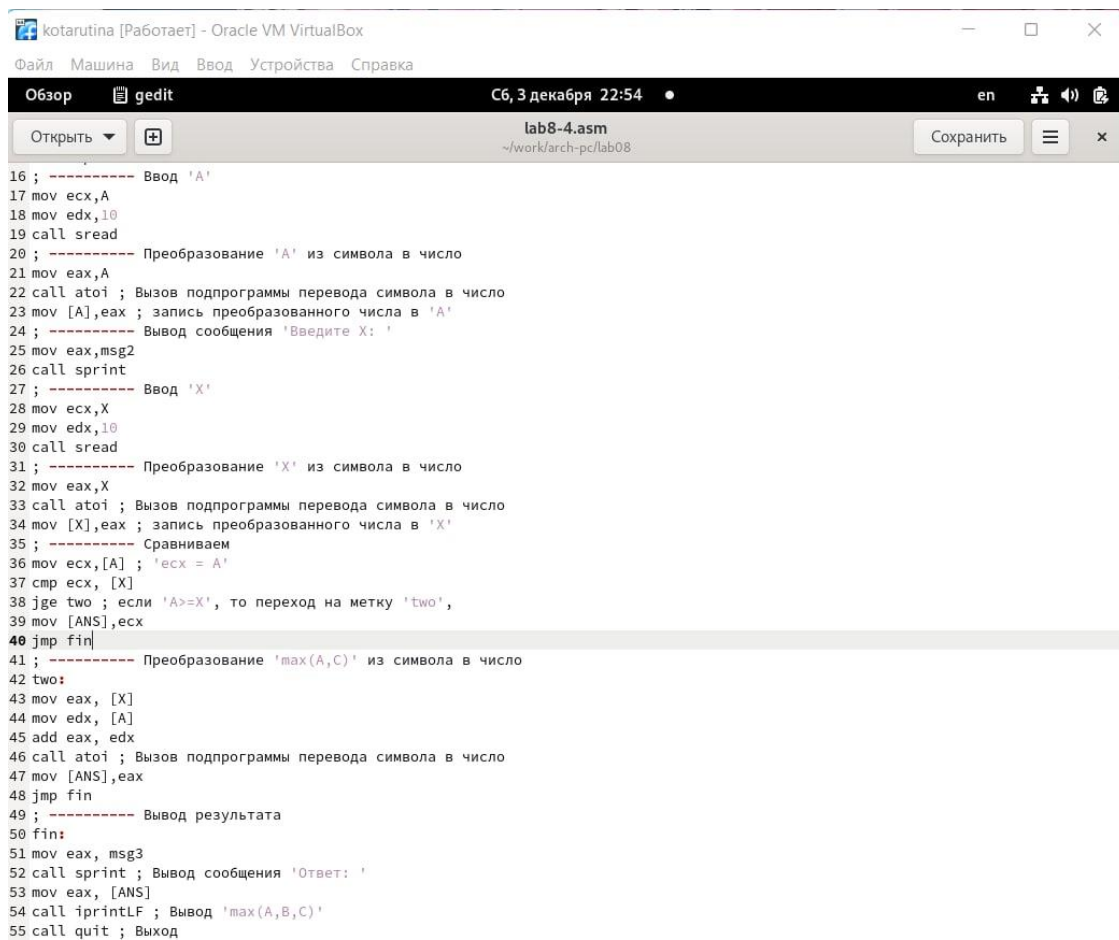
```

1 %include 'in_out.asm'
2 section .data
3 msg1 db "Наименьшее число: ",0h
4 A dd '24'
5 B dd '98'
6 C dd '15'
7 section .bss
8 min resb 10
9 section .text
10 global _start
11 _start:
12 ; ----- Записываем 'A' в переменную 'min'
13 mov ecx,[A] ; 'ecx = A'
14 mov [min],ecx ; 'min = A'
15 ; ----- Сравниваем 'A' и 'C' (как символы)
16 cmp ecx,[C] ; Сравниваем 'A' и 'C'
17 jle check_B ; если 'A<C', то переход на метку 'check_B',
18 mov ecx,[C] ; иначе 'ecx = C'
19 mov [min],ecx ; 'min = C'
20 ; ----- Преобразование 'min(A,C)' из символа в число
21 check_B:
22 mov eax,min
23 call atoi ; Вызов подпрограммы перевода символа в число
24 mov [min],eax ; запись преобразованного числа в 'min'
25 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
26 mov ecx,[min]
27 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
28 jle fin ; если 'min(A,C)<B', то переход на 'fin',
29 mov ecx,[B] ; иначе 'ecx = B'
30 mov [min],ecx
31 ; ----- Вывод результата
32 fin:
33 mov eax, msg1
34 call sprint ; Вывод сообщения 'Наименьшее число: '
35 mov eax,[min]
36 call iprintLF ; Вывод 'min(A,B,C)'
37 call quit ; Выход

```

Рис. 15: 15

Создаю программу по вычислению значений заданной функции для x и a(рис. 16)



```
16 ; ----- Ввод 'A'
17 mov ecx,A
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'A' из символа в число
21 mov eax,A
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [A],eax ; запись преобразованного числа в 'A'
24 ; ----- Вывод сообщения 'Введите X: '
25 mov eax,msg2
26 call sprint
27 ; ----- Ввод 'X'
28 mov ecx,X
29 mov edx,10
30 call sread
31 ; ----- Преобразование 'X' из символа в число
32 mov eax,X
33 call atoi ; Вызов подпрограммы перевода символа в число
34 mov [X],eax ; запись преобразованного числа в 'X'
35 ; ----- Сравниваем
36 mov ecx,[A] ; 'ecx = A'
37 cmp ecx, [X]
38 jge two ; если 'A>=X', то переход на метку 'two',
39 mov [ANS],ecx
40 jmp fin
41 ; ----- Преобразование 'max(A,C)' из символа в число
42 two:
43 mov eax, [X]
44 mov edx, [A]
45 add eax, edx
46 call atoi ; Вызов подпрограммы перевода символа в число
47 mov [ANS],eax
48 jmp fin
49 ; ----- Вывод результата
50 fin:
51 mov eax, msg3
52 call sprint ; Вывод сообщения 'Ответ: '
53 mov eax, [ANS]
54 call iprintLF ; Вывод 'max(A,B,C)'
55 call quit ; Выход
```

Рис. 16: 16

4 Выводы

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