

Лабораторная работа №8

Архитектура компьютера

Голованова Мария Константиновна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	14
5	Задание для самостоятельной работы	21
6	Выводы	24

Список иллюстраций

4.1	Создание каталога для программ лабораторной работы №8 и файла lab8-1.asm	8
4.2	Введение текста программы из листинга 8.1	9
4.3	Создание и запуск исполняемого файла lab8-1	9
4.4	Изменение текста файла lab8-1.asm в соответствии с листингом 8.2	10
4.5	Создание и запуск нового исполняемого файла lab8-1	10
4.6	Изменение текста файла lab8-1.asm	11
4.7	Создание и запуск нового исполняемого файла lab8-1	11
4.8	Создание файла lab8-2.asm	12
4.9	Введение текста программы из листинга 8.2	13
4.10	Введение текста программы из листинга 8.2	14
4.11	Создание исполняемого файла lab8 -2 и проверка его работы . . .	14
4.12	Создания файла листинга для программы из файла lab8-2.asm .	15
4.13	Содержимое файла листинга lab8-2.lst (ч.1)	15
4.14	Содержимое файла листинга lab8-2.lst (ч.2)	16
4.15	Содержимое файла листинга lab8-2.lst (ч.3)	17
4.16	Содержимое файла листинга lab8-2.lst (ч.4)	18
4.17	Содержимое файла листинга lab8-2.lst (ч.5)	19
4.18	Содержимое файла листинга lab8-2.lst (ч.6)	19
5.1	Создание файла lab8-3.asm	21
5.2	Текст программы нахождения наименьшей из 3 целочисленных переменных a,b и c	22
5.3	текст программвы нахождения наименьшей из 3 целочисленных переменных a,b и c	23
5.4	Создание исполняемого файла lab8-3 и проверка его работы . . .	23

Список таблиц

1 Цель работы

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

2 Задание

Изученить команды условного и безусловного переходов, написать программы, содержащие эти команды.

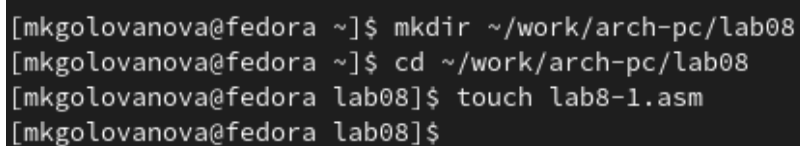
3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Выделяют 2 типа переходов: условный и безусловный. Условный переход – выполнение или невыполнение перехода в определенную точку программы в зависимости от проверки условия; безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий. Безусловный переход выполняется инструкцией `jmp`, команда имеет вид `jmp label`. Команда условного перехода имеет вид `j label`, мнемоника перехода связана со значением анализируемых флагов или со способом их формирования.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

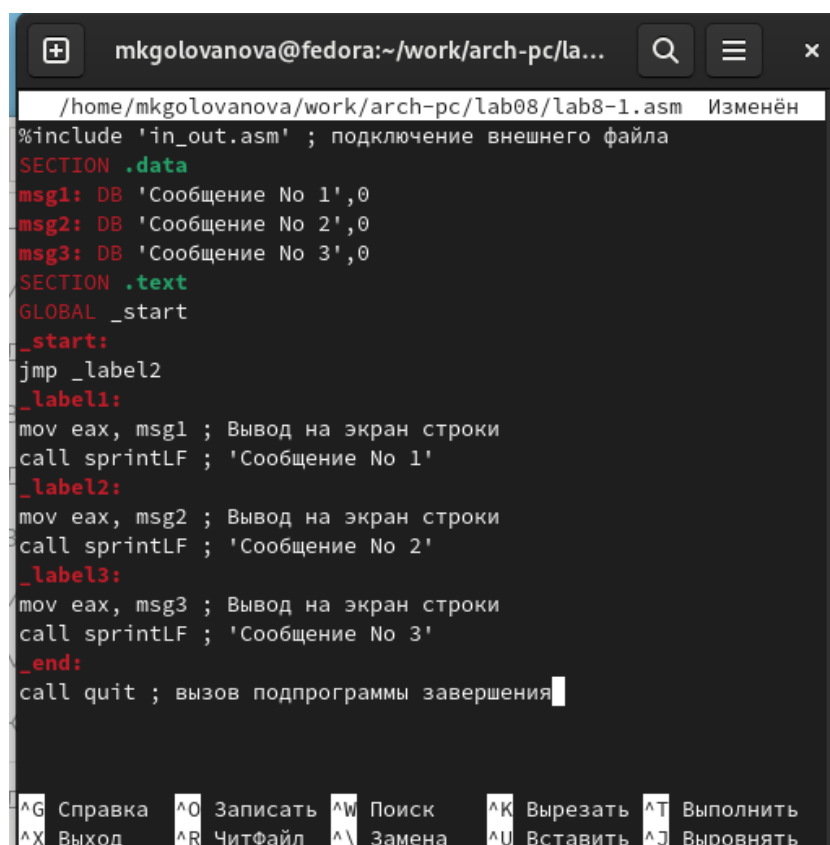
Я создала каталог для программ лабораторной работы №8, перешла в него и создала файл lab8-1.asm (рис. 4.1).



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

Рис. 4.1: Создание каталога для программ лабораторной работы №8 и файла lab8-1.asm

Я ввела в файл lab8-1.asm текст программы из листинга 8.1 (рис. 4.2).

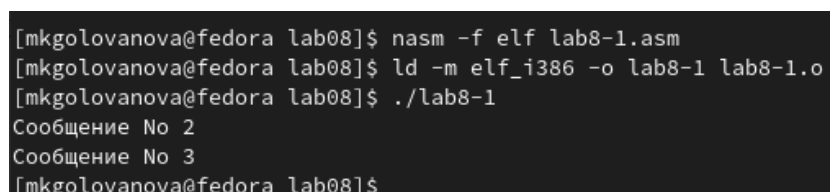


```
mkgolovanova@fedora:~/work/arch-pc/la...
/home/mkgolovanova/work/arch-pc/lab08/lab8-1.asm Изменён
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1: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

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять

Рис. 4.2: Введение текста программы из листинга 8.1

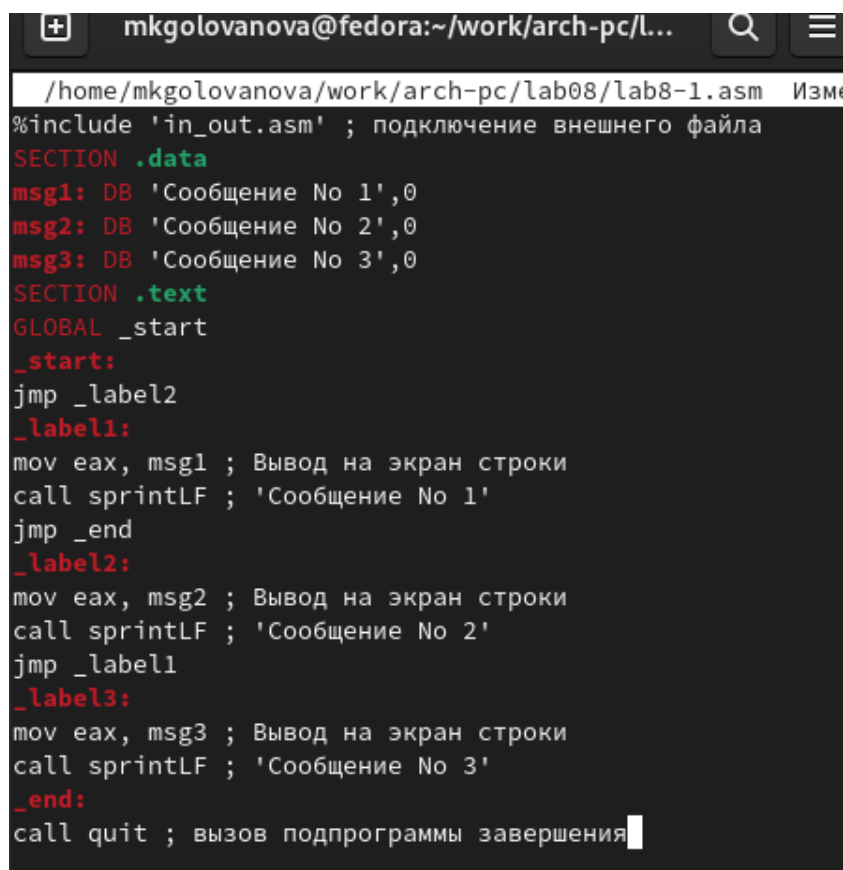
Я создала исполняемый файл и запустила его (рис. 4.3).



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

Рис. 4.3: Создание и запуск исполняемого файла lab8-1

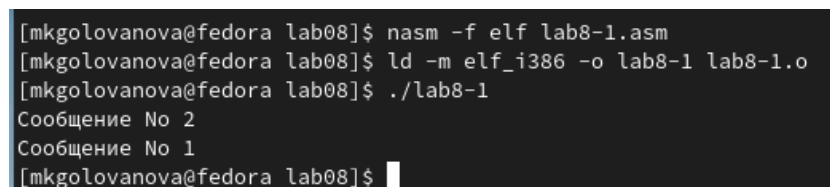
Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Я изменила текст программы в соответствии с листингом 8.2, чтобы программа выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу (рис. 4.4).



```
mkgolovanova@fedora:~/work/arch-pc/l...
/home/mkgolovanova/work/arch-pc/lab08/lab8-1.asm Изм
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение текста файла lab8-1.asm в соответствии с листингом 8.2

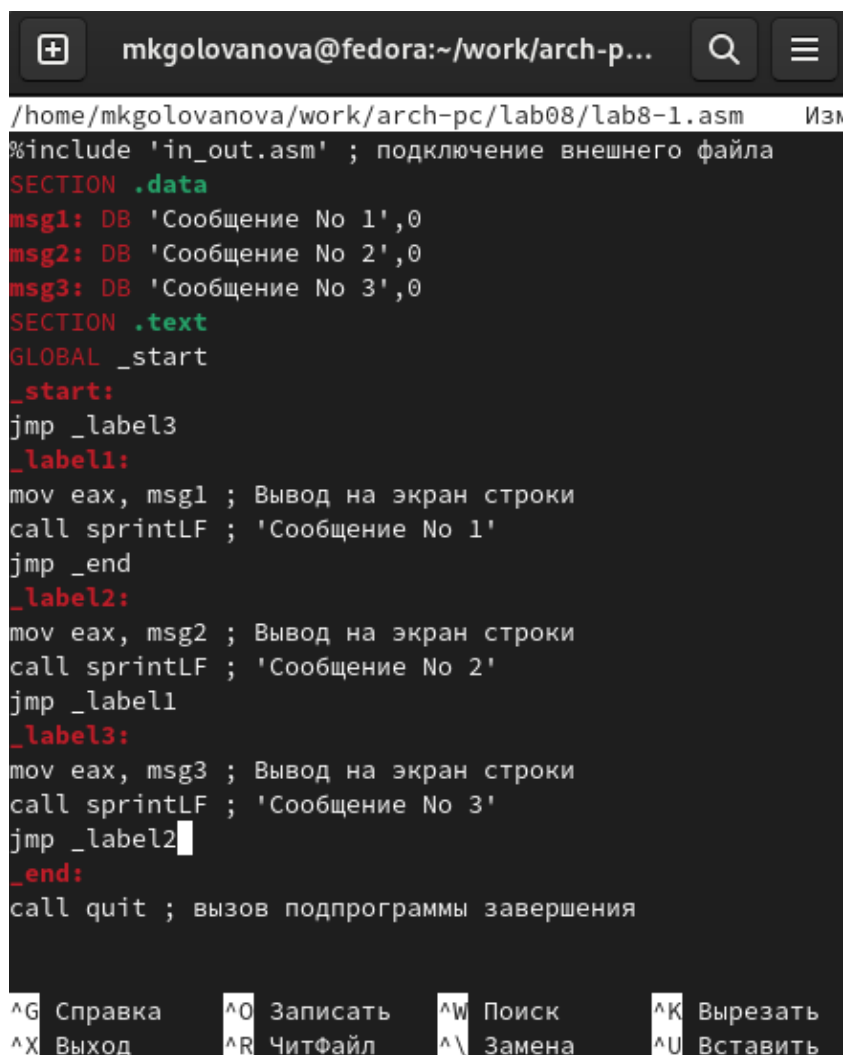
Я создала исполняемый файл и проверила его работу (рис. 4.5).



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

Рис. 4.5: Создание и запуск нового исполняемого файла lab8-1

Я изменила текст программы, добавив и изменив инструкции jmp, чтобы программа выводила 'Сообщение No 3', потом 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу (рис. 4.6).

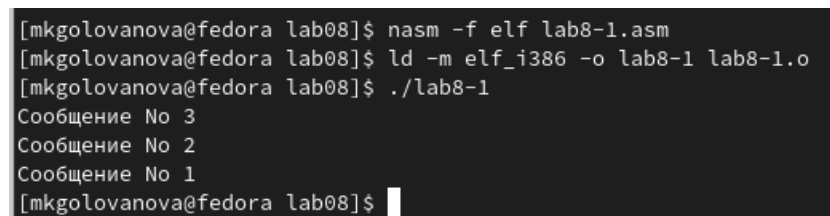


```
mkgolovanova@fedora:~/work/arch-p...
/home/mkgolovanova/work/arch-pc/lab08/lab8-1.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax,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

^G Справка ^O Записать ^W Поиск ^K Вырезать
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить

Рис. 4.6: Изменение текста файла lab8-1.asm

Я создала исполняемый файл и проверила его работу (рис. 4.7).



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

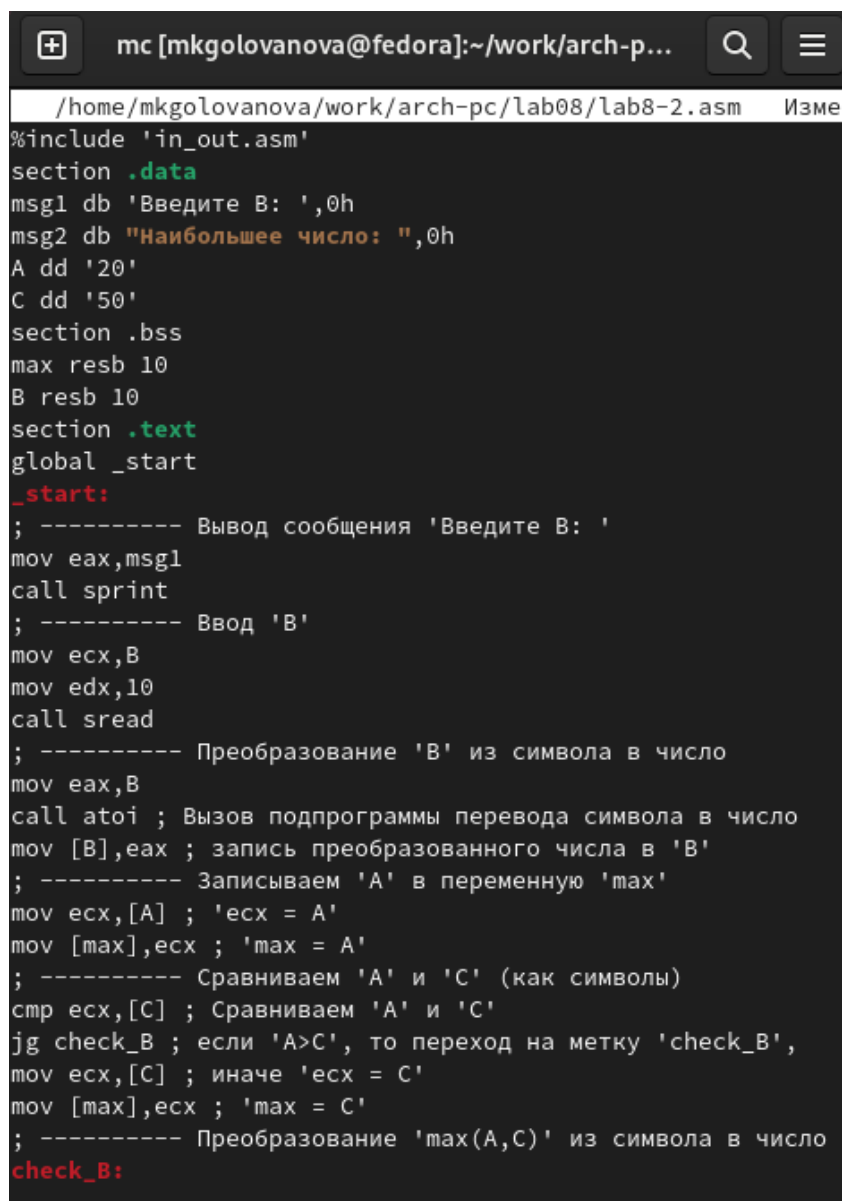
Рис. 4.7: Создание и запуск нового исполняемого файла lab8-1

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы

(переход должен происходить если выполнено какое-либо условие). В качестве примера я рассмотрела 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. Я создала файл lab8-2.asm в каталоге ~/work/arch-рс/lab08 (рис. 4.8), внимательно изучила текст программы из листинга 8.3 и ввела в lab8-2.asm (рис. 4.9, рис. 4.10).

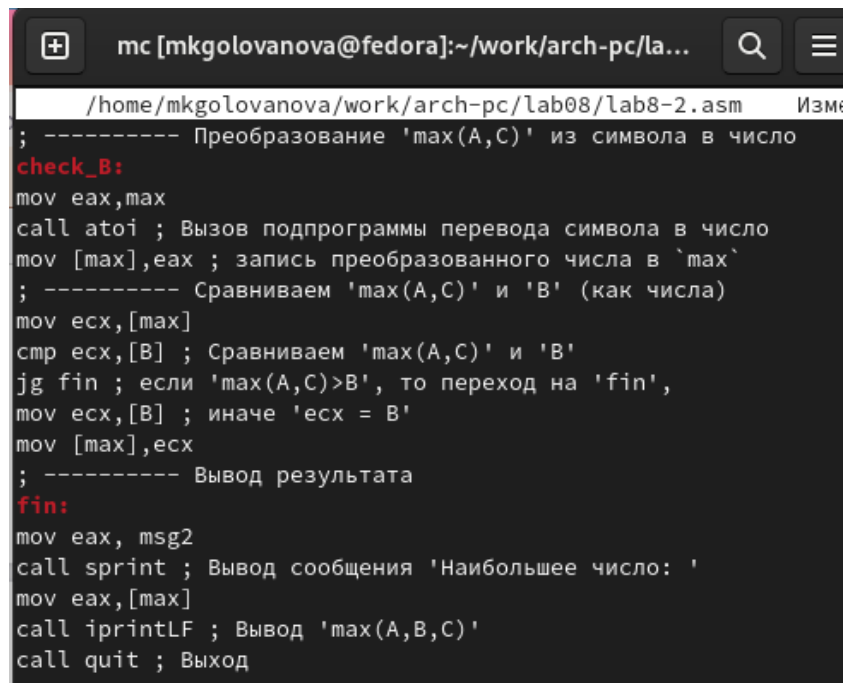
```
[mkgolovanova@fedora lab08]$ touch lab8-2.asm  
[mkgolovanova@fedora lab08]$
```

Рис. 4.8: Создание файла lab8-2.asm



```
mc [mkgolovanova@fedora]:~/work/arch-p...  Q  ≡
/home/mkgolovanova/work/arch-pc/lab08/lab8-2.asm  Изме
%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:
```

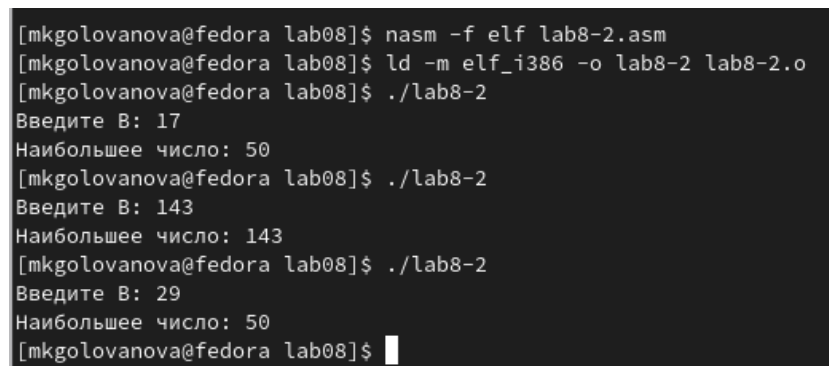
Рис. 4.9: Введение текста программы из листинга 8.2



```
mc [mkgolovanova@fedora]:~/work/arch-pc/la...
/home/mkgolovanova/work/arch-pc/lab08/lab8-2.asm Изм
; ----- 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
```

Рис. 4.10: Введение текста программы из листинга 8.2

Я создала исполняемый файл и проверила его работу для разных значений B (рис. 4.11).



```
[mkgolovanova@fedora lab08]$ nasm -f elf lab8-2.asm
[mkgolovanova@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[mkgolovanova@fedora lab08]$ ./lab8-2
Введите B: 17
Наибольшее число: 50
[mkgolovanova@fedora lab08]$ ./lab8-2
Введите B: 143
Наибольшее число: 143
[mkgolovanova@fedora lab08]$ ./lab8-2
Введите B: 29
Наибольшее число: 50
[mkgolovanova@fedora lab08]$
```

Рис. 4.11: Создание исполняемого файла lab8 -2 и проверка его работы

4.2 Изучение структуры файла листинга

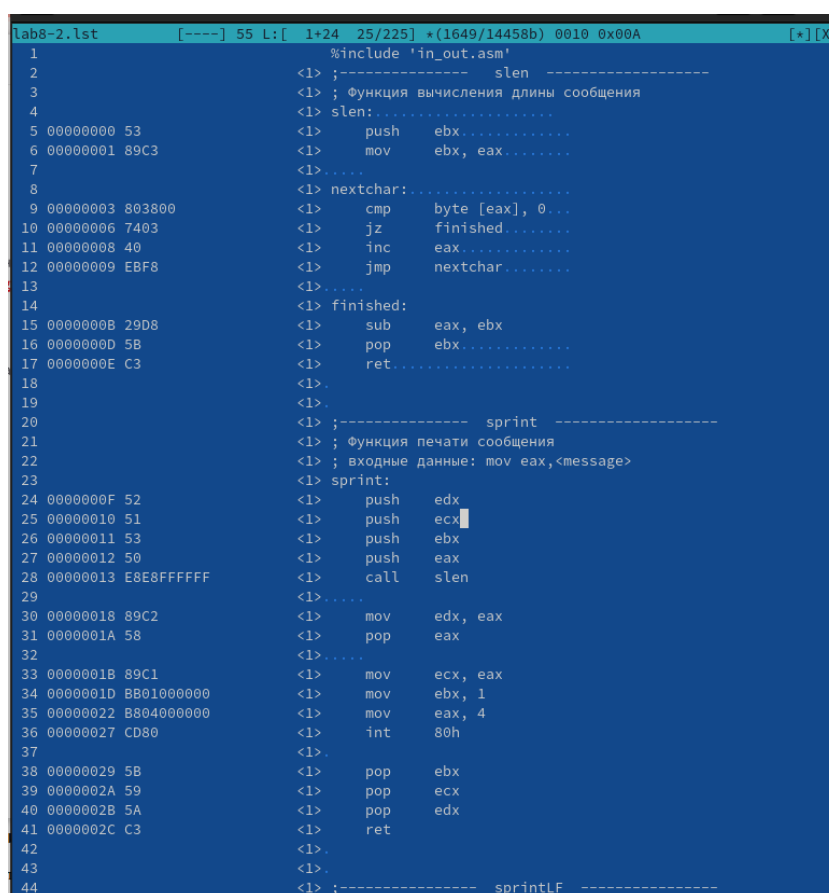
Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в

командной строке. Я создала файл листинга для программы из файла lab8-2.asm (рис. 4.12)

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

Рис. 4.12: Создания файла листинга для программы из файла lab8-2.asm

Я открыла файл листинга lab8-2.lst с помощью текстового редактора mcedit (mcedit lab8-2.lst) и внимательно ознакомилась с его форматом и содержанием (рис. 4.13, рис. 4.14, рис. 4.15, рис. 4.16, рис. 4.17, рис. 4.18)



```
lab8-2.lst  [----] 55 L: [ 1+24 25/225] *(1649/14458b) 0010 0x00A  [*][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          <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, <message>
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 E8E8FFFF   <1> call    slen
29          <1> .....
30 00000018 89C2       <1> mov     edx, eax
31 0000001A 58         <1> pop     eax
32          <1> .....
33 0000001B 89C1       <1> mov     ecx, eax
34 0000001D BB01000000   <1> mov     ebx, 1
35 00000022 B804000000   <1> mov     eax, 4
36 00000027 CD80       <1> int     80h
37          <1> .
38 00000029 5B         <1> pop     ebx
39 0000002A 59         <1> pop     ecx
40 0000002B 5A         <1> pop     edx
41 0000002C C3         <1> ret
42          <1> .
43          <1> .
44          <1> ;----- sprintLF -----
```

Рис. 4.13: Содержимое файла листинга lab8-2.lst (ч.1)

```

43      <1>.
44      <1> ;----- sprintf -----
45      <1> ; Функция печати сообщения с переводом строки
46      <1> ; входные данные: mov eax,<message>
47      <1> sprintf:
48      0000002D E8DDFFFFFF <1> call    sprintf
49      <1>.
50      00000032 50 <1> push    eax
51      00000033 B80A000000 <1> mov     eax, 0AH
52      00000038 50 <1> push    eax
53      00000039 89E0 <1> mov     eax, esp
54      0000003B E8CFFFFFFF <1> call    sprintf
55      00000040 58 <1> pop     eax
56      00000041 58 <1> pop     eax
57      00000042 C3 <1> ret
58      <1>.
59      <1> ;----- sread -----
60      <1> ; Функция считывания сообщения
61      <1> ; входные данные: mov eax,<buffer>, mov ebx,<N>
62      <1> sread:
63      00000043 53 <1> push    ebx
64      00000044 50 <1> push    eax
65      <1>...
66      00000045 B800000000 <1> mov     ebx, 0
67      0000004A B803000000 <1> mov     eax, 3
68      0000004F CD80 <1> int     80h
69      <1>.
70      00000051 5B <1> pop     ebx
71      00000052 59 <1> pop     ecx
72      00000053 C3 <1> ret
73      <1>.....
74      <1> ;----- iprint -----
75      <1> ; Функция вывода на экран чисел в формате ASCII
76      <1> ; входные данные: mov eax,<int>
77      <1> iprint:
78      00000054 50 <1> push    eax.....
79      00000055 51 <1> push    ecx.....
80      00000056 52 <1> push    edx.....
81      00000057 56 <1> push    esi.....
82      00000058 B900000000 <1> mov     ecx, 0.....
83      <1>.....
84      <1> divideLoop:
85      0000005D 41 <1> inc     ecx.....
86      0000005E BA00000000 <1> mov     edx, 0.....

```

Рис. 4.14: Содержимое файла листинга lab8-2.lst (ч.2)


```

87 00000063 BE0A000000 <1> mov esi, 10.
88 00000068 F7FE <1> idiv esi....
89 0000006A 83C230 <1> add edx, 48..
90 0000006D 52 <1> push edx...
91 0000006E 83F800 <1> cmp eax, 0...
92 00000071 75EA <1> jnz divideLoop..
93 <1>..
94 <1> printLoop:
95 00000073 49 <1> dec ecx.....
96 00000074 89E0 <1> mov eax, esp..
97 00000076 E894FFFFFF <1> call sprint...
98 0000007B 58 <1> pop eax....
99 0000007C 83F900 <1> cmp ecx, 0...
100 0000007F 75F2 <1> jnz printLoop..
101 <1>..
102 00000081 5E <1> pop esi...
103 00000082 5A <1> pop edx....
104 00000083 59 <1> pop ecx...
105 00000084 58 <1> pop eax.....
106 00000085 C3 <1> ret
107 <1>..
108 <1>..
109 <1> ;----- iprintLF -----
110 <1> ; Функция вывода на экран чисел в формате ASCII
111 <1> ; входные данные: mov eax,<int>
112 <1> iprintLF:
113 00000086 E8C9FFFFFF <1> call iprint.....
114 <1>..
115 0000008B 50 <1> push eax.....
116 0000008C B80A000000 <1> mov eax, 0Ah.....
117 00000091 50 <1> push eax.....
118 00000092 89E0 <1> mov eax, esp.....
119 00000094 E876FFFFFF <1> call sprint.....
120 00000099 58 <1> pop eax.....
121 0000009A 58 <1> pop eax.....
122 0000009B C3 <1> ret
123 <1>..
124 <1> ;----- atoi -----
125 <1> ; Функция преобразования ascii-код символа в целое число
126 <1> ; входные данные: mov eax,<int>
127 <1> atoi:
128 0000009C 53 <1> push ebx.....
129 0000009D 51 <1> push ecx.....
130 0000009E 52 <1> push edx.....

```

1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Копия 6 Переименовать 7 Поиск 8 Удалить 9 Меню MS 10 Выход

Рис. 4.15: Содержимое файла листинга lab8-2.lst (ч.3)

```

131 0000009F 56          <I>  push  esi.....
132 000000A0 89C6        <I>  mov   esi, eax.....
133 000000A2 B800000000    <I>  mov   eax, 0.....
134 000000A7 B900000000    <I>  mov   ecx, 0.....
135                                     <I> ..
136                                     <I> .multiplyLoop:
137 000000AC 31DB        <I>  xor   ebx, ebx.....
138 000000AE 8A1C0E      <I>  mov   bl, [esi+ecx]
139 000000B1 80FB30      <I>  cmp   bl, 48.
140 000000B4 7C14        <I>  jl    .finished.
141 000000B6 80FB39      <I>  cmp   bl, 57..
142 000000B9 7F0F        <I>  jg    .finished.
143                                     <I> ..
144 000000BB 80EB30      <I>  sub   bl, 48.
145 000000BE 01D8        <I>  add   eax, ebx
146 000000C0 BB0A000000    <I>  mov   ebx, 10..
147 000000C5 F7E3        <I>  mul   ebx..
148 000000C7 41          <I>  inc   ecx...
149 000000C8 EBE2        <I>  jmp   .multiplyLoop..
150                                     <I> ..
151                                     <I> .finished:
152 000000CA 83F900      <I>  cmp   ecx, 0..
153 000000CD 7407        <I>  je    .restore...
154 000000CF BB0A000000    <I>  mov   ebx, 10..
155 000000D4 F7F3        <I>  div   ebx.....
156                                     <I> ..
157                                     <I> .restore:
158 000000D6 5E          <I>  pop   esi...
159 000000D7 5A          <I>  pop   edx...
160 000000D8 59          <I>  pop   ecx..
161 000000D9 5B          <I>  pop   ebx.
162 000000DA C3          <I>  ret
163                                     <I> ..
164                                     <I> ..
165                                     <I> ;----- quit -----
166                                     <I> ; Функция завершения программы
167                                     <I> quit:
168 000000DB B800000000    <I>  mov   ebx, 0.....
169 000000E0 B801000000    <I>  mov   eax, 1.....
170 000000E5 CD80        <I>  int   80h
171 000000E7 C3          <I>  ret
2                                     section .data
3 00000000 D092D0B2D0B5D0B4D0-   msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-

```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Пере-ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.16: Содержимое файла листинга lab8-2.lst (ч.4)

```

3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00.....
4 00000013 D09D00B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
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 'Введите B: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
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 на 'fin',
41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'

```

Рис. 4.17: Содержимое файла листинга lab8-2.lst (ч.5)

```

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 на 'fin',
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 'Наибольшее число: '
47 00000163 A1[00000000] mov eax,[max]
48 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)'
49 0000016D E869FFFFFF call quit ; Выход

```

Рис. 4.18: Содержимое файла листинга lab8-2.lst (ч.6)

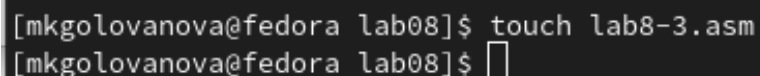
Я рассмотрела содержимое последних трёх строк файла листинга: 47 00000163 A1[00000000] mov eax,[max] 48 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)' 49 0000016D E869FFFFFF call quit ; Выход

47 - номер строки (не соответствует номеру строки в файле с исходным текстом программы), 00000163 - адрес строки (смещение машинного кода от начала текущего сегмента), A1[00000000] - машинный код (ассемблированная исходная

строка в виде шестнадцатеричной последовательности), mov eax,[max} - исходный текст программы; 48 - номер строки (не соответствует номеру строки в файле с исходным текстом программы), 00000168 - адрес строки, E819FFFFFF - машинный код, call iprintLF ; Вывод 'max(A,B,C)' - исходный текст программы; 49 - номер строки (не соответствует номеру строки в файле с исходным текстом программы), 0000016D - адрес строки, E869FFFFFF - машинный код, call quit ; Выход - исходный текст программы.

5 Задание для самостоятельной работы

Я создала файл lab8-3.asm (рис. 5.1) и написала в нём программу нахождения наименьшей из 3 целочисленных переменных a, b и c (рис. 5.2, рис. 5.3). Значения переменных выбрала из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы No 7(вариант 19).



```
[mkgolovanova@fedora lab08]$ touch lab8-3.asm  
[mkgolovanova@fedora lab08]$
```

Рис. 5.1: Создание файла lab8-3.asm

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '46'
C dd '74'
section .bss
min 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 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jb check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jb fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx

```

Рис. 5.2: Текст программы нахождения наименьшей из 3 целочисленных переменных a,b и c

```

jb fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 5.3: текст программы нахождения наименьшей из 3 целочисленных переменных a, b и c

Я создала исполняемый файл и проверила его работу (рис. 5.4).

```

[mkgolovanova@fedora lab08]$ nasm -f elf lab8-3.asm
[mkgolovanova@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[mkgolovanova@fedora lab08]$ ./lab8-3
Введите B: 32
Наименьшее число: 32
[mkgolovanova@fedora lab08]$

```

Рис. 5.4: Создание исполняемого файла lab8-3 и проверка его работы

6 Выводы

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