

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Студент: Верниковская Екатерина Андреевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	8
Реализация переходв в NASM	8
Изучение структуры файла листинга	18
Задание для самостоятельной работы	22
Выводы	30

Список таблиц

Список иллюстраций

1	Создание первого файла	8
2	Копирование файла «in_out.asm»	9
3	Ввод текста программы с инструкцией jmp	10
4	Создание исполняемого файла и его запуск	11
5	Изменение программы	12
6	Исполняемый файл + запуск	12
7	Изменение программы	14
8	Исполняемый файл + запуск	14
9	Создание файла «lab7-2.asm»	14
10	Ввод текста программы	17
11	Создание исполняемого файла и его запуск	18
12	Создание листинга	18
13	Открытый файл листинга	19
14	Удаление одного операнда	20
15	Получение файла листинга	21
16	Файл листинга	21
17	Создание файла «lab7-3.asm»	22
18	Ввод программы 1	25
19	Ввод программы 2	26
20	Исполняемый файл + запуск + проверка	26
21	Создание файла «lab7-4.asm»	27
22	Исполняемый файл + запуск + проверка	29

Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

Задание

1. Создать каталог для программ лабораторной работы №7 и в нём создать файл «lab7-1.asm».
2. Ввести в файл «lab7-1.asm» определённый текст программы с использованием инструкции jmp. Создать исполняемый файл и запустить его.
3. Изменить текст программы. Снова создать исполняемый файл и запустить его.
4. Опять изменить текст программы, чтобы строки выводились в нужном порядке, создать исполняемый файл и запустить его.
5. Создать файл «lab7-2.asm» и ввести в него определённый текст. Создать исполняемый файл и запустить его.
6. Создать файл листинга для программы из файла «lab7-2.asm». Открыть его с помощью любого текстового редактора. Подробно объяснить содержимое трёх строк файла листинга по выбору.
7. Открыть файл «lab7-2.asm» и в любой инструкции двумя операндами удалить один операнд. Ответить на поставленные вопросы.
8. Напишите программу, которая находит из 3 целочисленных переменных наименьшее. Значения переменных брать из нужной таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы №6 (У меня 17 вариант).
9. Написать программу, которая для введённых значений x и a вычисляет значение заданной функции $F(x)$. Вид функции и значения брать из определённой таблицы, в соответствии с полученным вариантом (В нашем случае это 17

вариант).

Выполнение лабораторной работы

Реализация переходов в NASM

В созданном каталоге «~/work/arch-pc/lab07» создаём файл «lab7-1.asm» (рис. [-@fig:001])

```
eavernikovskaya@dk6n51 ~ $ mkdir ~/work/arch-pc/lab07
eavernikovskaya@dk6n51 ~ $ cd ~/work/arch-pc/lab07
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ touch lab7-1.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ls
lab7-1.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $
```

Рис. 1: Создание первого файла

Копируем из каталога «~/work/arch-pc/lab06» файл «in_out.asm» (рис. [-@fig:002])

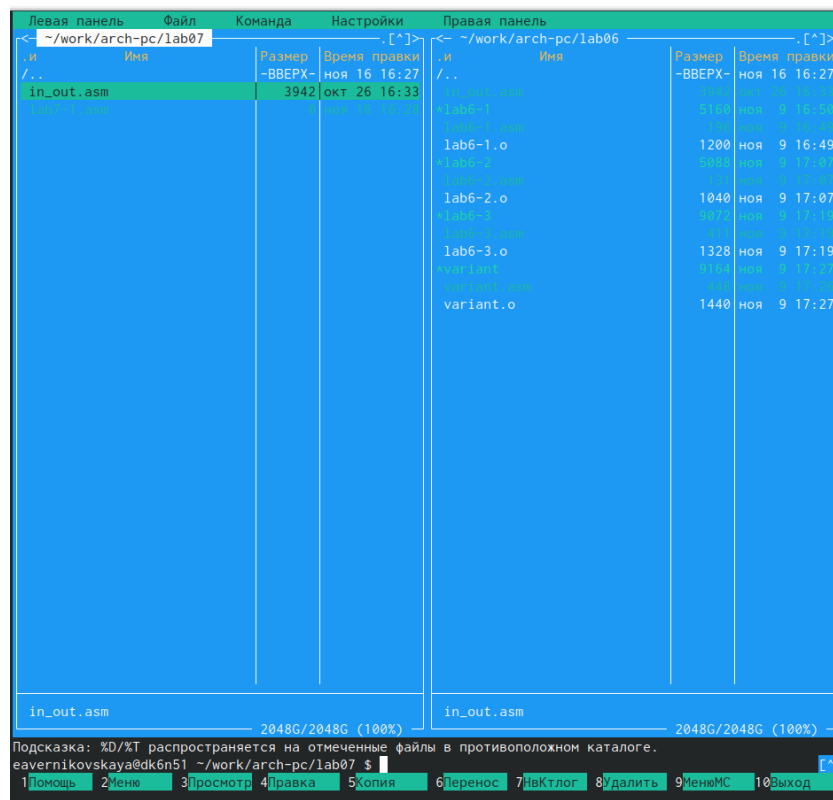


Рис. 2: Копирование файла «in_out.asm»

Вводим нужный текст программы с использованием инструкции jmp (рис. [-@fig:003])

Текст программы:

```
%include 'in_out.asm'

SECTION .data

msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text

GLOBAL _start

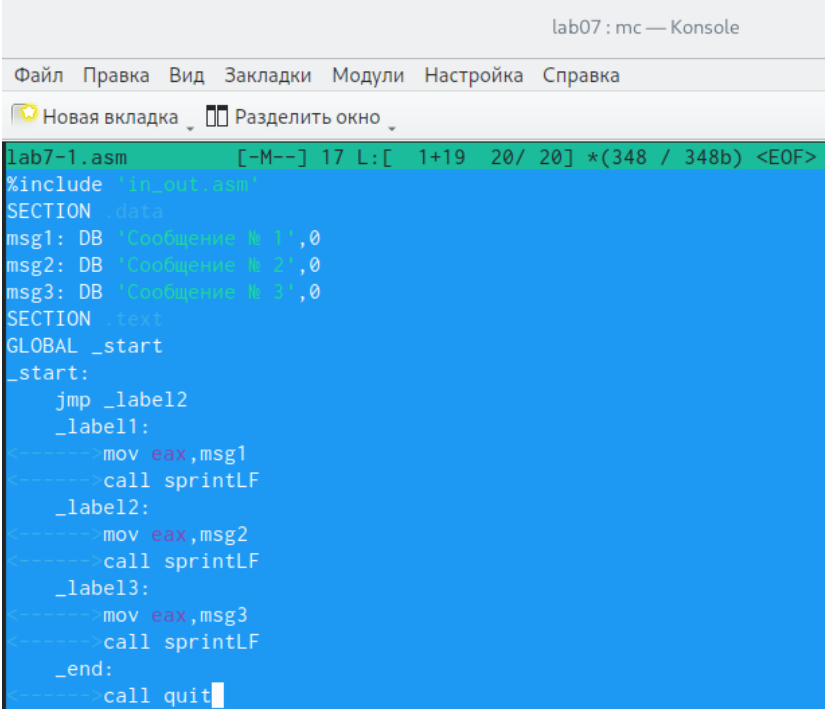
_start:

    jmp _label2
```

```

_label1:
    mov eax,msg1
    call sprintLF
_label2:
    mov eax,msg2
    call sprintLF
_label3:
    mov eax,msg3
    call sprintLF
_end:
    call quit

```



```

lab7-1.asm  [-M--] 17 L: [ 1+19 20/ 20] *(348 / 348b) <EOF>
#include 'in_out.asm'
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
    call sprintLF
_label2:
    mov eax,msg2
    call sprintLF
_label3:
    mov eax,msg3
    call sprintLF
_end:
    call quit

```

Рис. 3: Ввод текста программы с инструкцией jmp

Создаём исполняемый файл и запускаем его (рис. [-@fig:004])

```
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ █
```

Рис. 4: Создание исполняемого файла и его запуск

Изменяем текст программы так, чтобы она выводила сначала ‘Сообщение № 2’ а потом ‘Сообщение № 1’ и завершала работу (рис. [-@fig:005])

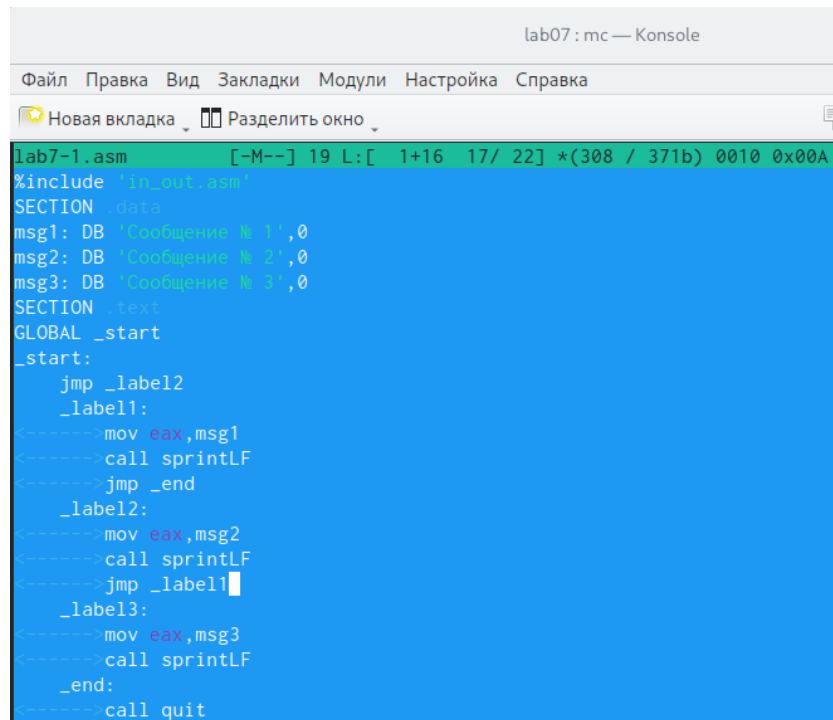
Изменённый текст программы:

```
%include 'in_out.asm'
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
    call sprintLF
    jmp _end
_label2:
    mov eax,msg2
    call sprintLF
    jmp _label1
_label3:
    mov eax,msg3
    call sprintLF
```

```

_end:
    call quit

```



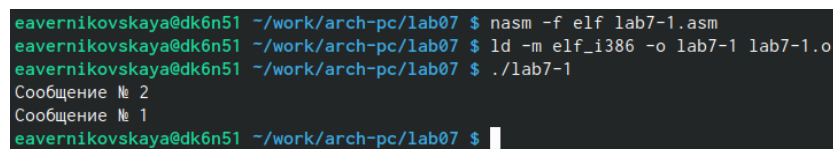
```

lab7-1.asm  [-M--] 19 L:[ 1+16 17/ 22] *(308 / 371b) 0010 0x00A
#include 'in_out.asm'
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
    call sprintf
    jmp _end
_label2:
    mov eax,msg2
    call sprintf
    jmp _label1
_label3:
    mov eax,msg3
    call sprintf
_end:
    call quit

```

Рис. 5: Изменение программы

Снова создаём исполняемый файл и запускаем его (рис. [-@fig:006])



```

eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ 

```

Рис. 6: Исполняемый файл + запуск

Изменяем текст программы так, чтобы она выводила сообщения в обратном порядке, т.е. сначала 3, потом 2, потом 1 и завершала работу (рис. [-@fig:007])

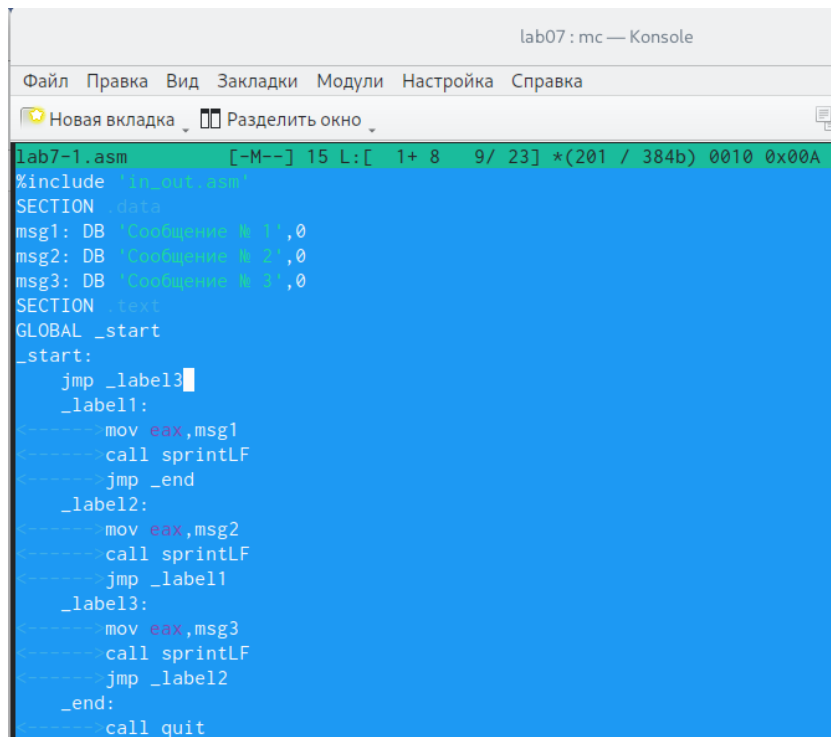
Изменённый текст программы:

```

#include 'in_out.asm'
SECTION .data

```

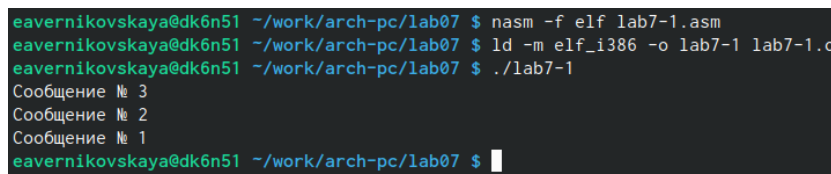
```
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
    jmp _label3
_label1:
    mov eax,msg1
    call sprintLF
    jmp _end
_label2:
    mov eax,msg2
    call sprintLF
    jmp _label1
_label3:
    mov eax,msg3
    call sprintLF
    jmp _label2
_end:
    call quit
```



```
lab7-1.asm [-M--] 15 L: [ 1+ 8 9/ 23] *(201 / 384b) 0010 0x00A
#include "in_out.asm"
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
    call sprintf
    jmp _end
_label2:
    mov eax,msg2
    call sprintf
    jmp _label1
_label3:
    mov eax,msg3
    call sprintf
    jmp _label2
_end:
    call quit
```

Рис. 7: Изменение программы

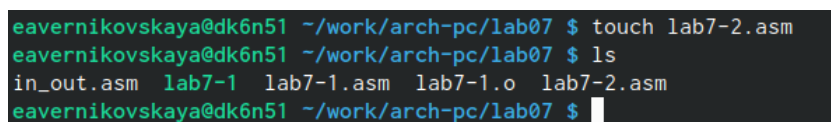
Опять создаём исполняемый файл и запускаем его (рис. [-@fig:008])



```
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $
```

Рис. 8: Исполняемый файл + запуск

Создаём файл «lab7-2.asm» (рис. [-@fig:009])



```
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ touch lab7-2.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $
```

Рис. 9: Создание файла «lab7-2.asm»

Вводим текст программы, которая определяет и выводит на экран наибольшее

число из 3 целочисленных переменных A, B, C (рис. [-@fig:010])

Текст программы:

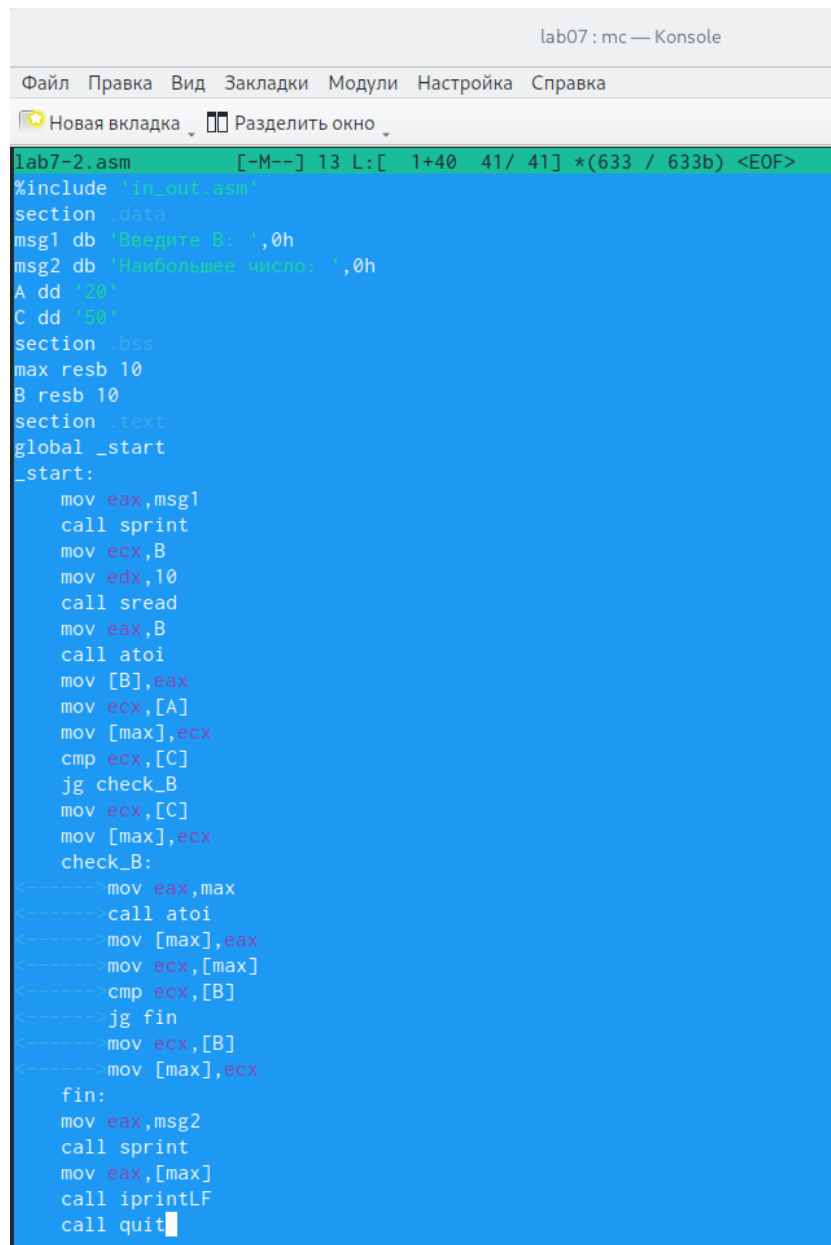
```
%include 'in_out.asm'

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:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx
    check_B:
```

```
    mov eax,max
    call atoi
    mov [max],eax
    mov ecx,[max]
    cmp ecx[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
fin:
    mov eax,msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

```
lab07: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно
lab7-2.asm  [-M--] 13 L: [ 1+40 41/ 41] *(633 / 633b) <EOF>
#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:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx
check_B:
    ----- mov eax,max
    ----- call atoi
    ----- mov [max],eax
    ----- mov ecx,[max]
    ----- cmp ecx,[B]
    ----- jg fin
    ----- mov ecx,[B]
    ----- mov [max],ecx
    fin:
    mov eax,msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

Рис. 10: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. [-@fig:011])

```

eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 3
Наибольшее число: 50
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 60
Наибольшее число: 60
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 30
Наибольшее число: 50
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $

```

Рис. 11: Создание исполняемого файла и его запуск

Изучение структуры файла листинга

Создаем файл листинга для программы из файла «lab7-2.asm» (рис. [-@fig:012])

```

eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-2.o
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $

```

Рис. 12: Создание листинга

Открываем файл листинга с помощью текстового редактора (рис. [-@fig:013])

```

lab7-2.lst  [----] 59 L: [166+41 207/217] *(12539/13028b) 0010 0x00A [*][X]
165                                     <I> ; Функция завершения программы
166                                     <I> quit:
167 000000DB B00000000 <I> mov ebx, 0
168 000000E0 B01000000 <I> mov eax, 1
169 000000E5 CD80 <I> int 80h
170 000000E7 C3 <I> ret
2
3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00
4 00000013 D09DD0B0D0B8D0B1D0- 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
8 00000000 <res Ah> section .bss
9 0000000A <res Ah> max resb 10
10
11 section .text
12 global _start
13 000000E8 B8[00000000] mov eax,msg1
14 000000ED E81DFFFFFF call sprint
15 000000F2 B9[0A000000] mov ecx,B
16 000000F7 BA0A000000 mov edx,10
17 000000FC E842FFFFFF call spread
18 00000101 B8[0A000000] mov eax,B
19 00000106 E891FFFFFF call atoi
20 0000010B A3[0A000000] mov [B],eax
21 00000110 8B0D[35000000] mov ecx,[A]
22 00000116 890D[00000000] mov [max],ecx
23 0000011C 3B0D[39000000] cmp ecx,[C]
24 00000122 7F0C jg check_B
25 00000124 8B0D[39000000] mov ecx,[C]
26 0000012A 890D[00000000] mov [max],ecx
27 check_B:
28 00000130 B8[00000000] mov eax,max
29 00000135 E862FFFFFF call atoi
30 0000013A A3[00000000] mov [max],eax
31 0000013F 8B0D[00000000] mov ecx,[max]
32 00000145 3B0D[0A000000] cmp ecx,[B]

```

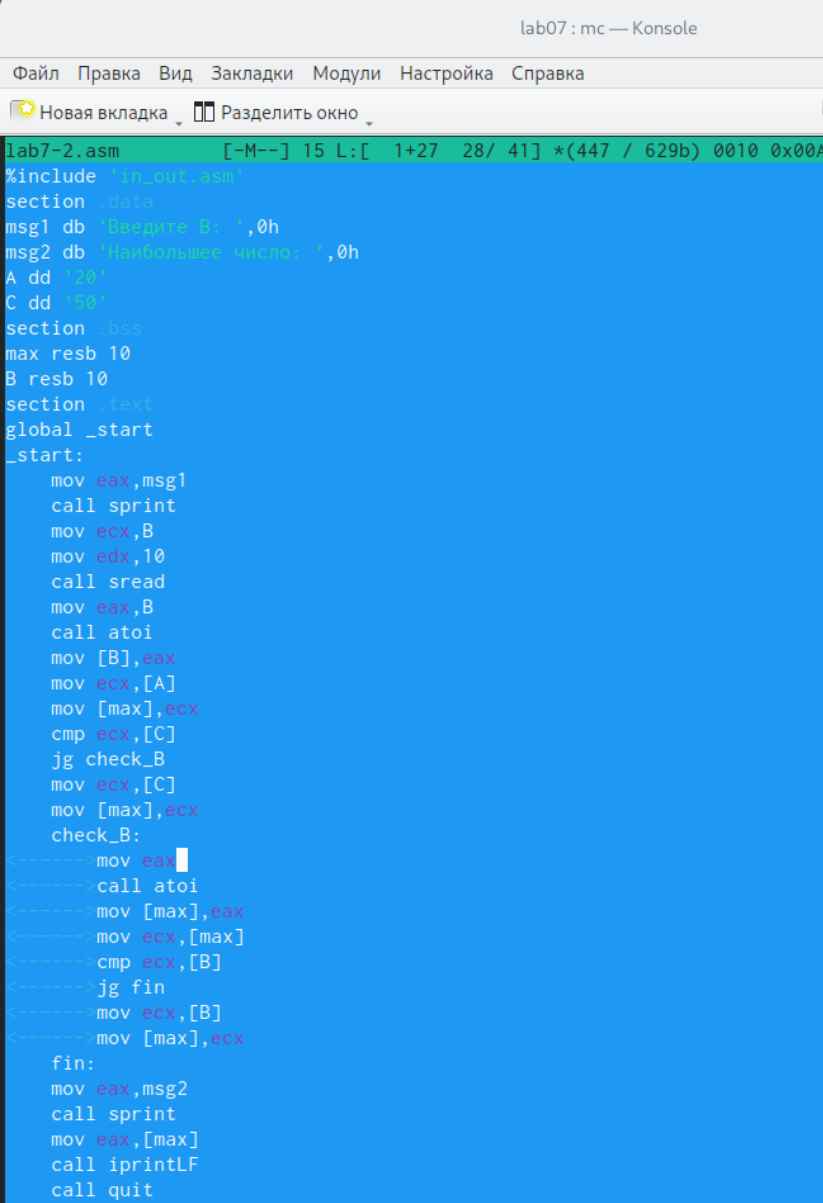
Рис. 13: Открытый файл листинга

Опишем строки под номерами 13, 14 и 15.

- 1) 13 (номер строки); 000000E8 (адрес, начинается по смещению 000000E8 в сегменте кода); B8[00000000] (машинный код); mov eax,msg1 (исходный текст программы, в котором мы перемещаем адрес метки msg1 в регистр eax. msg1 - адрес строки, которую мы хотим вывести)
- 2) 14 (номер строки); 000000ED (адрес, начинается по смещению 000000ED в сегменте кода); E81DFFFFFF (машинный код); call sprint (исходный текст программы, в котором мы вызываем подпрограмму печати сообщения)
- 3) 15 (номер строки); 000000F2 (адрес, начинается по смещению 000000F2 в сегменте кода); B9[0A000000] (машинный код); mov ecx,B (исходный текст программы, в котором мы перемещаем адрес метки B в регистр ecx)

программы, в котором мы записываем адрес введенной переменной в есх)

Снова открываем файл «lab7-2.asm» и в любой строке с двумя операндами удаляем один операнд. В 28 строке удаляем операнд ‘max’ (рис. [-@fig:014])



```
lab07: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно
lab7-2.asm  [-M--]  15  L: [ 1+27  28/ 41]  *(447 / 629b)  0010  0x00A
#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:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx
    check_B:
    ----- mov eax,
    ----- call atoi
    ----- mov [max],eax
    ----- mov ecx,[max]
    ----- cmp ecx,[B]
    ----- jg fin
    ----- mov ecx,[B]
    ----- mov [max],ecx
    fin:
    mov eax,msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

Рис. 14: Удаление одного операнда

Выполняем трансляцию с получением файла листинга (рис. [-@fig:015])

```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$
```

Рис. 15: Получение файла листинга

В ходе трансляции система выдаёт ошибку и создаёт файлы «lab7-2» и «lab7-2.lst». Заходим в листинг для изучения того, что добавилось в него после возникновения ошибки (рис. [-@fig:016])

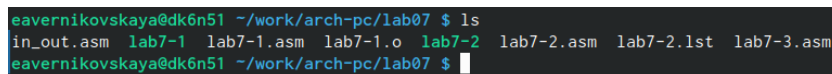
```
lab07: mcedit — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
/home/eavernikovskaya/work/arch-pc/lab07/lab7-2.lst  [----]  0 L:[166+38 204/218] *(12322/13115b) 0032 0x020 [*][X]
166                                     <i> ; Функция завершения программы
167                                     <i> quit:
168 000000D8 B800000000               <i> mov     ebx, 0
169 000000E0 B801000000               <i> mov     eax, 1
170 000000E5 CD00                     <i> int     80h
171 000000E7 C3                       <i> ret
2                                     section .data
3 00000000 D092D0B2D0B5D0B4D0-      msg1 db 'Введите B: ',0h
3 00000000 B8D182D0B520423A20-
3 00000012 00
4 00000013 D09D0D0B0D0B0D0B1D0-      msg2 db 'Наибольшее число: ',0h
4 0000001C BCD0B8D18CD18D0D0B5-
4 00000025 D0B52D0D187D0B8D0181-
4 0000002E D0B8D0B8E3A2000
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 000000E8 B8[00000000]               mov     eax,msg1
14 000000ED E81DFFFFFF               call    sprint
15 000000F2 B9[0A000000]               mov     ecx,B
16 000000F7 BA0A000000               mov     edx,10
17 000000FC E842FFFFFF               call    spread
18 00000101 B8[0A000000]               mov     eax,B
19 00000106 E891FFFFFF               call    atoi
20 0000010B A3[0A000000]               mov     [B],eax
21 00000110 8B0D[35000000]             mov     ecx,[A]
22 00000116 890D[00000000]             mov     [max],ecx
23 0000011C 3B0D[39000000]             cmp     ecx,[C]
24 00000122 7F0C                       jg      check_B
25 00000124 8B0D[39000000]             mov     ecx,[C]
26 0000012A 890D[00000000]             mov     [max],ecx
27                                     check_B:
28                                     <mov     eax,
28                                     ***** error: invalid combination of opcode and operands
29 00000130 E867FFFFFF               <mov     call    atoi
30 00000135 A3[00000000]               <mov     mov     [max],eax
31 0000013A 8B0D[00000000]             <mov     mov     ecx,[max]
32 00000140 3B0D[0A000000]             <cmp     cmp     ecx,[B]
33 00000146 7F0C                       <jg      jg      fin
34 00000148 8B0D[0A000000]             <mov     mov     ecx,[B]
35 0000014E 890D[00000000]             <mov     mov     [max],ecx
36                                     <fin:
37 00000154 B8[13000000]               mov     eax,msg2
38 00000159 E8B1FFFFFF               call    sprint
39 0000015E A1[00000000]               mov     eax,[max]
40 00000163 E81EFFFFFF               call    iprintLF
41 00000168 E86EFFFFFF               call    quit
1 Помощь  2 Сохранить  3 Блок  4 Замена  5 Копия  6 Пере-тить  7 Поиск  8 Удалить  9 Уменьш  10 Выход
```

Рис. 16: Файл листинга

В 28 строке выводится сообщение об ошибке

Задание для самостоятельной работы

Создаём файл «lab7-3.asm» (рис. [-@fig:017])



```
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-3.asm
eavernikovskaya@dk6n51 ~/work/arch-pc/lab07 $
```

Рис. 17: Создание файла «lab7-3.asm»

Вводим текст программы, которая находит наименьшее число из 3 целочисленных введённых переменных A, B и C (рис. [-@fig:018]) и (рис. [-@fig:019])

Текст программы:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите A: ',0h
msg2 db 'Введите B: ',0h
msg3 db 'Введите C: ',0h
msg4 db 'Наименьшее число: ',0h

section .bss
min resb 10
B resb 10
A resb 10
C resb 10

section .text
global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,A
    mov edx,10
    call sread
```

```
mov eax,A
call atoi
mov [A],eax
```

```
mov eax,msg2
call sprint
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi
mov [B],eax
```

```
mov eax,msg3
call sprint
mov ecx,C
mov edx,10
call sread
mov eax,C
call atoi
mov [C],eax
```

```
mov ecx,[A]
mov [min],ecx
```

```
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx
```

```
check_B:
mov ecx,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx
fin:
mov eax,msg4
call sprint
mov eax,[min]
call iprintLF
call quit
```



```
lab7-3.asm [----] 13 L: [ 1+ 1 2/ 61] *(35 / 923b) 0010 0x00A
#include "in_out.asm"
section .data
msg1 db "Введите A: ",0h
msg2 db "Введите B: ",0h
msg3 db "Введите C: ",0h
msg4 db "Наименьшее число: ",0h
section .bss
min resb 10
B resb 10
A resb 10
C resb 10
section .text
global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,A
    mov edx,10
    call sread
    mov eax,A
    call atoi
    mov [A],eax

    mov eax,msg2
    call sprint
    mov ecx,B
    mov edx,10
    call sread
    mov eax,B
    call atoi
    mov [B],eax

    mov eax,msg3
    call sprint
    mov ecx,C
    mov edx,10
    call sread
    mov eax,C
    call atoi
    mov [C],eax
```

Рис. 18: Ввод программы 1

```

mov ecx,[A]
mov [min],ecx
...
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx
check_B:
<----->mov ecx,[min]
<----->cmp ecx,[B]
<----->jl fin
<----->mov ecx,[B]
<----->mov [min],ecx
fin:
mov eax,msg4
call sprint
mov eax,[min]
call iprintLF
call quit

```

Рис. 19: Ввод программы 2

Создаём исполняемый файл и запускаем его. Проверяем работу программы для введённых значений ($A = 26$; $B = 12$; $C = 68$) из нужной таблицы (рис. [-@fig:020])

```

eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ gedit lab7-3.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 26
Введите B: 12
Введите C: 68
Наименьшее число: 12
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 3
Введите B: 67
Введите C: 23
Наименьшее число: 3
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 45
Введите B: 32
Введите C: 12
Наименьшее число: 12
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$

```

Рис. 20: Исполняемый файл + запуск + проверка

Создаём файл «lab7-4.asm» (рис. [-@fig:021])

```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ touch lab7-4.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-3 lab7-3.asm lab7-3.o lab7-4.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$
```

Рис. 21: Создание файла «lab7-4.asm»

Вводим текст программы. Программа вычисляет значение функции ($F(x)$: $a+8$, если $a < 8$ и $a*x$, если $a \geq 8$) и выводит результат вычислений (рис. [-@fig:022])

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Введите x: ',0h
msg2: DB 'Введите a: ',0h
otv: 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 edx,8
cmp edx,[a]
jg check_1
    mov eax,[a]
    mov ebx,[x]
    mul ebx
    mov [res],eax
    jmp fin
check_1:

    mov eax,[a]
    add eax,8
    mov [res],eax
    jmp fin

fin:
    mov eax,otv
    call sprint
    mov eax,[res]
    call iprintLF
    call quit

```

Создаём исполняемый файл и запускаем его. Проверяем работу программы для

введённых значений ($x_1=3$, $a_1=4$ и $x_2=2$, $a_2=9$) из нужной таблицы (рис. [-@fig:023])

```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 4
F(x) = 12
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 9
F(x) = 18
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab07$
```

Рис. 22: Исполняемый файл + запуск + проверка

Выводы

В ходе выполнения лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов. Также мы познакомились с назначением и структурой файла листинга.