

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Иван Салиндер

Содержание

1	Цель работы	5
2	Задание	6
2.1	1. Команды условного перехода	6
2.2	2. Реализация переходов в NASM	6
2.3	3. Изучение структуры файлы листинга	6
2.4	4. Самостоятельная работа	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Самостоятельная работа	20
6	Выводы	29

Список иллюстраций

4.1	Создание директории	8
4.2	Создание копии файла для дальнейшей работы, редактирование файла	9
4.3	Запуск исполняемого файла	9
4.4	Редактирование программы	10
4.5	Создание исполняемого файла	11
4.6	Создание файла	11
4.7	Вставляю текст в файл	12
4.8	Вставляю текст в файл	13
4.9	Запуск исполняемого файла	14
4.10	Запуск исполняемого файла	14
4.11	Файл листинга	15
4.12	Файл листинга	17
4.13	asm -f elf -l lab7-2.lst lab7-2.asm	17
4.14	gedit lab7-2.lst	18
4.15	18
4.16	19
5.1	Создание запуск файла	20
5.2	Редактирование файла	21
5.3	Запуск исполняемого файла	21
5.4	создание файла	24
5.5	ввод программы в файл	25
5.6	Создание исполняемого файла	26
5.7	запуск исполняемого файла	26

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

2.1 1. Команды условного перехода

2.2 2. Реализация переходов в NASM

2.3 3. Изучение структуры файлы листинга

2.4 4. Самостоятельная работа

3 Теоретическое введение

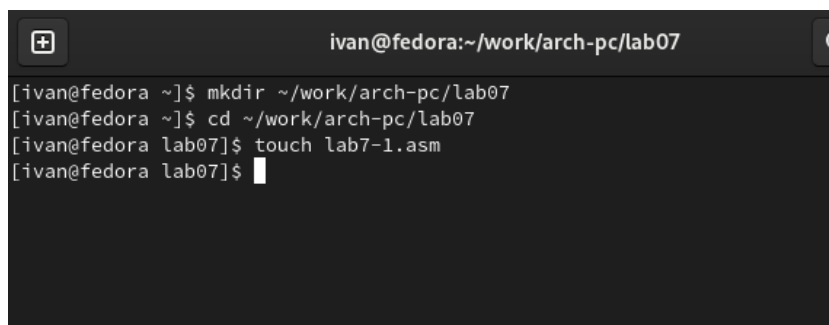
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

1

С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [4.1]).

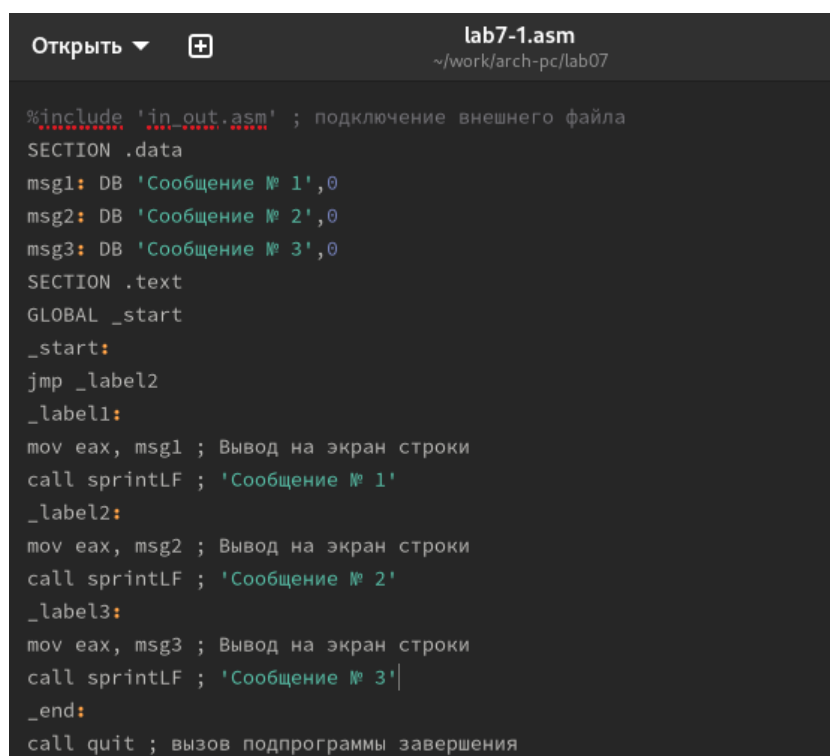


```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora ~]$ mkdir ~/work/arch-pc/lab07
[ivan@fedora ~]$ cd ~/work/arch-pc/lab07
[ivan@fedora lab07]$ touch lab7-1.asm
[ivan@fedora lab07]$
```

Рис. 4.1: Создание директории

2

Копирую в текущий каталог файл `in_out.asm` из загрузок, т.к. он будет использоваться в других программах. Открываю созданный файл `lab7-1.asm`, вставляю в него программу реализации безусловных переходов(рис. [fig002?]).




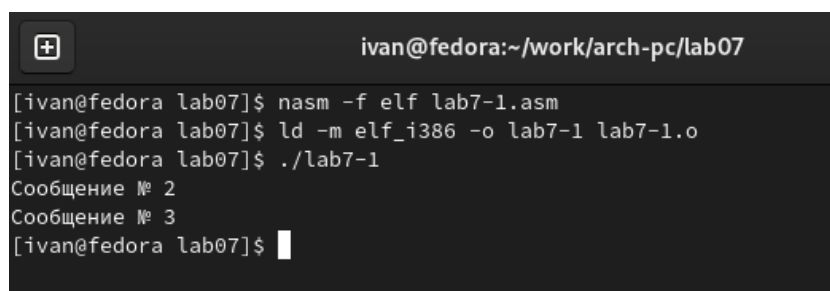
```
Открыть ▾  lab7-1.asm  
~/work/arch-pc/lab07  
  
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
_start:  
jmp _label2  
_label1:  
mov eax, msg1 ; Вывод на 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
```

Рис. 4.2: Создание копии файла для дальнейшей работы, редактирование файла

3

Создаю исполняемый файл программы и запускаю его (рис. [4.3]). Инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`.

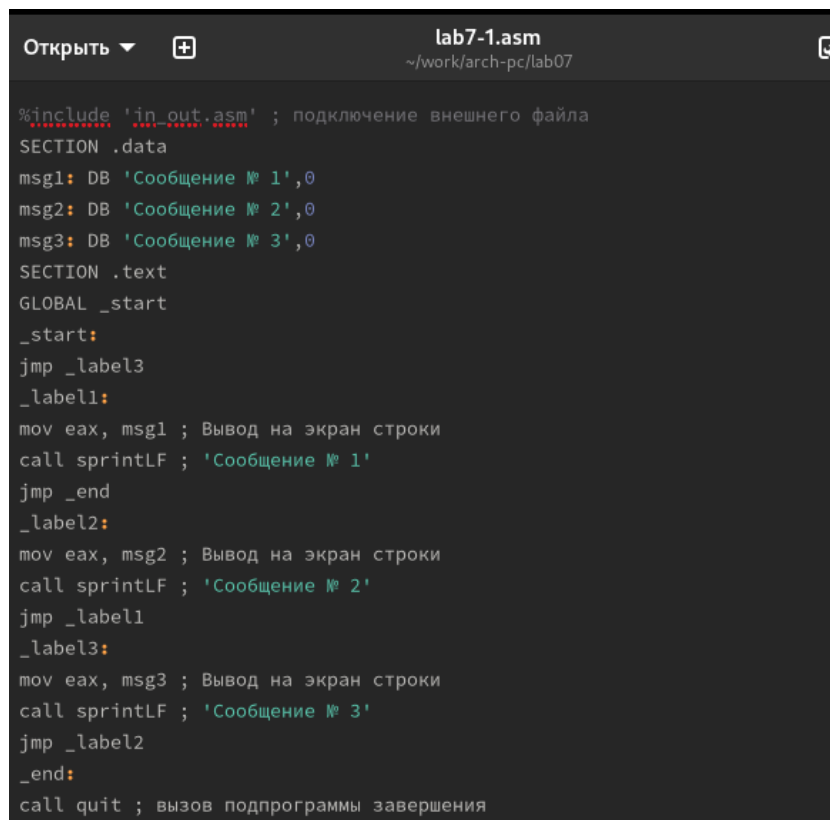


```
ivan@fedora:~/work/arch-pc/lab07  
[ivan@fedora lab07]$ nasm -f elf lab7-1.asm  
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o  
[ivan@fedora lab07]$ ./lab7-1  
Сообщение № 2  
Сообщение № 3  
[ivan@fedora lab07]$
```

Рис. 4.3: Запуск исполняемого файла

4

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [4.4]).




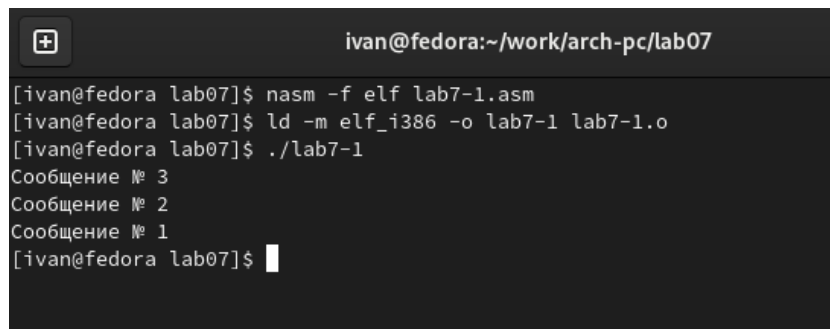
```
Открыть ▾  lab7-1.asm  
~/work/arch-pc/lab07  
  
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
_start:  
jmp _label3  
_label1:  
mov eax, msg1 ; Вывод на 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
```

Рис. 4.4: Редактирование программы

5

Создаю исполняемый файл и проверяю работу программы (рис. [4.5]). Программа отработало верно.

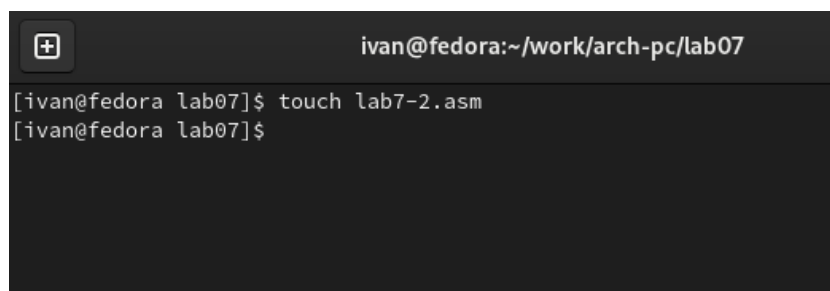
A terminal window with a dark background. The title bar shows a window icon and the text "ivan@fedora:~/work/arch-pc/lab07". The terminal content shows a series of commands and their outputs: "nasm -f elf lab7-1.asm", "ld -m elf_i386 -o lab7-1 lab7-1.o", and ". /lab7-1". Following these are three lines of Russian text: "Сообщение № 3", "Сообщение № 2", and "Сообщение № 1". The prompt "[ivan@fedora lab07]" is followed by a cursor.

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

Рис. 4.5: Создание исполняемого файла

6

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис. [4.6]).

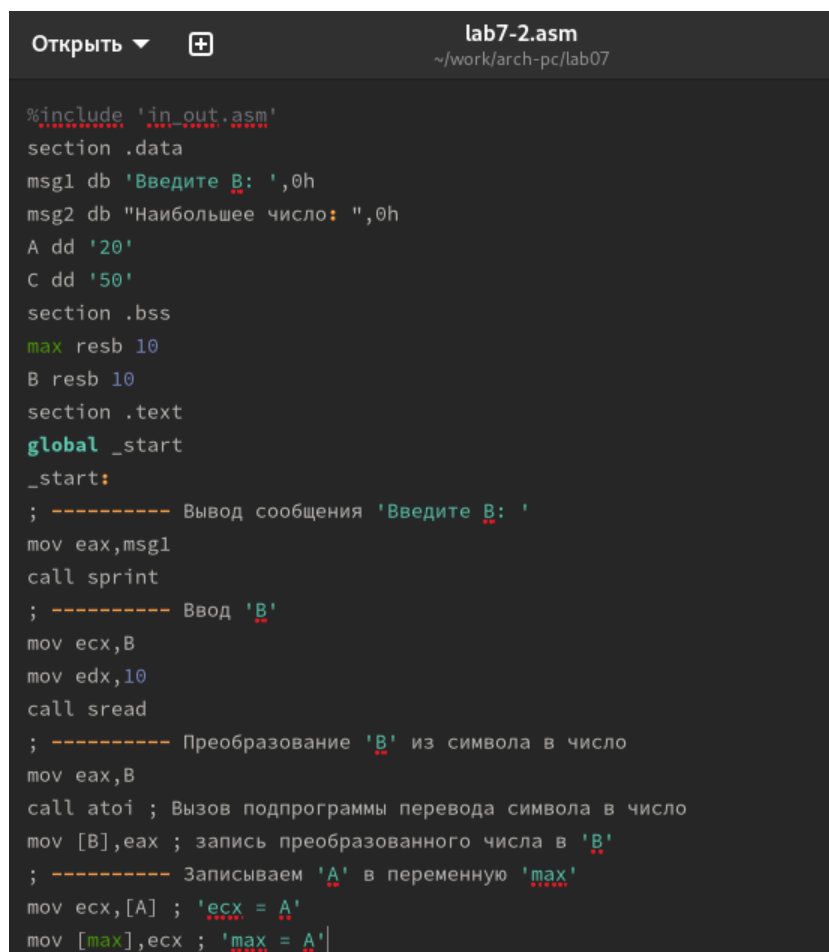
A terminal window with a dark background. The title bar shows a window icon and the text "ivan@fedora:~/work/arch-pc/lab07". The terminal content shows the command "touch lab7-2.asm" being executed. The prompt "[ivan@fedora lab07]" is followed by a cursor.

```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ touch lab7-2.asm
[ivan@fedora lab07]$
```

Рис. 4.6: Создание файла

7

Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[4.8]).




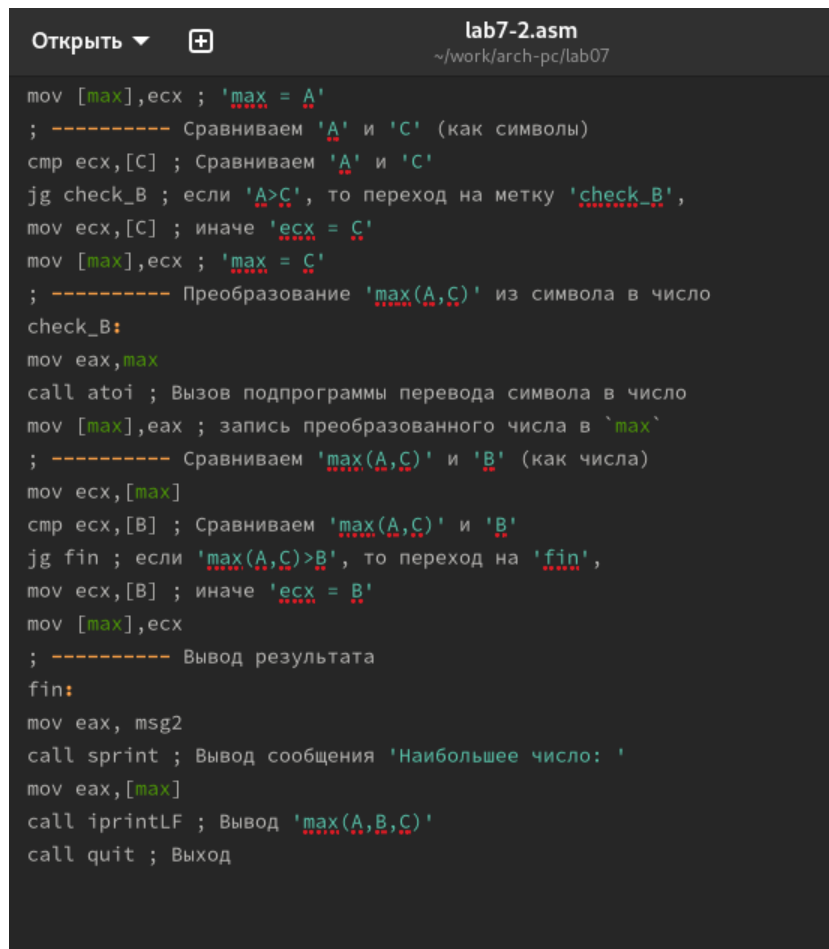
```
Открыть ▾  lab7-2.asm  
~/work/arch-pc/lab07  
  
%include 'in_out.asm'  
section .data  
msg1 db 'Введите B: ',0h  
msg2 db "Наибольшее число: ",0h  
A dd '20'  
C dd '50'  
section .bss  
max resb 10  
B resb 10  
section .text  
global _start  
_start:  
; ----- Вывод сообщения 'Введите B: '  
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'
```

Рис. 4.7: Вставляю текст в файл



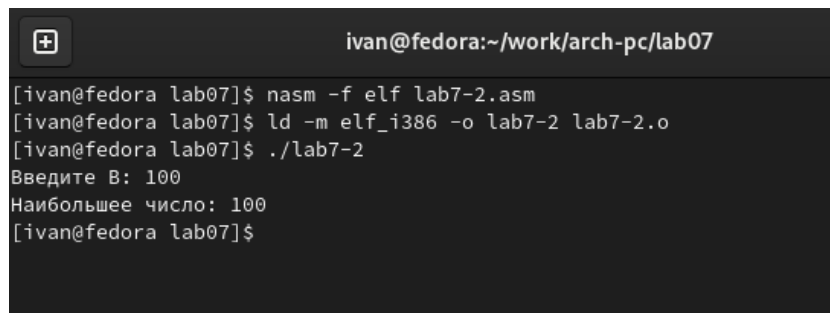
```
Открыть ▾ + lab7-2.asm
~/work/arch-pc/lab07

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`
; ----- 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 sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 4.8: Вставляю текст в файл

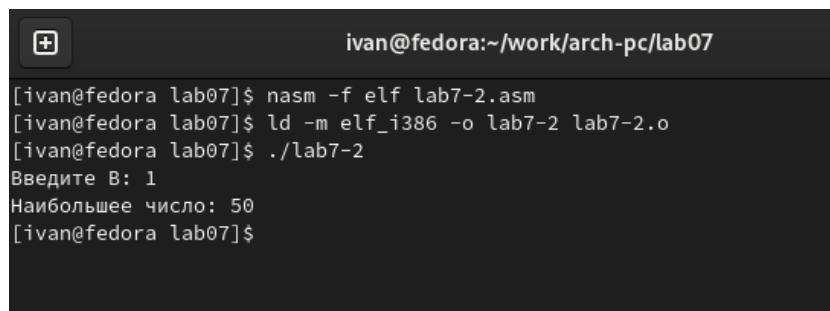
8

Создаю и запускаю новый исполняемый файл, проверяю работу программы (рис. [4.10]).



```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ nasm -f elf lab7-2.asm
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ivan@fedora lab07]$ ./lab7-2
Введите В: 100
Наибольшее число: 100
[ivan@fedora lab07]$
```

Рис. 4.9: Запуск исполняемого файла



```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ nasm -f elf lab7-2.asm
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ivan@fedora lab07]$ ./lab7-2
Введите В: 1
Наибольшее число: 50
[ivan@fedora lab07]$
```

Рис. 4.10: Запуск исполняемого файла

9

Открываю файл листинга с помощью редактора `msedit`. Рассмотрим 9-11 строки: (рис. [4.11]).

```

ivan@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/dayanchberdyev/work/arch-pc/lab07/lab7-2.asm
#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'
[ Прочитано 49 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

```

Рис. 4.11: Файл листинга

9 строка:

- Первые цифры [9] - это номер строки файла листинга.
- Следующие цифры [00000006] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [7403] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латинского алфавита.
- следующее [jz finished] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями.

10 строка:

- Первые цифры [10] - это номер строки файла листинга.

- Следующие цифры [00000008] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [40] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [inc eax] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

11 строка:

- Первые цифры [11] - это номер строки файла листинга.
- Следующие цифры [00000009] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [EBF8] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [jmp nextchar] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

10

Открываю файл листинга с помощью редактора mcedit и замечаю, что в файле листинга появляется ошибка. (рис. [4.12]).


```

ivan@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/dayanchberdyev/work/arch-pc/lab07/lab7-2.asm
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'
; ----- 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

```

[^]G Справка [^]O Записать [^]W Поиск [^]K Вырезать [^]T Выполнить [^]S Позиция
[^]X Выход [^]R ЧитФайл [^]\ Замена [^]U Вставить [^]J Выводить [^]/ К строке

Рис. 4.12: Файл листинга

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга

11

Создал файл листинга для программы из файла lab7-2.asm (рис. [4.13]).

```

ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[ivan@fedora lab07]$

```

Рис. 4.13: `asm -f elf -l lab7-2.lst lab7-2.asm`

12

Открыл файл листинга lab7-2.lst с помощью любого текстового редактора,

например `gedit:{#fig:012 width=70%}`

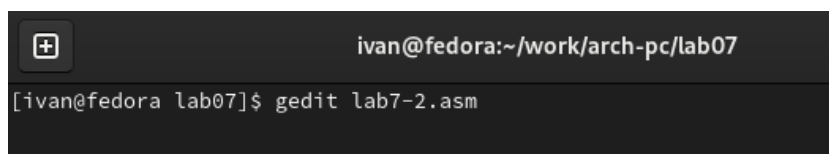


Рис. 4.14: gedit lab7-2.lst

13

Открыл файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга:`{#fig:013 width=70%}`

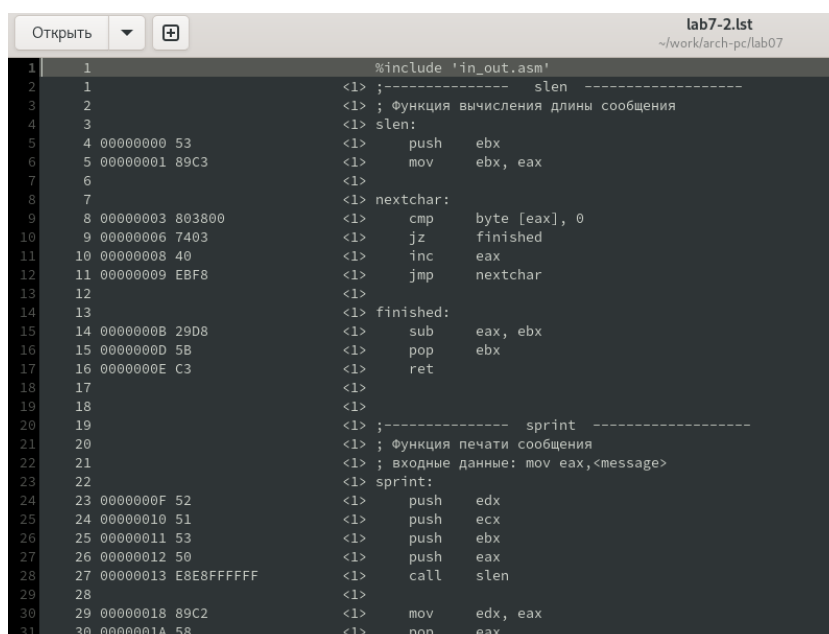


Рис. 4.15:

Открыть

lab7-2.lst

~/work/arch-pc/lab07

```

195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
206 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
211 36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
212 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000] mov ecx,[max]
214 39 00000145 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
215 40 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
216 41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:
220 45 00000159 B8[13000000] mov eax,msg2
221 46 0000015E E8ACFFFFFF call sprintf ; Вывод сообщения 'Наибольшее число: '
222 47 00000163 A1[00000000] mov eax,[max]
223 48 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)'
224 49 0000016D E869FFFFFF call quit ; Выход
225 50

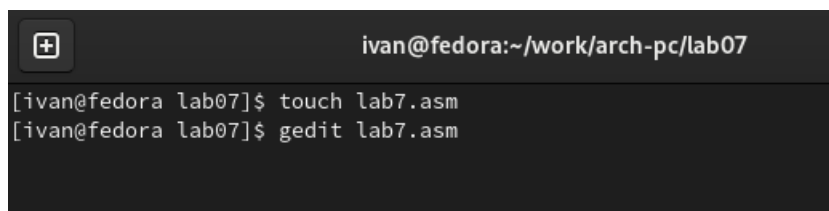
```

Рис. 4.16:

5 Самостоятельная работа

1

Создаю файл lab7.asm с помощью утилиты touch и запускаю редактора gedit (рис. [5.1]).

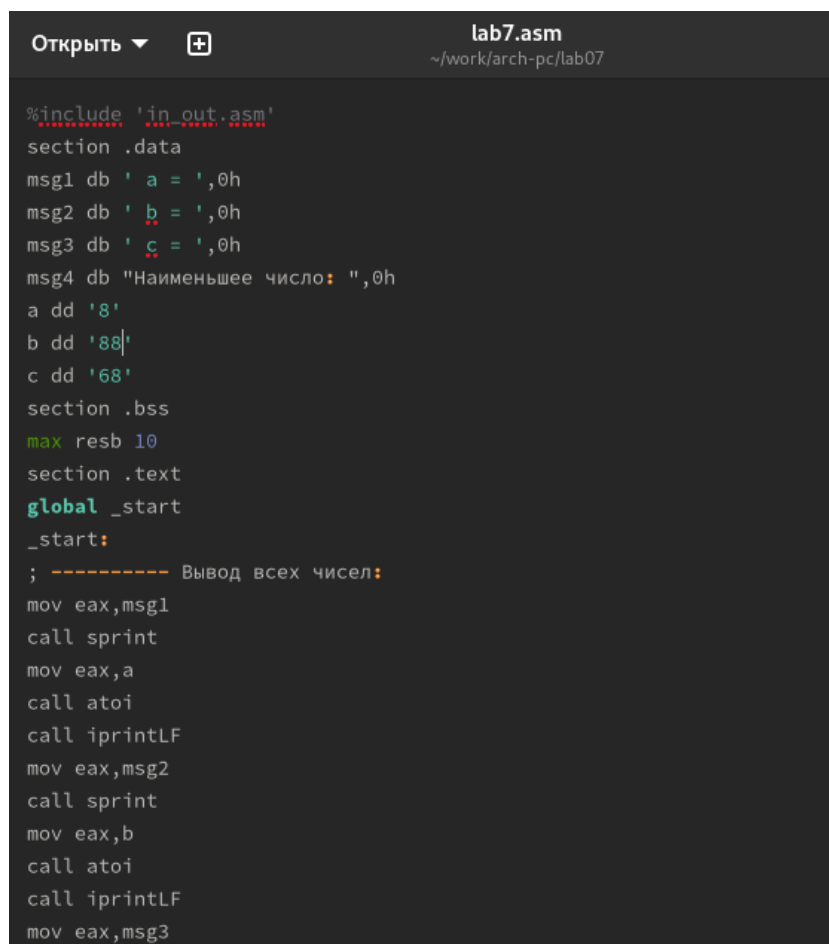
A screenshot of a terminal window with a dark background. The title bar at the top shows a window icon on the left and the text 'ivan@fedora:~/work/arch-pc/lab07' on the right. The terminal contains two lines of text: the first line is '[ivan@fedora lab07]\$ touch lab7.asm' and the second line is '[ivan@fedora lab07]\$ gedit lab7.asm'.

```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ touch lab7.asm
[ivan@fedora lab07]$ gedit lab7.asm
```

Рис. 5.1: Создание запуск файла

2

Ввожу в созданный файл текст программы для вычисления наименьшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 4 вариант (рис. [5.2]).



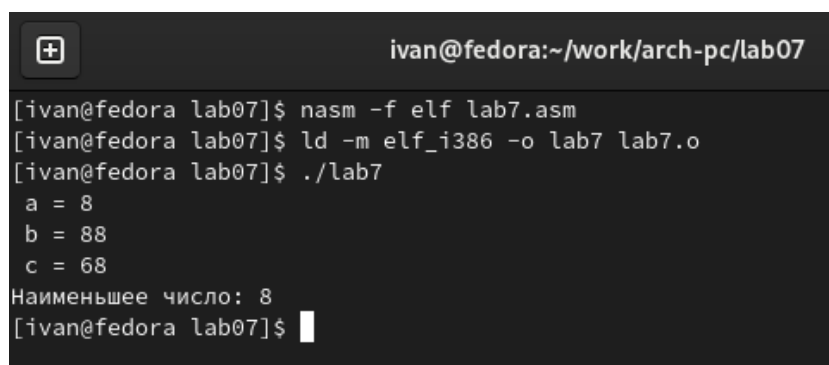
```
Открыть ▾ + lab7.asm
~/work/arch-pc/lab07

%include 'in_out.asm'
section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '8'
b dd '88'
c dd '68'
section .bss
max resb 10
section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF
mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF
mov eax,msg3
```

Рис. 5.2: Редактирование файла

3

Создаю исполняемый файл и запускаю его (рис. [5.3]).



```
ivan@fedora:~/work/arch-pc/lab07

[ivan@fedora lab07]$ nasm -f elf lab7.asm
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7 lab7.o
[ivan@fedora lab07]$ ./lab7
a = 8
b = 88
c = 68
Наименьшее число: 8
[ivan@fedora lab07]$
```

Рис. 5.3: Запуск исполняемого файла

Текст программы

```
%include 'in_out.asm'

section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '8'
b dd '88'
c dd '68'

section .bss
max resb 10

section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF
```

```

mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintLF

;-----сравнивание чисел
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov ecx,[a] ;
mov [max],ecx ;
;-----сравнивание чисел a c
cmp ecx,[c]; if a>c
jl check_b ; то перход на метку
mov ecx,[c] ;
mov [max],ecx ;
;-----метка check_b
check_b:
mov eax,max ;
call atoi
mov [max],eax ;
;-----
mov ecx,[max] ;
cmp ecx,[b] ;
jl check_c ;
mov ecx,[b] ;

```

```
mov [max],ecx ;
;-----
check_c:
mov eax,msg4 ;
call sprint ;
mov eax,[max];
call iprintLF ;
call quit
```

4

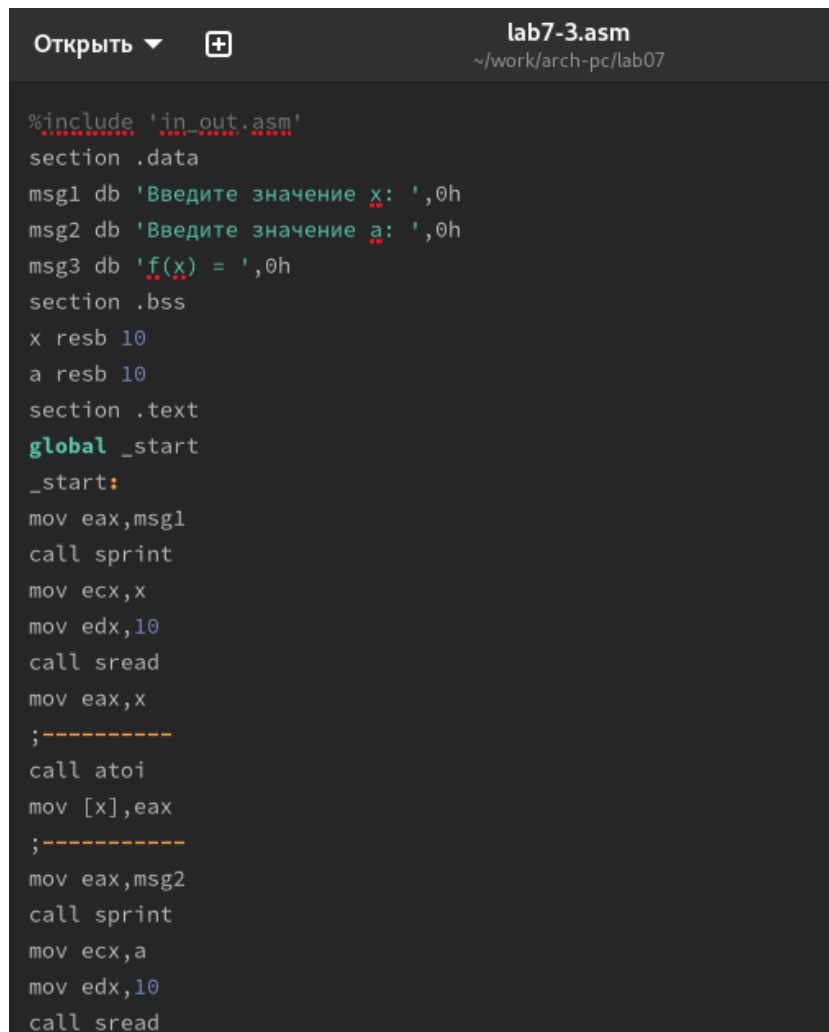
Создаю новый файл lab7-3 для написания программы второго задания. (рис. [5.4]).



Рис. 5.4: создание файла

5

Ввожу в него программу, в которую ввожу значения 4 х и а, и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (рис. [5.5]).



```
Открыть ▾ + lab7-3.asm
~/work/arch-pc/lab07

%include 'in_out.asm'
section .data
msg1 db 'Введите значение x: ',0h
msg2 db 'Введите значение a: ',0h
msg3 db 'f(x) = ',0h
section .bss
x resb 10
a resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
;-----
call atoi
mov [x],eax
;-----
mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
```

Рис. 5.5: ввод программы в файл

6

Создаю исполняемый файл и проверяю её выполнение при $x=3$, $a=0$

```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ nasm -f elf lab7-3.asm
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[ivan@fedora lab07]$ ./lab7-3
Введите значение x: 3
Введите значение a: 0
f(x) = 3
[ivan@fedora lab07]$
```

Рис. 5.6: Создание исполняемого файла

7

Повторный раз запускаю программу и проверяю ее выполнение при $x=3$ и $a=2$
Программа отработала верно!

```
ivan@fedora:~/work/arch-pc/lab07
[ivan@fedora lab07]$ nasm -f elf lab7-3.asm
[ivan@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[ivan@fedora lab07]$ ./lab7-3
Введите значение x: 3
Введите значение a: 2
f(x) = 1
[ivan@fedora lab07]$
```

Рис. 5.7: запуск исполняемого файла

Текст программы

```
%include 'in_out.asm'

section .data
msg1 db 'Введите значение x: ',0h
msg2 db 'Введите значение a: ',0h
msg3 db 'f(x) = ',0h

section .bss
x resb 10
a resb 10
```

```

section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
;-----
call atoi
mov [x],eax
;-----

mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a ;
call atoi
mov [a],eax ;
;-----
mov ecx,[a]
cmp ecx,[x] ;x<a
jg check_a ;
mov eax,[a]
mov ebx,-1

```

```
mul ebx
mov ecx,[x]
add ecx,eax
jmp _end
check_a:
mov ecx,5;
_end:
mov eax,msg3 ;
call sprintf ;
mov eax,ecx ;
call iprintLF;
call quit ;
```

6 Выводы

При выполнении данной лабораторной работы я освоил инструкции условного и безусловного вывода и ознакомился с структурой файла листинга.

:: {#refs} :::