

Шаблон отчёта по лабораторной работе

Простейший вариант

Лупупа Чилеше

Содержание

1	Цель работы	5
2	Задание	6
3	Задание для самостоятельной работы	15
4	Выводы	17

Список иллюстраций

2.1	lab7-1.asm	6
2.2	листинг 7.1	7
2.3	./lab7-1	8
2.4	листинг 7.2	9
2.5	./lab7-1	9
2.6	листинг 7.3	10
2.7	исполняемый файл	11
2.8	mcedit lab7-2.lst	12
2.9	lab7-2.asm	13
2.10	nasm -f elf -l lab7-2.lst lab7-2.asm	14
3.1	sr7	15
3.2	sr7	16

Список таблиц

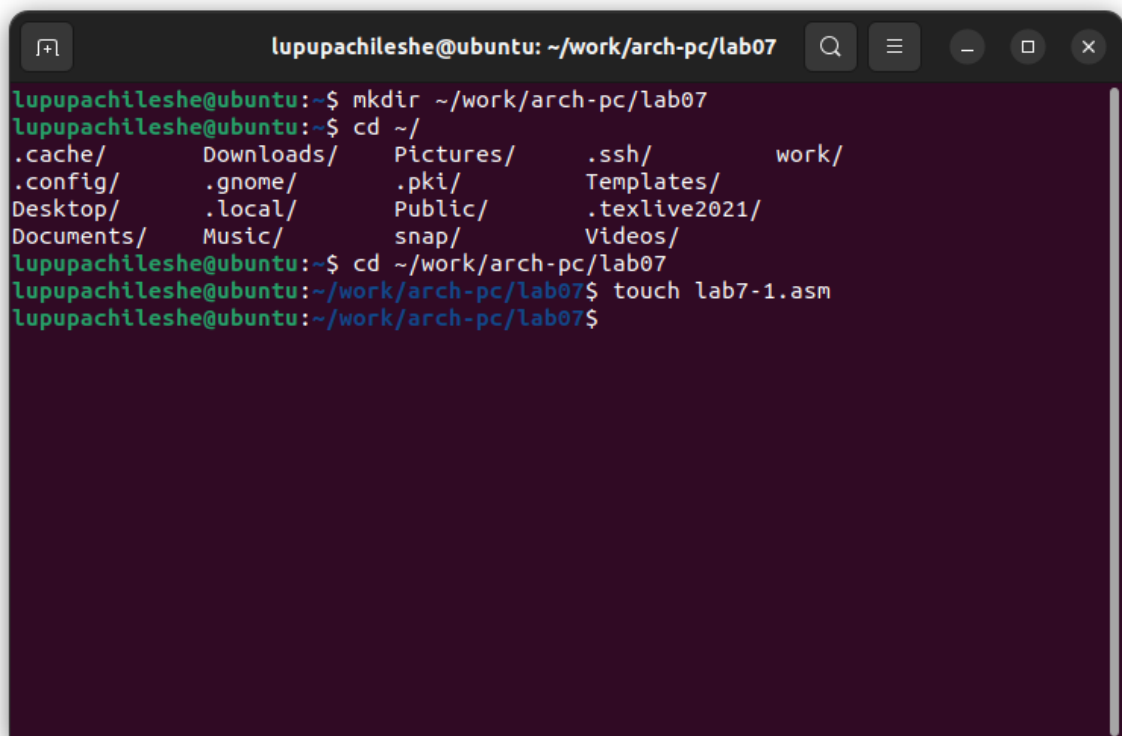
1 Цель работы

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

2 Задание

1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm:

```
mkdir ~/work/arch-pc/lab07 cd ~/work/arch-pc/lab07 touch lab7-1.asm
```

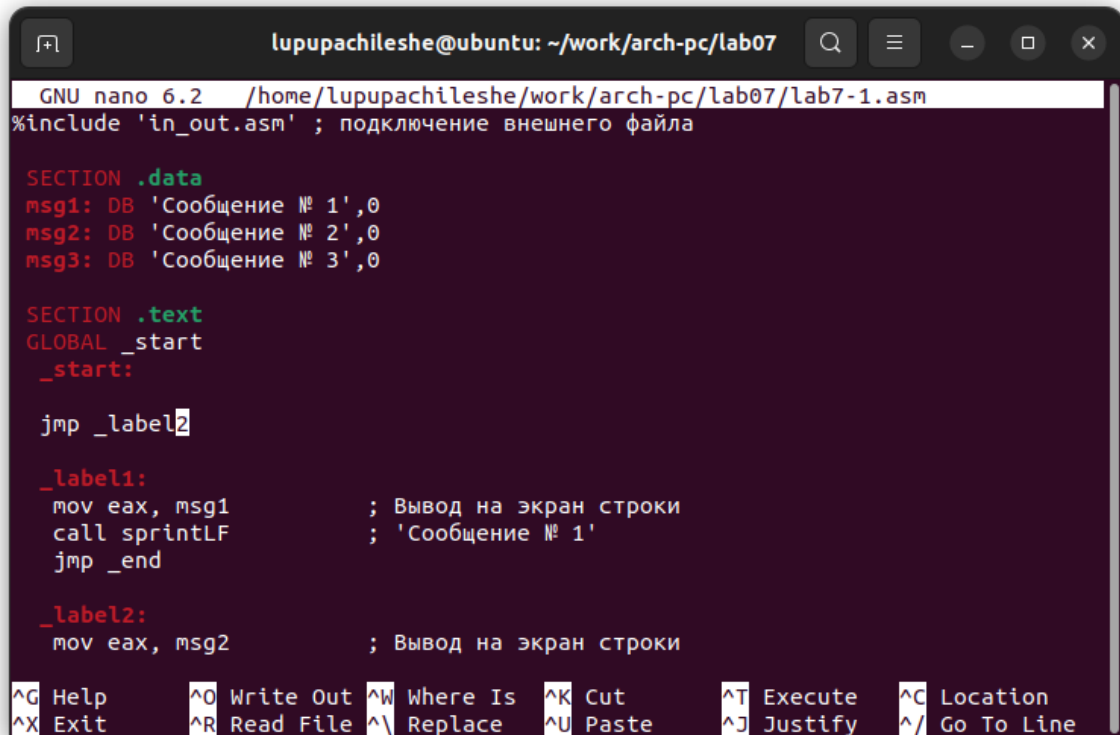


```
lupupachileshe@ubuntu: ~/work/arch-pc/lab07
lupupachileshe@ubuntu:~$ mkdir ~/work/arch-pc/lab07
lupupachileshe@ubuntu:~$ cd ~/work/arch-pc/lab07
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ touch lab7-1.asm
lupupachileshe@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.1: lab7-1.asm

Комментарий: Создал каталог под названием lab07. В нем создал файл с названием lab7-1.asm.

2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab7-1.asm` текст программы из листинга 7.1.



```
GNU nano 6.2 /home/lupupachileshe/work/arch-pc/lab07/lab7-1.asm
%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 sprintLF      ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2      ; Вывод на экран строки
```

Рис. 2.2: листинг 7.1

Комментарий: Текст из листинга 7.1 я поместил в файл `lab7-1.asm`.

Создайте исполняемый файл и запустите его. Результат работы данной программы будет следующим:

```
user@dk4n31:~$ ./lab7-1 Сообщение № 2 Сообщение № 3 user@dk4n31:~$
```

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

Рис. 2.3: ./lab7-1

Комментарий: Создал исполняемый файл и запустил его.

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. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 7.2.


```
mc [lupupachileshe@ubuntu]:~/work/arch-pc/lab07
GNU nano 6.2 /home/lupupachileshe/work/arch-pc/lab07/lab7-1.asm

jmp _label2

_label1:
mov eax, msg1      ; Вывод на экран строки
call printf        ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2      ; Вывод на экран строки
call printf        ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3      ; Вывод на экран строки
call printf        ; 'Сообщение № 3'

_end:
call _quit         ; вызов подпрограммы завершения

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 2.4: листинг 7.2

Комментарий: Я изменил текст в lab7-1.asm в соответствии с листингом 7.2.

Создайте исполняемый файл и проверьте его работу. Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим:

```
user@dk4n31:~$ ./lab7-1 Сообщение № 3 Сообщение № 2 Сообщение № 1
user@dk4n31:~$
```

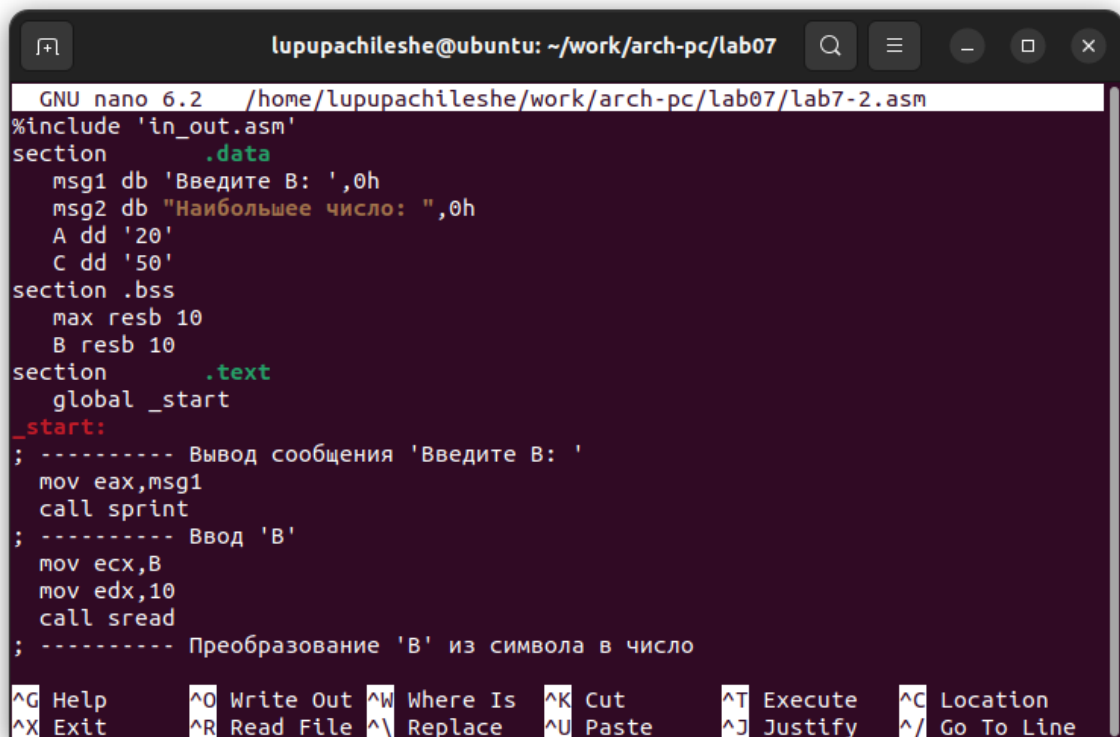
```
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
lupupachileshe@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.5: ./lab7-1

Комментарий: Создал исполняемый файл для lab7-1.asm и запустил его.

3. 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 рассмотрим 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.

Создайте файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Внимательно изучите текст программы из листинга 7.3 и введите в `lab7-2.asm`.



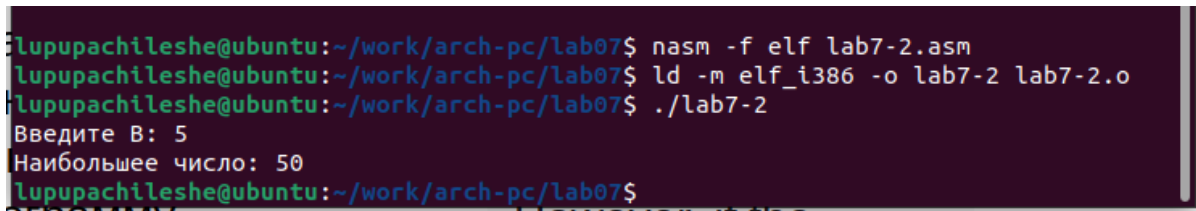
```
GNU nano 6.2 /home/lupupachileshe/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наибольшее число: ",0h
    A dd '20'
    C dd '50'
section .bss
    max resb 10
    B resb 10
section .text
    global _start
_start:
; ----- Вывод сообщения 'Введите B: '
    mov eax,msg1
    call sprint
; ----- Ввод 'B'
    mov ecx,B
    mov edx,10
    call sread
; ----- Преобразование 'B' из символа в число

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 2.6: листинг 7.3

Комментарий: Я создал файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Далее в созданный файл я ввел текст из листинга 7.3.

Создайте исполняемый файл и проверьте его работу для разных значений В.

A terminal window with a dark background and light green text. The prompt is 'lupupachileshe@ubuntu:~/work/arch-pc/lab07\$'. The commands entered are: 'nasm -f elf lab7-2.asm', 'ld -m elf_i386 -o lab7-2 lab7-2.o', and './lab7-2'. The output shows 'Введите В: 5' and 'Наибольшее число: 50'.

```
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
lupupachileshe@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.7: исполняемый файл

Комментарий: Я создал исполняемый файл и убедился, что он работает.

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. Создайте файл листинга для программы из файла `lab7-2.asm`

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

Откройте файл листинга `lab7-2.lst` с помощью любого текстового редактора, например `mcedit`:

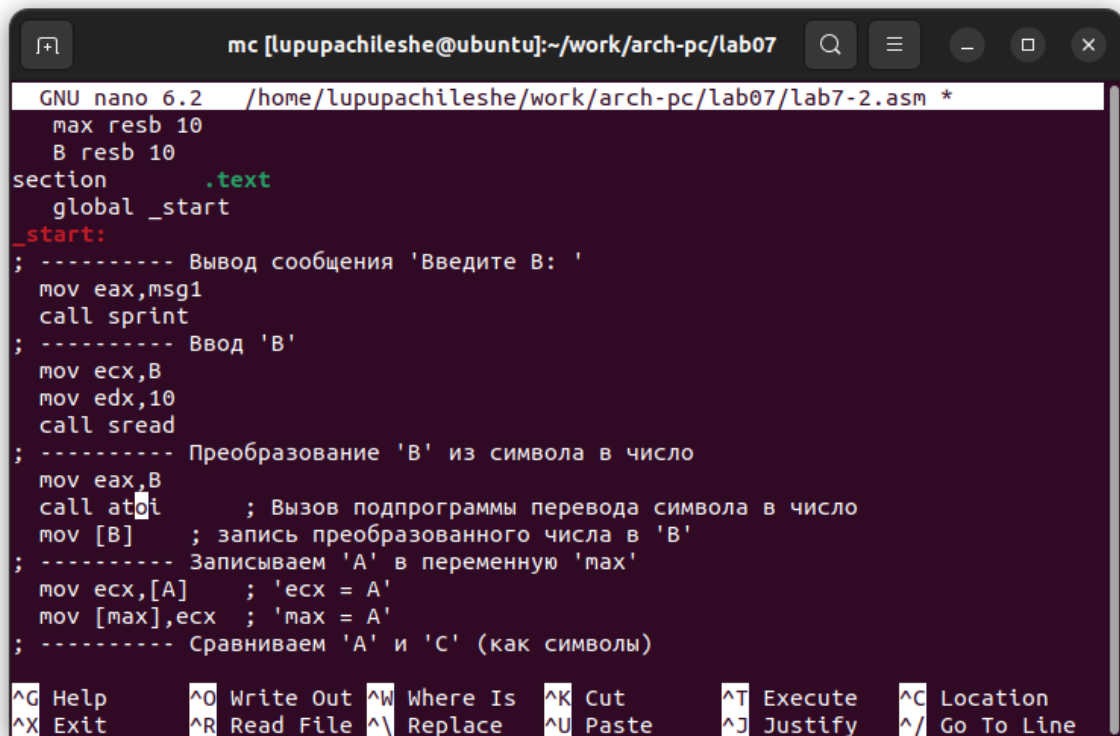
```
mcedit lab7-2.lst
```

```
lupupachileshe@ubuntu: ~/work/arch-pc/lab07
/home/lup~ab7-2.lst [----] 0 L:[ 1+ 0 1/225] *(0 /14594b) 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      00000003 803800  <1> nextchar:.....
10     00000006 7403    <1> cmp     byte [eax], 0...
11     00000008 40      <1> jz      finished.....
12     00000009 EBF8    <1> inc     eax.....
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>
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 2.8: mcedit lab7-2.lst

Комментарий: Я создал файл листинга программы в lab7-2.asm и открыл его с помощью текстового редактора mcedit.

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору. Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд.

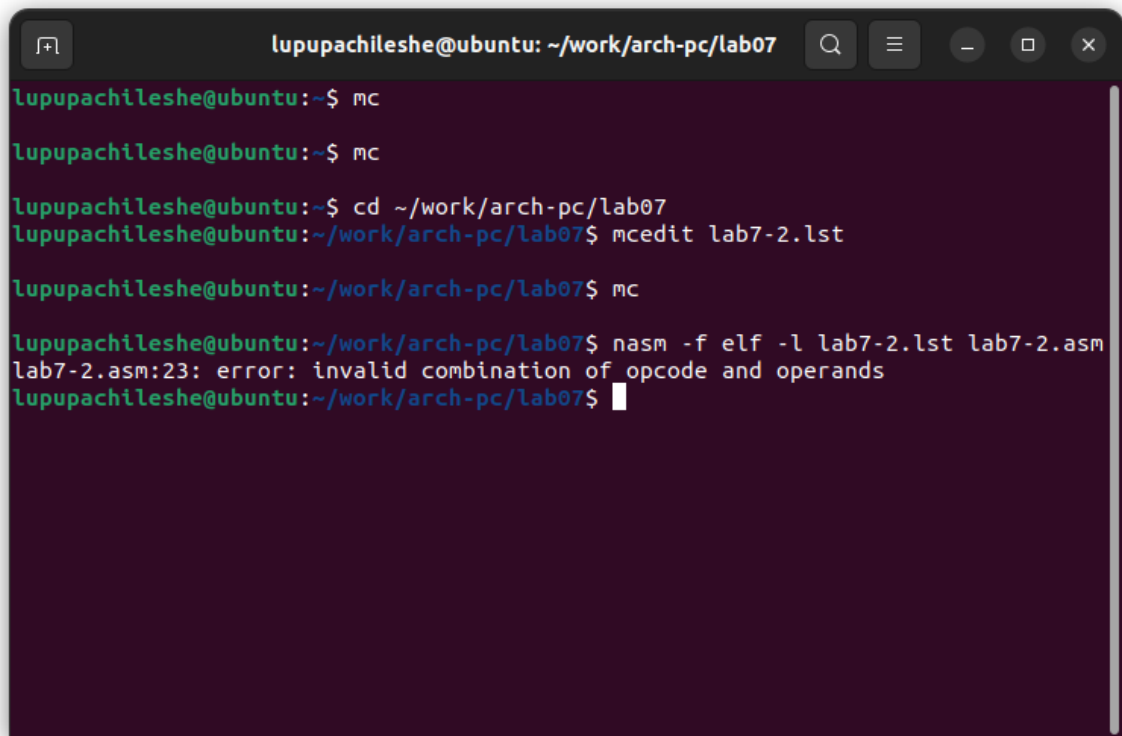


```
GNU nano 6.2 /home/lupupachileshe/work/arch-pc/lab07/lab7-2.asm *
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B] ; 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)
```

Рис. 2.9: lab7-2.asm

Выполните трансляцию с получением файла листинга:

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

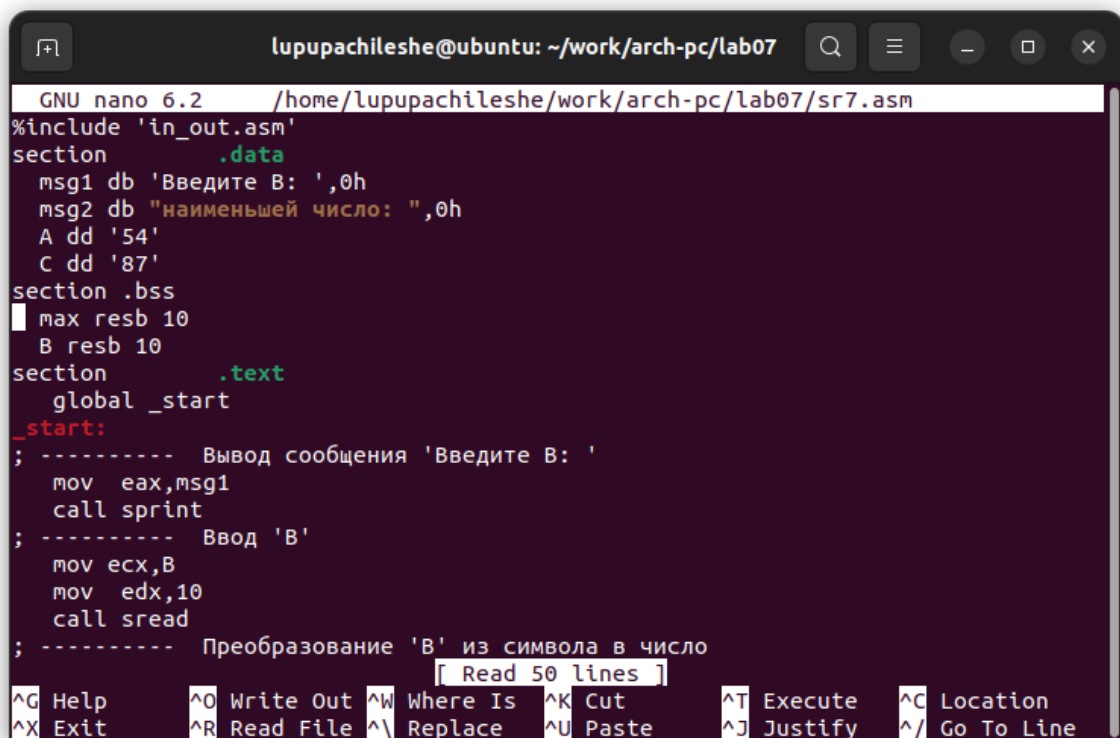


```
lupupachileshe@ubuntu: ~/work/arch-pc/lab07
lupupachileshe@ubuntu:~$ mc
lupupachileshe@ubuntu:~$ mc
lupupachileshe@ubuntu:~$ cd ~/work/arch-pc/lab07
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ mcedit lab7-2.lst
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ mc
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:23: error: invalid combination of opcode and operands
lupupachileshe@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.10: `nasm -f elf -l lab7-2.lst lab7-2.asm`

3 Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных \boxed{A} , \boxed{B} и \boxed{C} . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.



```
GNU nano 6.2 /home/lupupachileshe/work/arch-pc/lab07/sr7.asm
#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "наименьшей число: ",0h
    A dd '54'
    C dd '87'
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
    Read 50 lines
^G Help      ^O Write Out ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace  ^U Paste    ^J Justify  ^_ Go To Line
```

Рис. 3.1: sr7

```
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ nasm -f elf sr7.asm
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o sr7 sr7.o
lupupachileshe@ubuntu:~/work/arch-pc/lab07$ ./sr7
Введите В: 62
наименьшей число: 54
lupupachileshe@ubuntu:~/work/arch-pc/lab07$
```

Рис. 3.2: sr7

2. Напишите программу, которая для введенных с клавиатуры значений x и y вычисляет значение заданной функции $f(x, y)$ и выводит результат вычислений. Вид функции $f(x, y)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и y из 7.6.

4 Выводы

Я научился использовать команду безусловного и условного перехода. Далее я научился писать программы с использованием переходов. Я также хорошо понял назначение и структуру файла листинга.