

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Лопатин Павел Юрьевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Самостоятельная работа	14
4	Вывод	22

Список иллюстраций

2.1	Создание директории	5
2.2	Редактирование файла	6
2.3	Запуск исполняемого файла	6
2.4	Редактирование файла	7
2.5	Запуск исполняемого файла	7
2.6	Редактирование программы	8
2.7	Создание исполняемого файла	8
2.8	Создание файла	9
2.9	Вставляю текст в файл	9
2.10	Запуск исполняемого файла	10
2.11	Редактирование файла	10
2.12	Файл листинга	11
2.13	Файл листинга	12
2.14	Файл листинга	13
3.1	Создание файла	14
3.2	Редактирование файла	15
3.3	Запуск исполняемого файла	15
3.4	создание файла	18
3.5	ввод программы в файл	19
3.6	запуск исполняемого файла	19

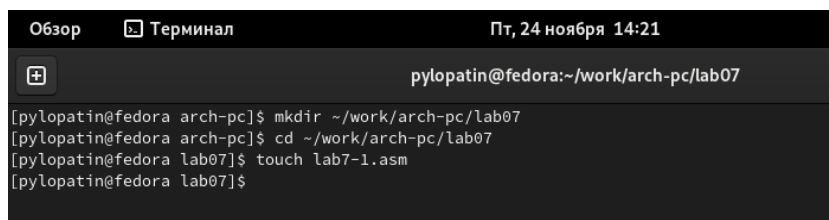
1 Цель работы

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

2 Выполнение лабораторной работы

1

С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [2.1])

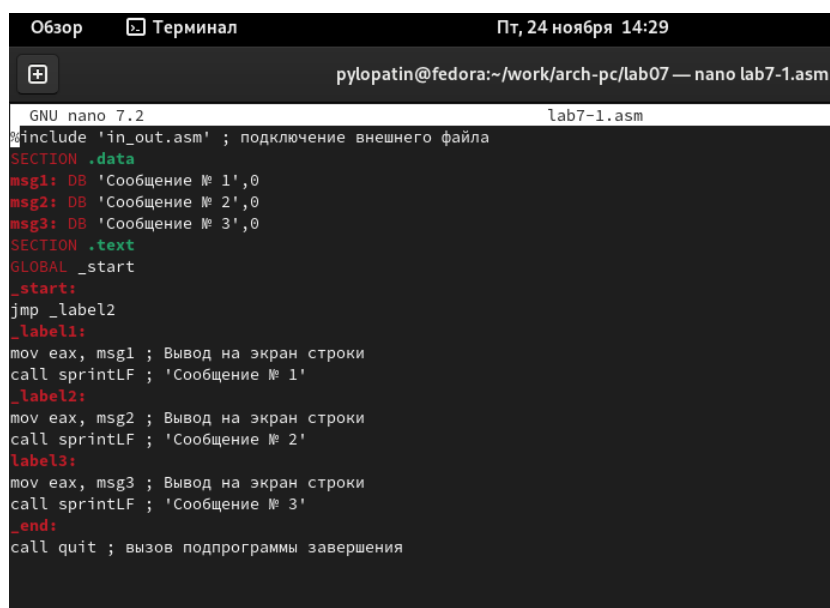


```
Обзор Терминал Пт, 24 ноября 14:21
pylopatin@fedora:~/work/arch-pc/lab07
[pylopatin@fedora arch-pc]$ mkdir ~/work/arch-pc/lab07
[pylopatin@fedora arch-pc]$ cd ~/work/arch-pc/lab07
[pylopatin@fedora lab07]$ touch lab7-1.asm
[pylopatin@fedora lab07]$
```

Рис. 2.1: Создание директории

2

Открываю созданный файл `lab7-1.asm`, вставляю в него программу реализации безусловных переходов(рис. [2.2]).



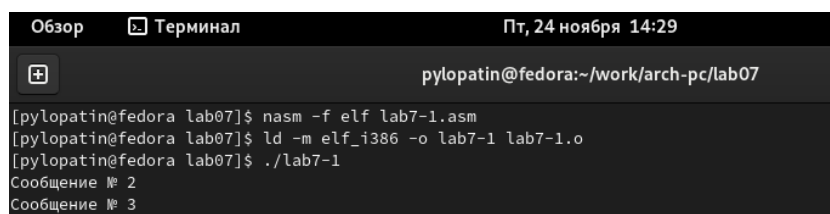
```
Обзор Терминал Пт, 24 ноября 14:29
pylopatin@fedora:~/work/arch-pc/lab07 — nano lab7-1.asm

GNU nano 7.2 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 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. 2.2: Редактирование файла

3

Создаю исполняемый файл программы и запускаю его (рис. [2.3]). Инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`.



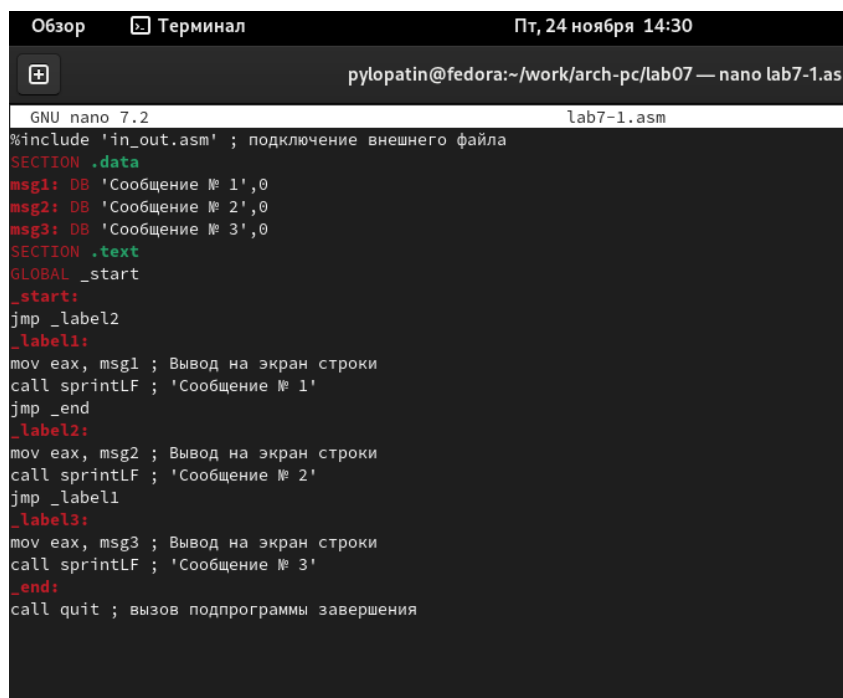
```
Обзор Терминал Пт, 24 ноября 14:29
pylopatin@fedora:~/work/arch-pc/lab07

[pylopatin@fedora lab07]$ nasm -f elf lab7-1.asm
[pylopatin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[pylopatin@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 2.3: Запуск исполняемого файла

4

Изменяю текст программы так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу (рис. [2.4]).

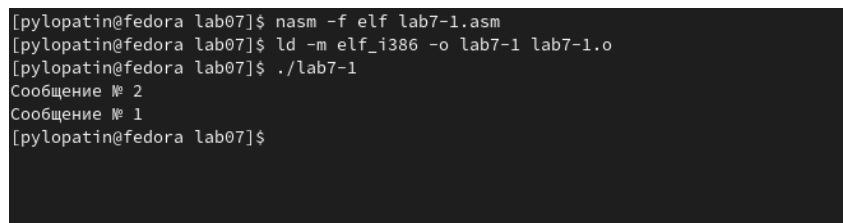


```
Обзор Терминал Пт, 24 ноября 14:30
pylopatin@fedora:~/work/arch-pc/lab07 — nano lab7-1.as
GNU nano 7.2 lab7-1.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call 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'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.4: Редактирование файла

5

Создаю новый исполняемый файл программы и запускаю его (рис. [2.5]). Убеждаюсь в том, программа работает верно.

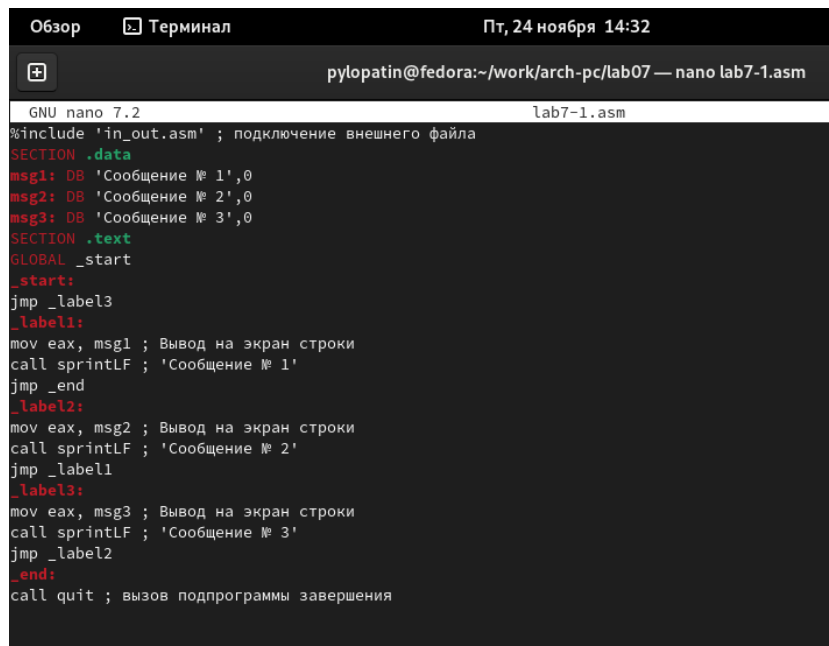


```
[pylopatin@fedora lab07]$ nasm -f elf lab7-1.asm
[pylopatin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[pylopatin@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[pylopatin@fedora lab07]$
```

Рис. 2.5: Запуск исполняемого файла

6

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [2.6]).

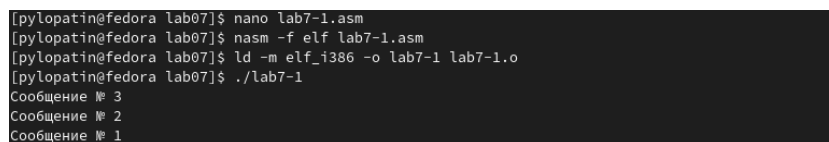


```
Обзор Терминал Пт, 24 ноября 14:32
pylopatin@fedora:~/work/arch-pc/lab07 — nano lab7-1.asm
GNU nano 7.2 lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _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. 2.6: Редактирование программы

7

Создаю исполняемый файл и проверяю работу программы (рис. [2.7]). Программа отработало верно.



```
[pylopatin@fedora lab07]$ nano lab7-1.asm
[pylopatin@fedora lab07]$ nasm -f elf lab7-1.asm
[pylopatin@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[pylopatin@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 2.7: Создание исполняемого файла

8

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис. [2.8]).


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

Рис. 2.8: Создание файла

9

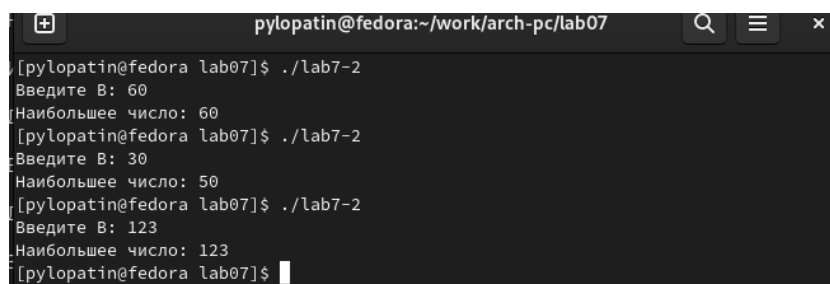
Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[2.9]).

```
GNU nano 7.2 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'
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
; Справка      ; Записать   ; Поиск      ; Вырезать   ; Выполнить  ; Позиция    ; Отмена
; Выход       ; ЧитФайл   ; Замена     ; Вставить   ; Выровнять  ; К строке   ; Повтор
```

Рис. 2.9: Вставляю текст в файл

10

Создаю и запускаю новый исполняемый файл, проверяю работу программы для разных B, при A=20 и C=50 (рис. [2.10]).

A terminal window titled 'pylopatin@fedora:~/work/arch-pc/lab07'. The user runs './lab7-2'. The program prompts 'Введите B: 60', then 'Наибольшее число: 60'. The user runs './lab7-2' again. The program prompts 'Введите B: 30', then 'Наибольшее число: 50'. The user runs './lab7-2' a third time. The program prompts 'Введите B: 123', then 'Наибольшее число: 123'. The terminal ends with the prompt '[pylopatin@fedora lab07]\$' and a cursor.

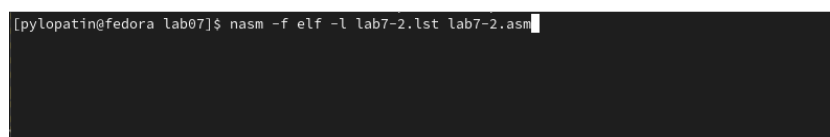
```
[pylopatin@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[pylopatin@fedora lab07]$ ./lab7-2
Введите B: 30
Наибольшее число: 50
[pylopatin@fedora lab07]$ ./lab7-2
Введите B: 123
Наибольшее число: 123
[pylopatin@fedora lab07]$
```

Рис. 2.10: Запуск исполняемого файла

11

Создаю файл листинга для программы в файле lab7-2.asm и открываю его в редакторе mcedit (рис. [2.11]).

.

A terminal window showing the command 'nasm -f elf -l lab7-2.lst lab7-2.asm' being entered at the prompt '[pylopatin@fedora lab07]\$'.

```
[pylopatin@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 2.11: Редактирование файла

12

Открываю файл листинга с помощью редактора mcedit. Рассмотрим 9-11 строки: (рис. [2.12]).

```

lab7-2.lst  [---]  0 L:[ 1+ 0  1/225] + (0  /14458b) 0032 0x020  [+][X]
1
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      00000003 803800  <1> nextchar: -----
9      00000006 7403    <1> cmp   byte [eax], 0
10     00000008 40      <1> jz     finished
11     00000009 EBF8    <1> inc   eax
12     <1> -----
13     <1> finished: -----
14     0000000B 29D8    <1> jmp   nextchar
15     0000000D 5B      <1> sub   eax, ebx
16     0000000E C3      <1> pop   ebx
17     <1> -----
18     <1> -----
19     <1> ;----- sprint -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov eax,<message>
22     <1> sprint: -----
23     0000000F 52      <1> push  edx
24     00000010 51      <1> push  ecx
25     00000011 53      <1> push  ebx
26     00000012 50      <1> push  eax
27     00000013 E8E8FFFF <1> call  slen
28     <1> -----
29     00000018 89C2    <1> mov   edx, eax
30     0000001A 58      <1> pop   eax
31     <1> -----
32     0000001B 89C1    <1> mov   ecx, eax
33     0000001D B801000000 <1> mov   ebx, 1
34     00000022 B804000000 <1> mov   eax, 4
35     00000027 CD80    <1> int   80h
36     <1> -----
37     00000029 5B      <1> pop   ebx
38     0000002A 59      <1> pop   ecx
39     0000002B 5A      <1> pop   edx
40     0000002C C3      <1> ret
41     <1> -----
42     <1> -----
43     <1> ;----- sprintf -----
44     <1> ; Функция печати сообщения с переводом строки

```

Рис. 2.12: Файл листинга

9 строка:

- Первая цифра [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 строкк исходной программы вместе с комментариями.

10 строка:

- Первое число [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:

- Первое число [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

13

Открываю файл lab7-2.asm с помощью редактора и Удаляю один операнд в инструкции cmp. (рис. [2.13]).

```

; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx, ; 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. 2.13: Файл листинга

14

Открываю файл листинга с помощью редактора mscedit и замечаю, что в файле листинга появляется ошибка. (рис. [2.14]).

```

28      cmp ecx, ; Сравниваем 'A' и 'C'
28      *****
29 0000011C 7F0C      jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011E 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 00000124 890D[00000000] mov [max],ecx ; 'max = C'
32      ; ----- Преобразование 'max(A,C)' из символа в число
33      check_B:

```

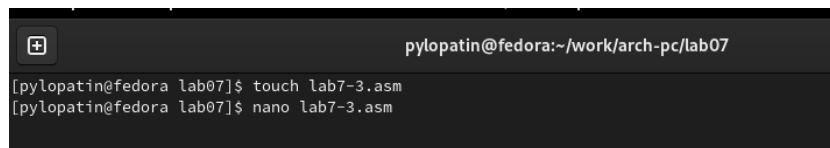
Рис. 2.14: Файл листинга

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

3 Самостоятельная работа

1

Создаю файл lab7-3.asm с помощью утилиты touch (рис. [3.1]).

A screenshot of a terminal window with a dark background. The window title is 'pylopatin@fedora:~/work/arch-pc/lab07'. The terminal shows two commands being executed: '[pylopatin@fedora lab07]\$ touch lab7-3.asm' and '[pylopatin@fedora lab07]\$ nano lab7-3.asm'.

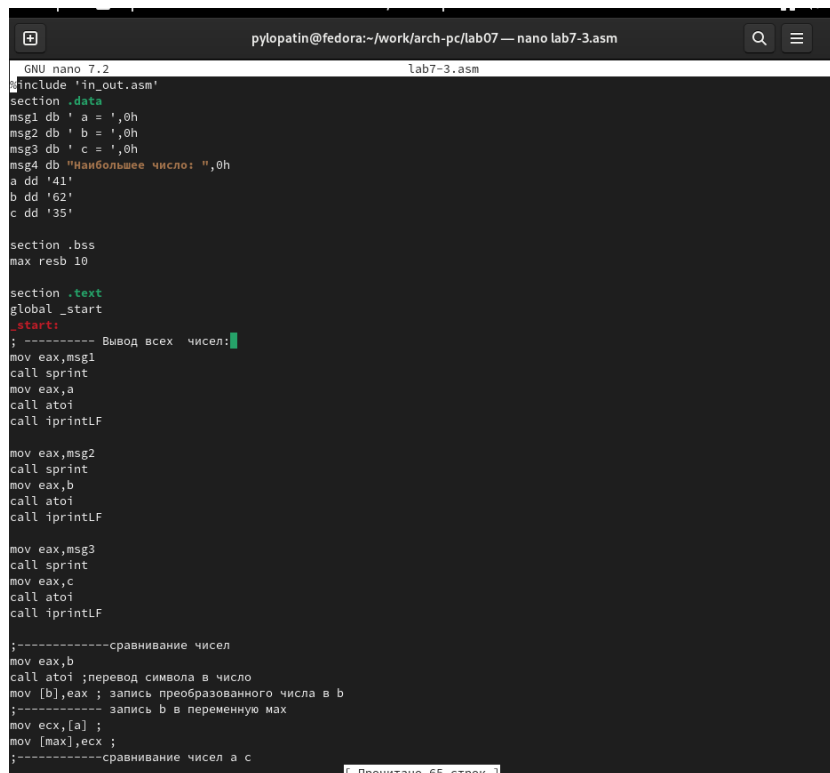
```
pylopatin@fedora:~/work/arch-pc/lab07
[pylopatin@fedora lab07]$ touch lab7-3.asm
[pylopatin@fedora lab07]$ nano lab7-3.asm
```

Рис. 3.1: Создание файла

2

Ввожу в созданный файл текст программы для вычисления наибольшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы.

10 вариант (рис. [3.2]).



```
GNU nano 7.2 lab7-3.asm
#include 'in_out.asm'
section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наибольшее число: ",0h
a dd '41'
b dd '62'
c dd '35'

section .bss
max resb 10

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

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintf

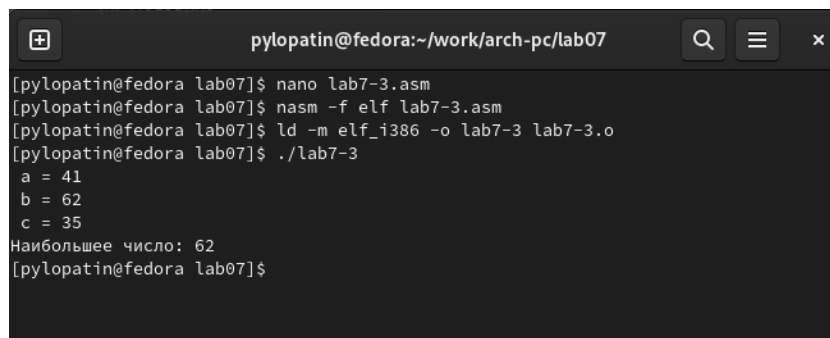
mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintf

;-----сравнение 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 ;
;-----сравнение чисел a с
```

Рис. 3.2: Редактирование файла

3

Создаю исполняемый файл и запускаю его (рис. [3.3]).



```
[pylopatin@fedora lab07]$ nano lab7-3.asm
[pylopatin@fedora lab07]$ nasm -f elf lab7-3.asm
[pylopatin@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[pylopatin@fedora lab07]$ ./lab7-3
a = 41
b = 62
c = 35
Наибольшее число: 62
[pylopatin@fedora lab07]$
```

Рис. 3.3: 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 "Наибольшее число: ",0h
a dd '41'
b dd '62'
c dd '35'

```

```

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
jg 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] ;
jg 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 lab7-4 для написания программы второго задания. (рис. [3.4]).



```
[pylopatin@fedora lab07]$ touch lab7-4.asm
[pylopatin@fedora lab07]$ nano lab7-4.asm
```

Рис. 3.4: создание файла

5

Ввожу в него программу, (рис. [3.5]). в которую ввожу 2 значения x и a, и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (Вариант №10)

```

Обзор Терминал Пт, 24 ноября 14:57 en
pylopatin@fedora:~/work/arch-pc/lab07 — nano lab7-4.asm
GNU nano 7.2 lab7-4.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите 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,2
cmp [x],ecx ;x>2
jg check_a ;
mov eax,[a]
mov ebx,3 ;
mul ebx ;
mov ecx,eax
jmp _end ;
check_a:
mov ecx,[x];

```

Рис. 3.5: ввод программы в файл

6

Создаю исполняемый файл и проверяю её выполнение при $x=3$, $a=0$, $x=1$ и $a=2$ (рис. [3.6]). Программа отработала верно!

```

[pylopatin@fedora lab07]$ nasm -f elf lab7-4.asm
[pylopatin@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[pylopatin@fedora lab07]$ ./lab7-4
Введите x: 3
Введите a: 0
f(x) = 1
[pylopatin@fedora lab07]$ ./lab7-4
Введите x: 1
Введите a: 2
f(x) = 6
[pylopatin@fedora lab07]$

```

Рис. 3.6: запуск исполняемого файла

Текст программы

```
%include 'in_out.asm'
```

```

section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите 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,2
cmp [x],ecx ;x>2
jg check_a ;
mov eax,[a]
mov ebx,3 ;
mul ebx ;
mov ecx,eax
jmp _end ;
check_a:
mov ecx,[x];
add ecx,-2 ; x-2
_end:
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintLF;
call quit ;

```

4 Вывод

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