

Отчёт по лабораторной работе №7

Архитектура компьютера НММбд-03-24

Туева Анастасия Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	14
5	Выводы	19

Список иллюстраций

3.1	Создание файла	7
3.2	Редактирование файла	8
3.3	Запуск исполняемого файла	8
3.4	Редактирование файла	9
3.5	Запуск исполняемого файла	9
3.6	Редактирование файла	10
3.7	Запуск исполняемого файла	10
3.8	Редактирование файла	11
3.9	Запуск исполняемого файла	12
3.10	Файл листинга “lab7-2.asm”	12
3.11	Строка 1	12
3.12	Строка 2	13
3.13	Строка 3	13
3.14	Создание файла без одного операнда	13
3.15	Файл листинга безодного операнда	13
4.1	Редактирование файла	15
4.2	Запуск исполняемого файла	16
4.3	Редактирование файла	17
4.4	Запуск исполняемого файла	18

Список таблиц

1 Цель работы

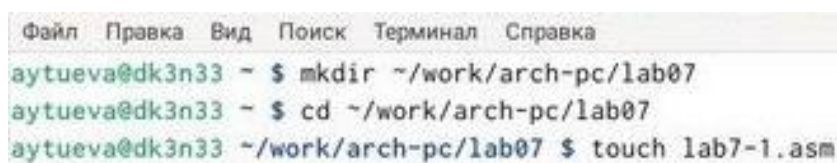
Изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

2 Задание

1. Изучение команд условного и безусловного переходов
2. Выполнение лабораторной работы
3. Выполнение самостоятельной работы

3 Выполнение лабораторной работы

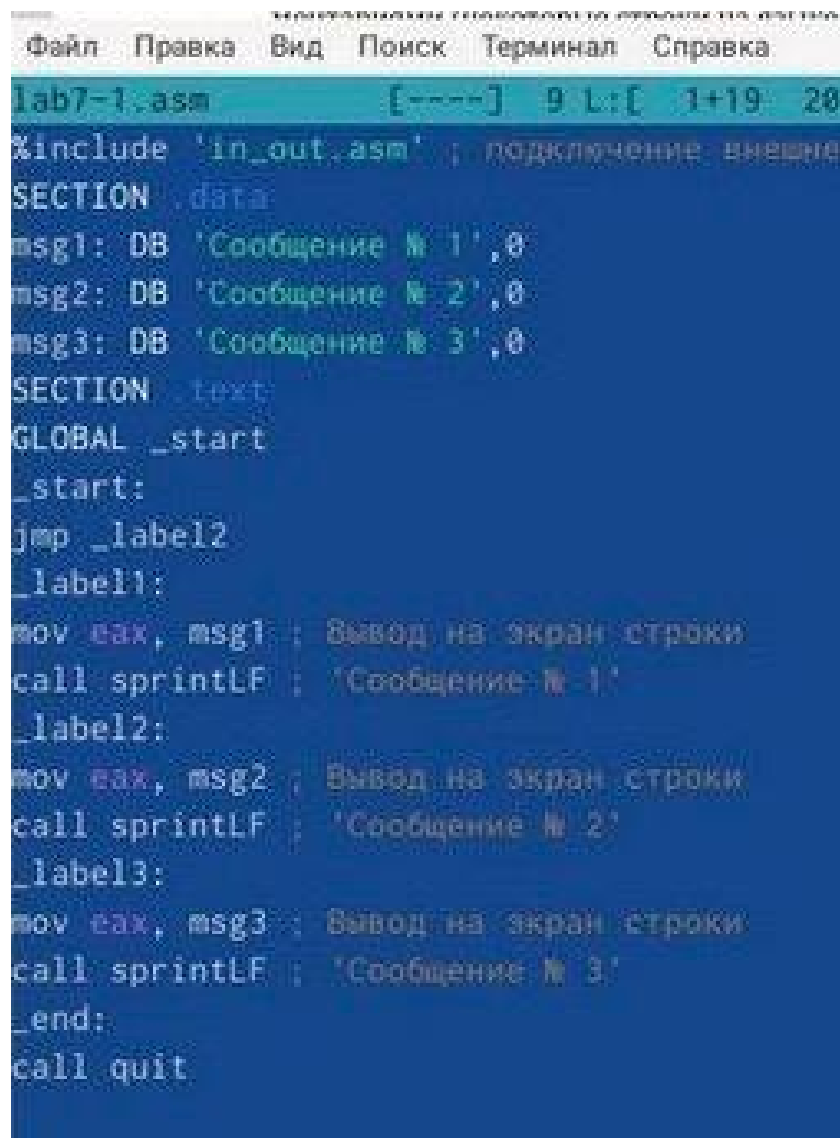
Создаём каталог для программ лабораторной работы № 7, переходим в него и создаём файл “lab7-1.asm” (рис. 3.1).



```
Файл  Правка  Вид  Поиск  Терминал  Справка
aytueva@dk3n33 ~ $ mkdir ~/work/arch-pc/lab07
aytueva@dk3n33 ~ $ cd ~/work/arch-pc/lab07
aytueva@dk3n33 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.1: Создание файла

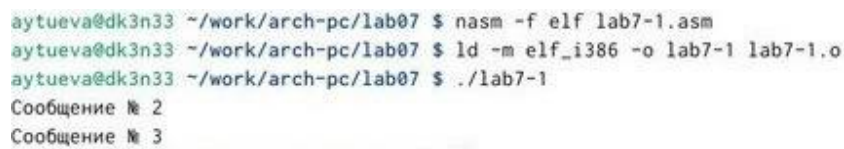
Открываем созданный файл “lab7-1.asm”, вставляем в него данный текст в соответствии с листингом 7.1. (рис. 3.2).



```
Файл  Правка  Вид  Поиск  Терминал  Справка
lab7-1.asm  [----]  9  L: [ 1+19  28
#include 'in_out.asm' ; подключение внешне
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 sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit
```

Рис. 3.2: Редактирование файла

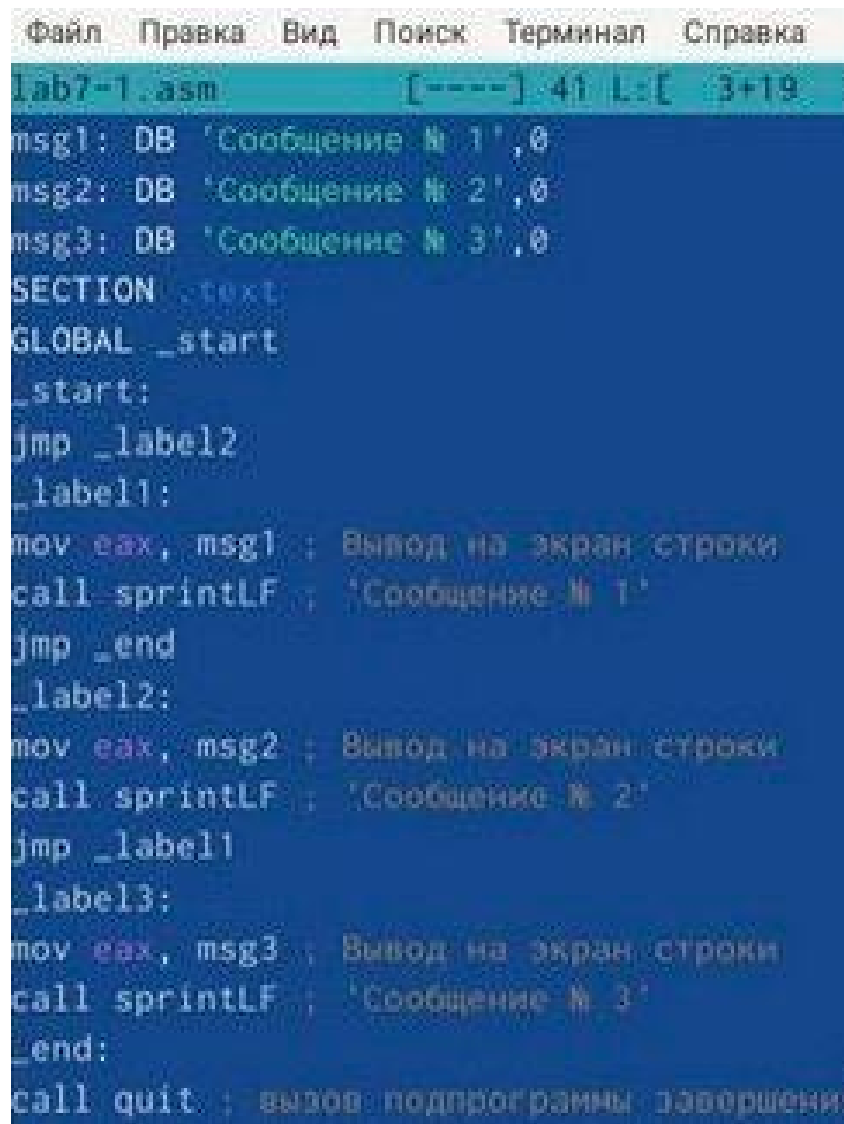
Создаем исполняемый файл программы и запускаем его (рис. 3.3).



```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Запуск исполняемого файла

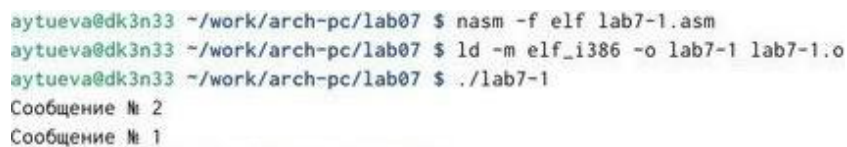
Изменяем в текст программы в соответствии с листингом 7.2. (рис. 3.4).



```
Файл  Правка  Вид  Поиск  Терминал  Справка
lab7-1.asm  [----] 41 L: [ 3+19
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 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 завершени
```

Рис. 3.4: Редактирование файла

Создаем новый исполняемый файл программы и запускаем его (рис. 3.5).



```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Запуск исполняемого файла

Вводим в файл текст программы так, чтобы вывод программы был как в задании (рис. 3.6).

```

lab7-1.asm          [----] 41 L:[ 1+22 23/ 2
#include 'in_out.asm' ; подключение внешнего
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. 3.6: Редактирование файла

Создаем новый исполняемый файл программы и запускаем его (рис. 3.7).

```

aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.7: Запуск исполняемого файла

Создаем новый файл lab7-2.asm с помощью команды “touch”.

Вводим в файл текст другой программы в соответствии с листингом 7.3. (рис. 3.8).

```
lab7-2.asm      [-M--] 47 L:[ 1+39 40/ 49] *(147)
#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'
; ----- Считаем max(A,C) + max(B,C) (преобразование)
```

Рис. 3.8: Редактирование файла

Создаем исполняемый файл программы и запускаем его. Для проверки работы программы я ввела 3 разных числа (рис. 3.9).

```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 40
Наибольшее число: 50
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 10
Наибольшее число: 50
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 80
Наибольшее число: 80
```

Рис. 3.9: Запуск исполняемого файла

Создадим файл листинга для программы из файла “lab7-2.asm” и откроем его с помощью любого текстового редактора, например “mcedit” (рис. 3.10).

```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst

aytueva@dk3n33 ~/work/arch-pc/lab07 $
```

Рис. 3.10: Файл листинга “lab7-2.asm”

Проанализировав файл, я поняла как он работает и какие значения выводит.

Эта строка находится на 21 месте, ее адрес “00000101”, Машинный код - “B8 [0A000000]”, а “mov eax,B” - исходный текст программы, означающий что в регистр “eax” мы вносим значения переменной В. (рис. 3.11).

```
20 00000000 00000000 mov eax,0
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
```

Рис. 3.11: Строка 1

Эта строка находится на 35 месте, ее адрес “00000135”, Машинный код - “E862FFFFFF”, а “call atoi” - исходный текст программы, означающий что символ лежащий в строке выше переводится в число. (рис. 3.12).

```

34 00000130 E867FFFFFF      call atoi ;
35 00000135 E862FFFFFF      call atoi ;
36 0000013A A3[00000000]      mov [max],e

```

Рис. 3.12: Строка 2

Эта строка находится на 47 месте, ее адрес “00000163”, Машинный код - “A1[00000000]”, а “mov eax,[max]” - исходный текст программы, означающий что число хранившееся в переменной “max” записывается в регистр “eax”. (рис. 3.13).

```

46 0000013E E8ACFFFFFF      call sprintf ;
47 00000163 A1[00000000]      mov eax,[max]
48 00000168 E819FFFFFF      call iprintlf

```

Рис. 3.13: Строка 3

В строке “mov eax,max” я убрала “max” и попробовала создать файл. Выдало ошибку, так как для программы нужно два операнда. (рис. 3.14).

```

iytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
iytueva@dk3n33 ~/work/arch-pc/lab07 $

```

Рис. 3.14: Создание файла без одного операнда

В файле листинга показывает, где именно ошибка и с чем она связана.(рис. 3.15).

```

34 ***** error: invalid combination of opcode and operands.
35 00000130 E867FFFFFF      call atoi ; Вызов подпрограммы перевода символа в
36 00000135 A3[00000000]      mov [max],eax ; запись преобразованного числа в 'ma
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числ

```

Рис. 3.15: Файл листинга безодного операнда

4 Выполнение самостоятельной работы

Создаем новый файл lab7-3.asm с помощью команды “touch”.

Напишем программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Мой вариант - 11. (рис. 4.1).

```

lab7-3.asm      [-M--] 21 L: [ 1+
%include 'in_out.asm'
SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
otv DB 'Наименьшее число: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20
SECTION .text
GLOBAL _start
_start:
mov eax,A1
call sprint
mov ecx,A
mov edx,20
call sread
mov eax, A
call atoi
mov [A],eax
xor eax,eax
mov eax,B1
call sprint
mov ecx,B
mov edx,20
call sread
mov eax,B
call atoi
mov [B],eax
xor eax,eax
mov ecx, [A]
mov [min],ecx
mov ecx,[min]
cmp ecx,[B]
jnl check_C
mov ecx, [B]
mov [min],ecx
check_C:
mov eax,C1
call sprint

```

Рис. 4.1: Редактирование файла

Создаем исполняемый файл программы и запускаем его. Ответ верный.(рис. 4.2).

```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-3
Введите число A: 21
Введите число B: 28
Введите число C: 34
Наименьшее число: 21
```

Рис. 4.2: Запуск исполняемого файла

Создаем новый файл lab7-4.asm с помощью команды “touch”.

Во втором номере необходимо написать программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. (рис. 4.3).


```

lab7-4.asm [-
call atoi
mov [A],eax

mov ecx,[X]
mov [F],ecx

cmp ecx,0
je check_or
mov eax,[A]
mov ebx,4
mul ebx
add eax,[X]
mov [F],eax
jmp fin

check_or:
mov eax,[A]
mov ebx,4
mul ebx
mov [F],eax

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

```

Рис. 4.3: Редактирование файла

Создаем и запускаем исполняемый файл. Ответы верны. (рис. 4.4).

```
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-4
4a ,x=0
4a+x, x!=0
Введите значение X:0
Введите значение a:3
Ответ: 12
aytueva@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
aytueva@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-4
4a ,x=0
4a+x, x!=0
Введите значение X:1
Введите значение a:2
Ответ: 9
```

Рис. 4.4: Запуск исполняемого файла

5 Выводы

Благодаря данной лабораторной работе я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файла листинга.