

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

Студент: Барханоева Раина Магометовна

Ст.билет: 1032252468

Группа: НКАбд-01-25

МОСКВА

2025 г

Содержание

1. Цель работы.....	2
2. Работа.....	3
2.1. Ответы на вопросы:.....	8
3. Самостоятельная работа.....	9
4. Вывод.....	11

1. Цель работы

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

2. Работа

Каталог (рисунок 1).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ mkdir lab07
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ cd lab07/
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ touch lab7-1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ls
lab7-1.asm
```

Рисунок 1.

Создала каталог и файл lab7-1.asm.

Программа (рисунок 2).

```
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
call sprintLF
; Вывод на экран строки
; 'Сообщение № 1'

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

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

_end:
call quit
; вызов подпрограммы завершения

^G Справка      ^O Записать      ^W Поиск      ^K Вырезать      ^T Выполнить
^X Выход      ^R ЧитФайл      ^\ Замена      ^U Вставить      ^J Выровнять
```

Рисунок 2.

Вставила программу с использованием инструкции jmp

Тест программы (рисунок 3).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рисунок 3.

Создали исполняемый файл и запустили программу. Использование инструкции jmp _label2 меняет порядок исполнения инструкций и позволяет

выполнить инструкции начиная с метки _label2, пропустив вывод первого сообщения.

Программа (рисунок 4).

```
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
call sprintLF
jmp _end
; Вывод на экран строки
; 'Сообщение № 1'

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

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

_end:
call quit
; вызов подпрограммы завершения
Имя файла для записи: lab7-1.asm
^G Справка      M-D Формат DOS      M-A Доп. в начало      M-B Резерв. копия
^C Отмена      M-M Формат Mac      M-P Доп. в конец      ^T Обзор
```

Рисунок 4.

Изменила немного программу.

Тест программы (рисунок 5.)

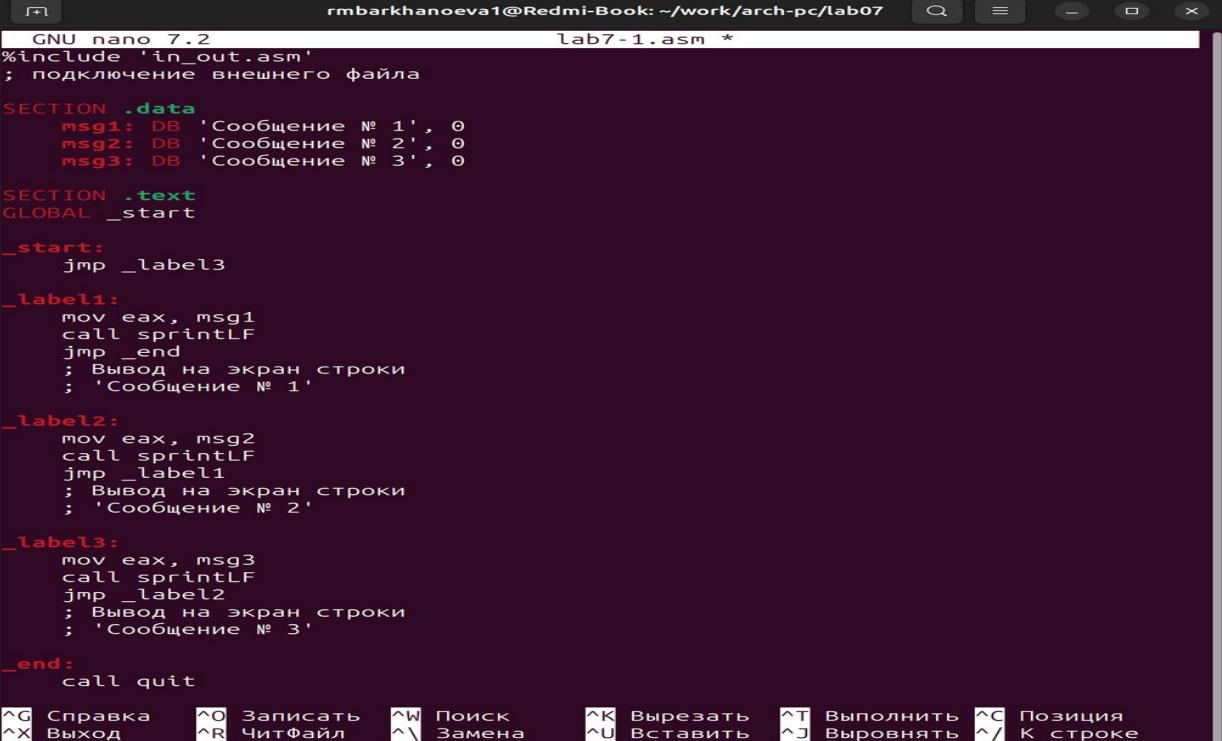
```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рисунок 5.

Изменив программу, добавив 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), мы сделали так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу.

Программа (рисунок).6



```
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
    call sprintLF
    jmp _end
; Вывод на экран строки
; 'Сообщение № 1'

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

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

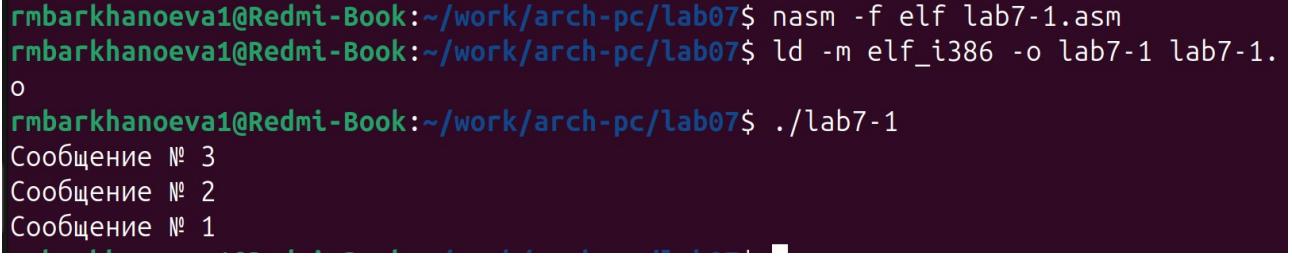
_end:
    call quit

^G Справка   ^O Записать   ^W Поиск
^X Выход   ^R ЧитФайл   ^\ Замена   ^K Вырезать
          ^U Вставить   ^T Выполнить   ^C Позиция
          ^J Выровнять   ^/ К строке
```

Рисунок 6.

Изменила программу, чтобы вывод был 3 2 1

Тест программы (рисунок 7).



```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рисунок 7.

Запустили и протестирували программу.

Программа (рисунок 8).

Рисунок 8.

The screenshot shows a terminal window titled "GNU nano 7.2" with the command "lab7-2.asm *". The assembly code is as follows:

```
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:
```

At the bottom of the terminal window, there is a menu bar with Russian keyboard shortcuts:

- $\wedge G$ Справка
- $\wedge O$ Записать
- $\wedge W$ Поиск
- $\wedge K$ Вырезать
- $\wedge T$ Выполнить
- $\wedge C$ Позиция
- $\wedge X$ Выход
- $\wedge R$ ЧитФайл
- $\wedge \backslash$ Замена
- $\wedge U$ Вставить
- $\wedge J$ Выровнять
- $\wedge /$ К строке

Программа которая выводит наибольшее число из 3.

Тест программы (рисунок 9).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$
```

Рисунок 9.

Выводит наибольшее число

Создание файла (рисунок 10).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рисунок 10.

Создали файл листинга.

2.1. Ответы на вопросы:

1. `jc fin` ; если '`max(A,C)>B`', то переход на '`fin`', :

Инструкция `jg` (jump if greater — переход если больше) выполняет условный переход, если результат предыдущего сравнения показал, что первое значение больше второго (при знаковом сравнении).

2. `mov ecx, [B] ; иначе 'ecx = B'` :

Эта инструкция выполняется только если переход `jg fin` не сработал, то есть когда $\max(A, C) \leq B$.

Значение переменной `B` загружается из памяти в регистр `ecx`.

3. `mov [max], ecx:`

Здесь значение из регистра `ecx` записывается в переменную `max`.

Поскольку в `ecx` находится `B`, это означает, что максимальным значением считается `B`.

Выходные файлы (рисунок 11).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst
lab7-1      lab7-1.o   lab7-2.asm  lab7-2.o
```

Рисунок 11.

3. Самостоятельная работа

Программа (рисунок 12).

```
GNU nano 7.2                               lab7-samrab1.asm *
%include 'in_out.asm'

SECTION .data
msg_result db 'Наименьшее число: ', 0
msg_test1  db 'Тест 1: a=54, b=62, c=87', 0
msg_test2  db 'Тест 2: разные комбинации', 0

a dd 54
b dd 62
c dd 87

SECTION .bss
min resd 1      ; 4 байта для числа

SECTION .text
GLOBAL _start

_start:
; Тест 1: a=54, b=62, c=87
mov eax, msg_test1
call sprintLF

; Находим минимум из a, b, c
mov eax, [a]          ; eax = a
mov [min], eax        ; min = a

; Сравниваем min и b
cmp eax, [b]
jle .compare_c        ; если a <= b, сравниваем с c
mov eax, [b]          ; иначе min = b
mov [min], eax

.compare_c:
mov eax, [min]        ; текущий минимум
cmp eax, [c]
jle .print_result1   ; если min <= c, оставляем
mov eax, [c]          ; иначе min = c

^G Справка  ^O Записать  ^W Поиск  ^K Вырезать  ^T Выполнить  ^C Позиция
^X Выход    ^R ЧитФайл  ^\ Замена   ^U Вставить  ^J Выровнять  ^/ К строке
```

Рисунок 12.

Это программа нахождения наименьшей из 3 целочисленных переменных a, b и C

Тест программы (рисунок 13).

```
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-samrab1.asm
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-samrab1 lab7-samrab1.o
rbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-samrab1
Тест 1: a=54, b=62, c=87
Наименьшее число: 54
```

Рисунок 13.

Запуск программы. Вывод наименьшего числа.

Программа (рисунок 14).

```
GNU nano 7.2                               lab7-samrab2.asm *
%include 'in_out.asm'

SECTION .data
msg_x db 'x: ', 0
msg_a db 'a: ', 0
result db 'f(x)=', 0

SECTION .bss
x resb 10
a resb 10

SECTION .text
GLOBAL _start

_start:
; Ввод x
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 10
call sread

; Ввод a
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 10
call sread

; Преобразование
mov eax, x
call atoi
push eax
mov eax, a
call atoi
mov ebx, eax
pop eax

; Вычисление
mov eax, ebx
mul eax, a
mov ebx, eax
add ebx, result
mov eax, ebx
call sprint

; Вывод результата
mov eax, result
call sprint

; Выход
mov eax, 0
call exit

^G Справка   ^O Записать   ^W Поиск   ^K Вырезать   ^T Выполнить   ^C Позиция
^X Выход   ^R ЧитФайл   ^\ Замена   ^U Вставить   ^J Выровнять   ^/ К строке
```

Рисунок 14.

Программа, которая для введенных с клавиатуры значений x и a вычисляет

значение заданной функции $f(x)$ и выводит результат вычислений.

Тест программы (рисунок 14).

```
rmbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ nasm -f elf lab7-samrab2.asm
rmbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-samrab2 lab7-samrab2.o
rmbarkhaneva1@Redmi-Book:~/work/arch-pc/lab07$ ./lab7-samrab2
x: 10
a: 5
f(x)=10
```

Рисунок 14.

Создали исполняемый файл и проверили на корректность.

4. Вывод

В ходе лабораторной работы были изучены команды условного и безусловного переходов в ассемблере NASM. Приобретены навыки написания программ с использованием инструкций jmp для безусловных переходов и различных вариантов jcc (например, je, jne, jq) для реализации ветвлений на основе состояния флагов процессора. Изучена работа регистра флагов и команды сравнения cmp, которая устанавливает эти флаги для последующего

анализа. Также освоена структура файла листинга, создаваемого транслятором, и его роль в отладке программ, что позволяет анализировать соответствие исходного кода сгенерированному машинному коду и смещениям.