

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

****Факультет физико-математических и естественных наук****

****Кафедра прикладной информатики и теории вероятностей****

ОТЧЕТ

по лабораторной работе № 7

****дисциплина:**** Архитектура компьютера

****Студент:**** ванриму Брайан киарис

****Группа:**** НКАбл - 01- 25

****МОСКВА****

2025 г.

СОДЕРЖАНИЕ

1. Цель работы
2. Задание
3. Теоретическое введение
4. Выполнение лабораторной работы
5. Задание для самостоятельной работы
6. Выводы

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 Основные задания

1. Создать программу с использованием инструкции безусловного перехода jmp
2. Создать программу с использованием инструкций условного перехода стр и jg
3. Изучить структуру файла листинга
4. Создать файл листинга и проанализировать его содержимое

2.2 Задания для самостоятельной работы

1. Написать программу нахождения наименьшей из 3 целочисленных переменных
2. Написать программу вычисления значения функции $f(x)$ для введенных x и a

3 Теоретическое введение

3.1 Команды переходов

- ****Безусловный переход**** - выполнение передачи управления в определенную точку программы без каких-либо условий (инструкция jmp)
- ****Условный переход**** - выполнение или не выполнение перехода в зависимости от проверки условия (инструкции je, jne, js, jl и др.)

3.2 Регистр флагов

Флаги отражают результат выполнения арифметических инструкций:

- CF - флаг переноса
- ZF - флаг нуля
- SF - флаг знака
- OF - флаг переполнения

3.3 Инструкция cmp

Инструкция cmp сравнивает два операнда и устанавливает флаги в зависимости от результата сравнения.

4 Выполнение лабораторной работы

4.1 Программа с безусловными переходами

```
_zsh ➤ lab07 0% main ↑1 ?98 282ms
>> ./lab-7
Сообщение № 2
Сообщение № 3
_zsh ➤ lab07 0% main ↑1 ?98 22ms
>> touch lab7-2.asm
_zsh ➤ lab07 0% main ↑1 ?98 73ms
>> nano lab7-2.asm
_zsh ➤ lab07 0% main ↑1 ?98 3m 14s 960ms
>> nasm -f elf32 lab7-2.asm -o lab7-2.o
nasm: fatal: unable to open input file `lab7-2.asm' No such file or directory
_zsh ➤ lab07 0% main ↑1 ?98 29ms
>> nasm -f elf32 lab7-2.asm -o lab7-2.o
_zsh ➤ lab07 0% main ↑1 ?98 39ms
>> ld -m elf_i386 lab7-2.o -o lab7-2
_zsh ➤ lab07 0% main ↑1 ?98 30ms
>> ./lab7-2
Сообщение № 2
Сообщение № 1
_zsh ➤ lab07 0% main ↑1 ?98 20ms
>> 
```

```
GNU nano 8.6 lab-7.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
```

****Вывод:**** Обе программы демонстрируют работу инструкций безусловного перехода `jmp` в ассемблере. Первая программа показывает базовое использование прыжков для изменения порядка выполнения инструкций, вторая - более сложную логику с цепочкой переходов для управления последовательностью вывода сообщений. Программы успешно выполняются и выводят ожидаемые результаты, подтверждая правильность использования меток и инструкций перехода в NASM.

4.2 Программа с условными переходами (lab7-2.asm)

Программа демонстрирует использование инструкций безусловного перехода jmp для изменения порядка выполнения команд. Она использует метки и переходы между ними для управления последовательностью вывода сообщений.

****Логика выполнения:****

- Программа начинается с прыжка на _label2 (пропуская первый вывод)
- Выводит "Сообщение № 2"
- Переходит на _label1
- Выводит "Сообщение № 1"
- Завершает работу

```
23:01
Terminal
File Edit View Search Terminal Help
> zsh lab07 0P main f1 798 73ms
> nano lab7-2.asm
> zsh lab07 0P main f1 798 3m 14s 960ms
> nasm -f elf32 lab7-2.asm -o lab7-2.o
nasm: fatal: unable to open input file 'lab7-2.asm' No such file or directory
> zsh lab07 0P main f1 798 29ms
> nasm -f elf32 lab7-2.asm -o lab7-2.o
> zsh lab07 0P main f1 798 39ms
> ld -m elf_i386 lab7-2.o -o lab7-2
> zsh lab07 0P main f1 798 30ms
> ./lab7-2
Сообщение № 2
Сообщение № 1
> zsh lab07 0P main f1 798 20ms
> nano lab7-2.asm
> zsh lab07 0P main f1 798 5s 202ms
> touch lab7-3-1.asm
> zsh lab07 0P main f1 798 73ms
> nano lab7-3-1.asm
> zsh lab07 0P main f1 798 11s 0ms
> ld -f elf_i386 lab7-3-1.asm -o lab7-3-1.o
ld: -f may not be used without -shared
> zsh lab07 0P main f1 798 29ms
> nasm -f elf32 lab7-3-1.asm -o lab7-3-1.o
> zsh lab07 0P main f1 798 41ms
> ld -m elf_i386 lab7-3-1.o -o lab7-3-1
> zsh lab07 0P main f1 798 33ms
> ./lab7-3-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
> zsh lab07 0P main f1 798 22ms
>
```

```
22:59
Terminal
File Edit View Search Terminal Help
GNU nano 8.6 lab7-3-1.asm Modified
%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 ; Переход к выводу третьего сообщения

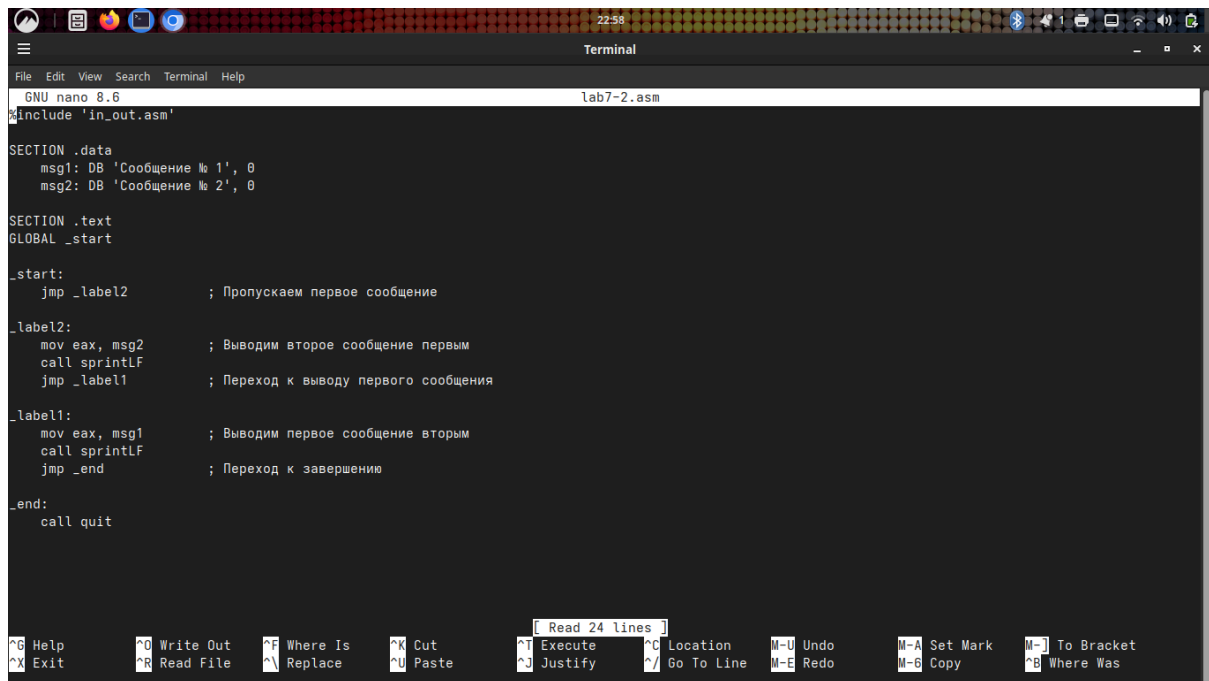
_label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 1'
    jmp _end ; Завершение программы

_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 2'
    jmp _label1 ; Переход к выводу первого сообщения

_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 3'
    jmp _label2 ; Переход к выводу второго сообщения

_end:

^G Help ^O Write Out ^F Where Is ^X Cut ^T Execute ^C Location ^M-U Undo ^M-A Set Mark ^M-] To Bracket
^X Exit ^R Read File ^_ Replace ^U Paste ^_ Justify ^_ Go To Line ^M-E Redo ^M-B Copy ^_B Where Was
```



```
GNU nano 8.6 lab7-2.asm
#include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1', 0
msg2: DB 'Сообщение № 2', 0

SECTION .text
GLOBAL _start

_start:
    jmp _label2      ; Пропускаем первое сообщение

_label2:
    mov eax, msg2    ; Выводим второе сообщение первым
    call sprintf
    jmp _label1      ; Переход к выводу первого сообщения

_label1:
    mov eax, msg1    ; Выводим первое сообщение вторым
    call sprintf
    jmp _end         ; Переход к завершению

_end:
    call quit
```

****Вывод:**** Программа наглядно демонстрирует, как инструкции `jmp` позволяют произвольно изменять поток выполнения, обеспечивая гибкое управление последовательностью операций в ассемблере.

****Основной код программы:****

Программа находит максимальное число из трех значений: A=20, C=50 и введенного пользователем числа B.

****Примеры работы:****

- ****Если ввести 21:****

Сравнивает 20, 50 и 21 → наибольшее 50

Вывод: Наибольшее число: 50

- ****Если ввести 56:****

Сравнивает 20, 50 и 56 → наибольшее 56

Вывод: Наибольшее число: 56

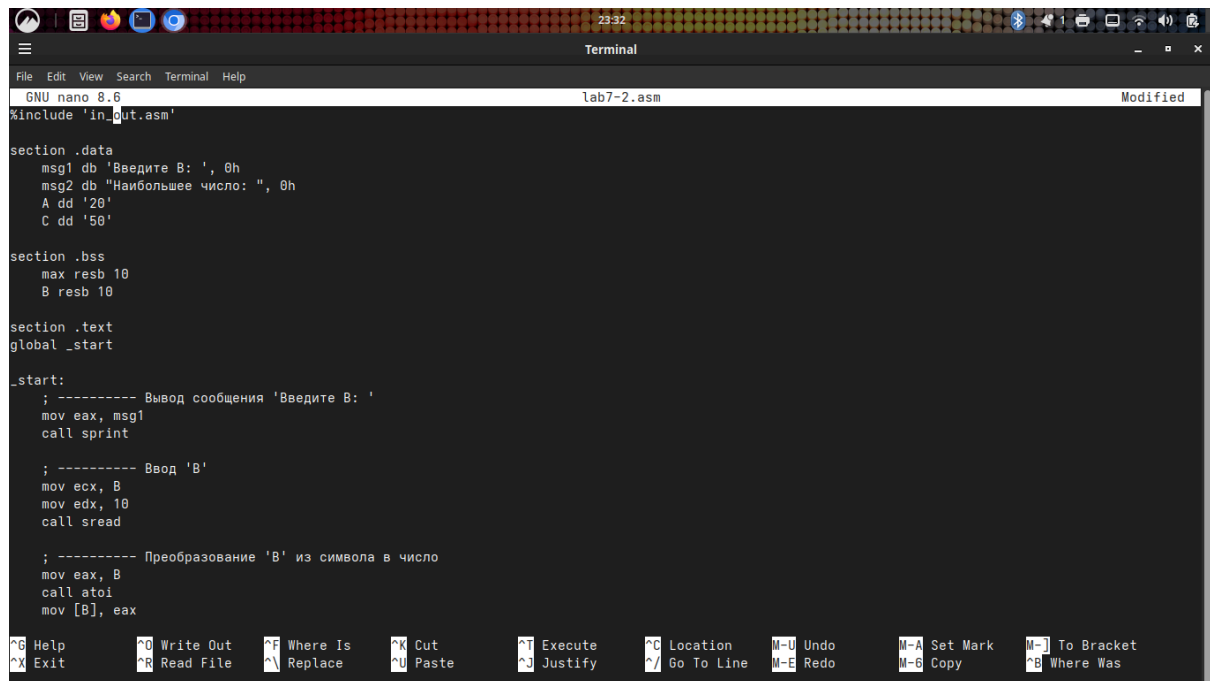
- ****Если ввести 10:****

Сравнивает 20, 50 и 10 → наибольшее 50

Вывод: Наибольшее число: 50

```
23:38
Terminal
File Edit View Search Terminal Help
>_zsh lab07 0P main ↑1 ?98 17s 273ms
>> nasm -f elf32 lab7-2.asm -o lab7-2.o
>_zsh lab07 0P main ↑1 ?98 34ms
>>
>_zsh lab07 0P main ↑1 ?98 34ms
>> ld -m elf_i386 lab7-2.o -o lab7-2
>_zsh lab07 0P main ↑1 ?98 127ms
>> ./lab7-2
Введите B: 21
Наибольшее число: 50
>_zsh lab07 0P main ↑1 ?98 4s 104ms
>> ./lab7-2
Введите B: 76
Наибольшее число: 76
>_zsh lab07 0P main ↑1 ?98 2s 424ms
>> ./lab7-2
Введите B: 28
Наибольшее число: 50
>_zsh lab07 0P main ↑1 ?98 3s 788ms
>> ./lab7-2
Введите B: 12
Наибольшее число: 50
>_zsh lab07 0P main ↑1 ?98 3s 556ms
>> ./lab7-2
Введите B: 56
Наибольшее число: 56
>_zsh lab07 0P main ↑1 ?98 3s 540ms
>>
```

```
>_zsh lab07 0P main ↑1 ?98 17s 273ms
>> nasm -f elf32 lab7-2.asm -o lab7-2.o
>_zsh lab07 0P main ↑1 ?98 34ms
>>
>_zsh lab07 0P main ↑1 ?98 34ms
>> ld -m elf_i386 lab7-2.o -o lab7-2
>_zsh lab07 0P main ↑1 ?98 127ms
>> ./lab7-2
Введите B: 21
Наибольшее число: 50
>_zsh lab07 0P main ↑1 ?98 4s 104ms
>>
```



```
GNU nano 8.6 lab7-2.asm Modified
#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
    mov [B], eax

^G Help      ^O Write Out  ^F Where Is   ^X Cut        ^T Execute    ^C Location   ^U Undo       ^A Set Mark   ^] To Bracket
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line ^E Redo       ^M Copy       ^B Where Was
```

****Итог:**** Программа сравнивает введенное число B с предустановленными значениями 20 и 50, и показывает максимальное из всех трех чисел.

4.3 Файл листинга

****Создание файла листинга:****

```
```bash
nasm -f elf -I lab7-2.lst lab7-2.asm
cat lab7-2.lst
```