Лабораторная работа №7

Дисциплина: Архитектура компьютера

Савостин Олег

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление:

jmp

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания:

cmp ,

Команда условного перехода имеет вид j label Мнемоника перехода связана со значением анализируемых флагов или со способом фор- мирования этих флагов

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

Сперва создаю нужный для работы файл lab7-1.asm(рис. 1).

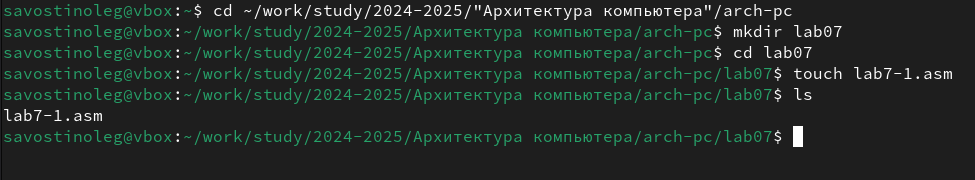


Рис. 1: Создание нужного файла.

Вставляю в него листинг 7.1 из файла на ТУИС(рис. 2). Создаю исполняемый файл и запускаю его. (рис. 3).

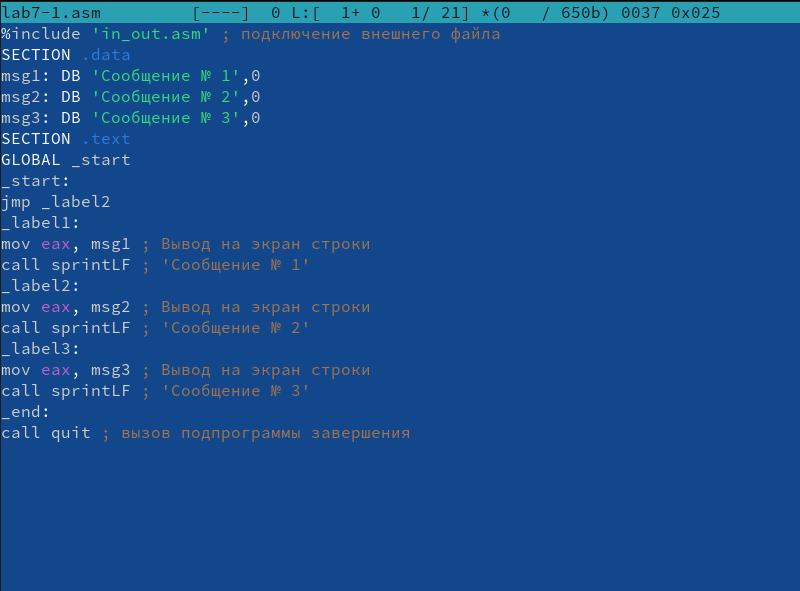


Рис. 2: Текст Листинга в файле

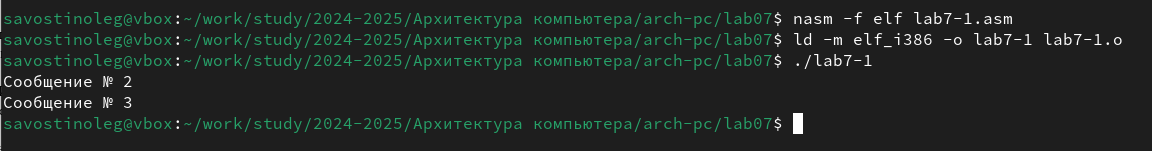


Рис. 3: Исполняемый файл Листинга 7.1

Изменяю текст файла так, чтобы выводились: 2, 1. Вставляю текст в файл из Листинга 7.2 (рис. 4). Создаю исполняемый файл и запускаю его. (рис. 5).

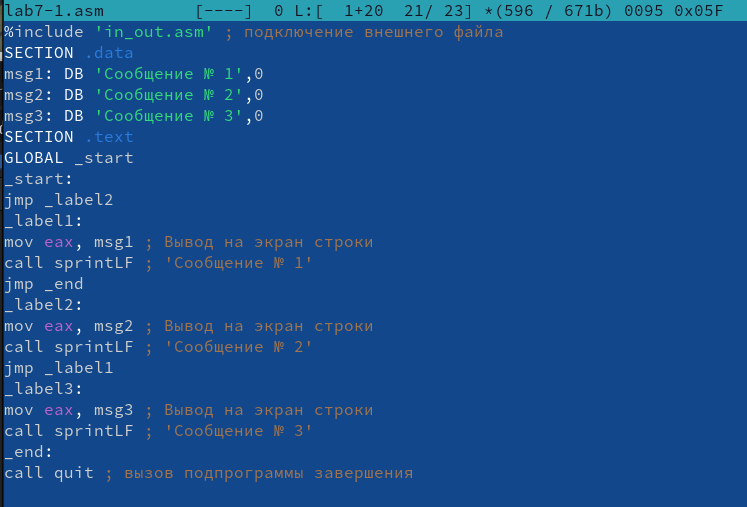


Рис. 4: Текст Листинга 7.2 в файле

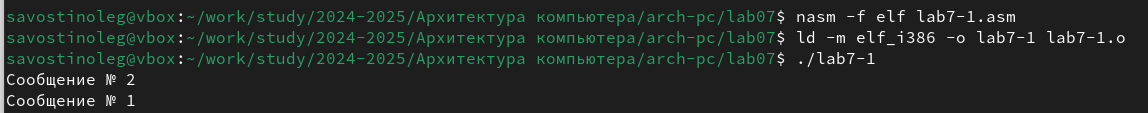


Рис. 5: Исполняемый файл Листинга 7.2

Теперь изменяю файл так, чтобы выводились все сообщения по порядку возрастания. Ввожу код (рис. 6). Затем создаю исполняемый файл и проверяю на правильность (рис. 7).

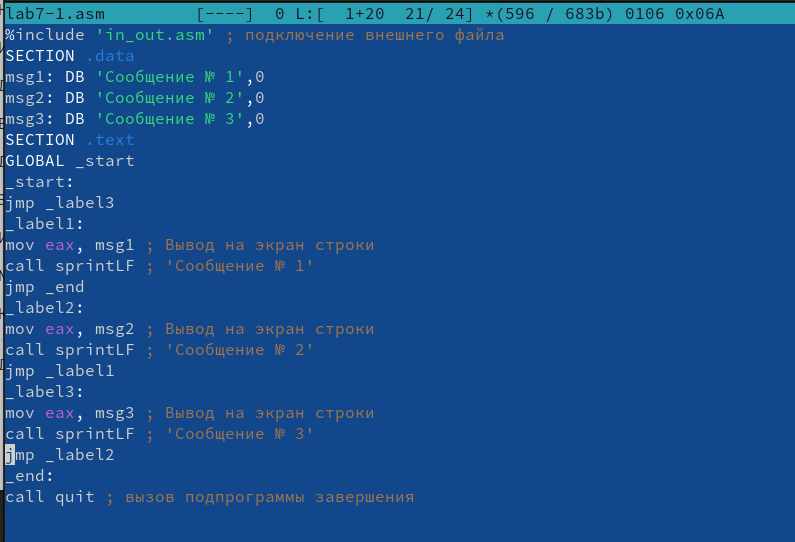


Рис. 6: Новый код.

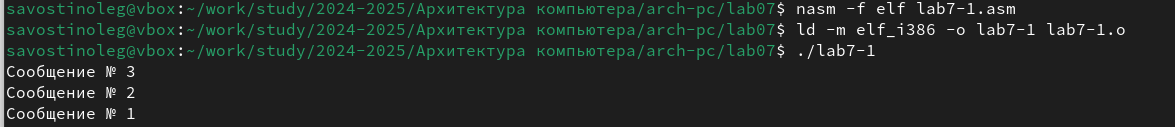


Рис. 7: Проверка на правильность кода

Создаю новый файл и вставляю в него текст из Листинга 7.3 (рис. 8).

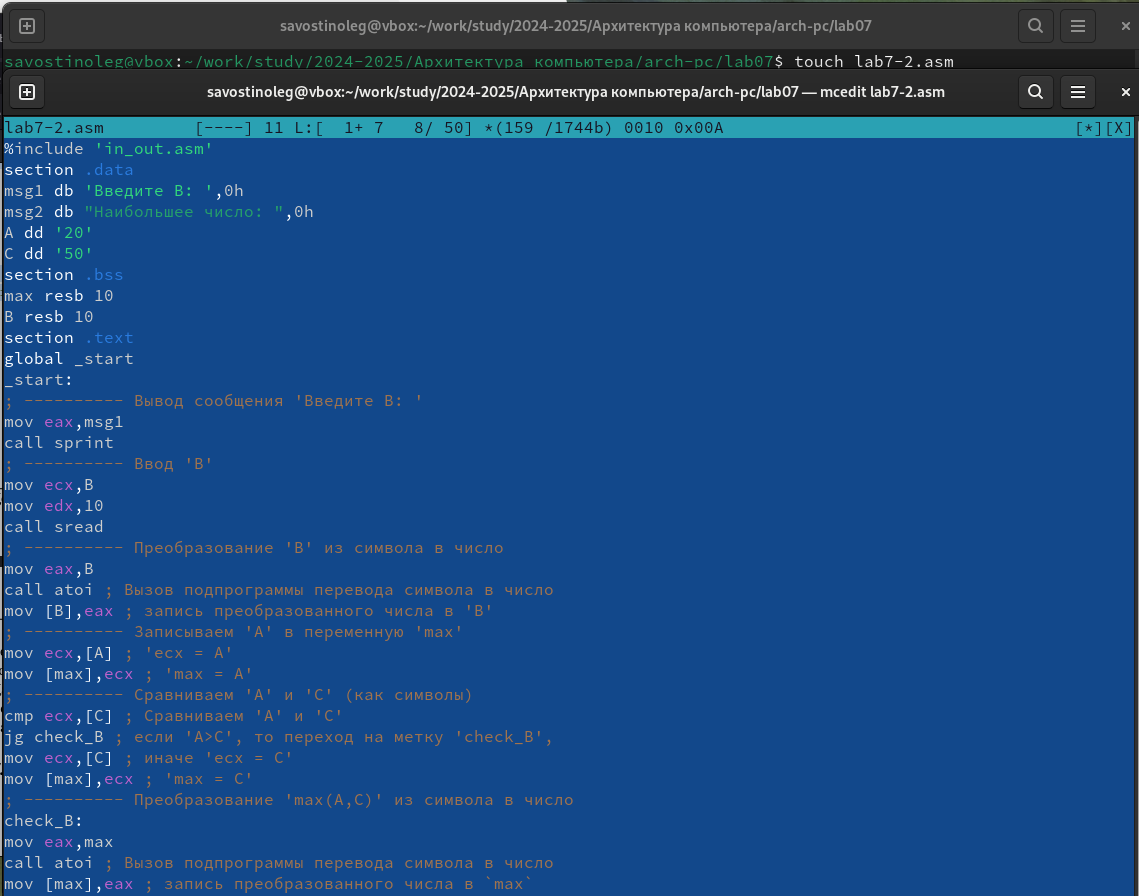


Рис. 8: Lab7-3.asm и текст из Листинга 7.3

Создаю исполняемый файл(рис. 9) и проверяю что он делает (рис. 10).

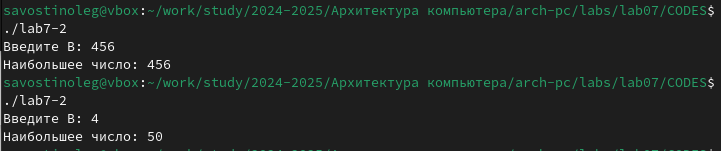


Рис. 9: Создание исполняемого файла

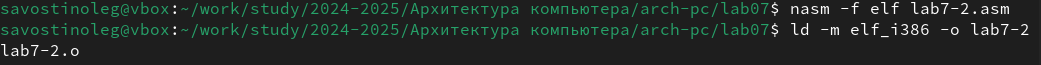


Рис. 10: Запуск данного файла

Для углубленного изучения файла, я создаю lab7-2.lst и изучаю содержимое(рис. 11).

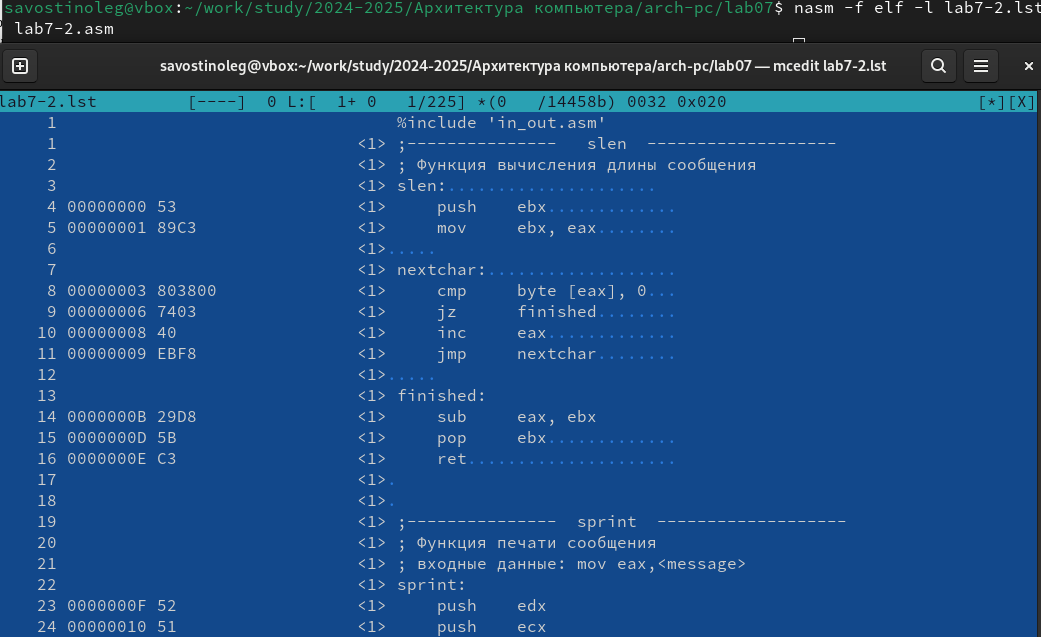


Рис. 11: lab7-2.lst

В данной строчке указан регистр на строчке 21. Её адрес - 00000101. Машинный код - В8[0A000000]. строчка содержит код mov eax,B, который переводит значение В в регистр (рис. 12).

Рис. 12: Первая строчка.

Рис. 12: Первая строчка.

В данной строчке указан регистр на строчке 23. Её адрес - 0000010В. Машинный код - А3[0A000000]. строчка содержит код mov [B], который переводит значение регистра в [B] (рис. 13).

Рис. 13: Вторая строчка.

Рис. 13: Вторая строчка.

В данной строчке указан регистр на строчке 22. Её адрес - 00000106. Машинный код - E891FFFFFF. строчка содержит код call atoi который переводит символ, лежащий выше, в число(рис. 14).

Рис. 14: Третья строчка.

Рис. 14: Третья строчка.

Удаляю строчку mov ecx,[max] (рис. 15). Пытаюсь создать исполняемый файл и .lst и получаю ошибку (рис. 16). Создается только lst. Открываю lst файл и нахожу ошибку, которая добавилась.(рис. 17).

Рис. 15: Строчка, которая будет удалена

Рис. 15: Строчка, которая будет удалена

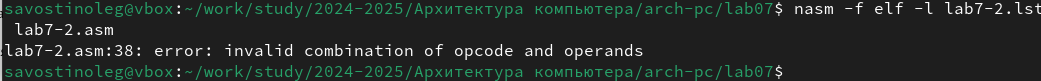


Рис. 16: Попытка создать исполняемый файл

Рис. 17: Ошибка в листинге

Рис. 17: Ошибка в листинге

## 4.2 Выполнение заданий для самостоятельной работы

1. Сперва, пишу код который будет находить наименьшее число.(рис. 18).

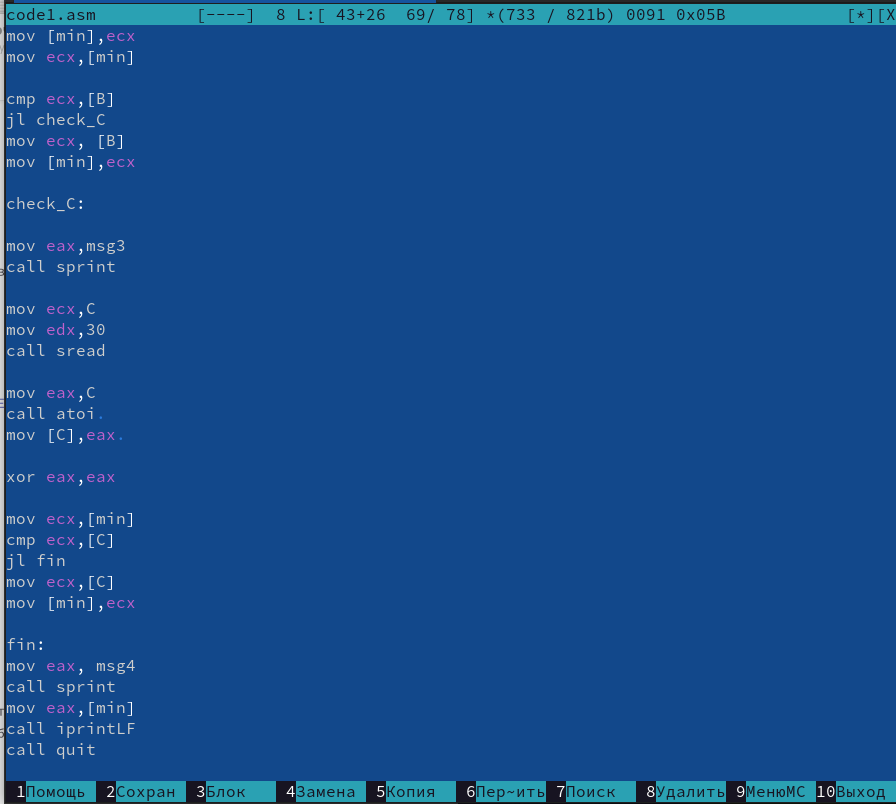


Рис. 18: Первый код

Создаю исполняемый, всё верно. (рис. 19).

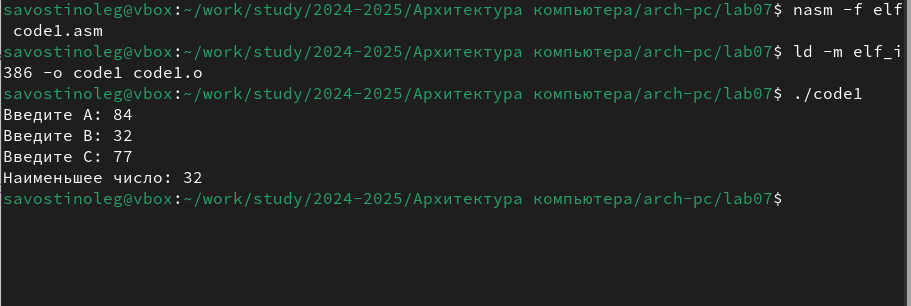


Рис. 19: Запуск исполняемого файла

КОД %include ‘in\_out.asm’

section .data

msg1 db ‘Введите A:’,0h

msg2 db ‘Введите B:’,0h

msg3 db ‘Введите C:’,0h

msg4 db “Наименьшее число:”,0h

section .bss

min resb 30

A resb 30

B resb 30

C resb 30

section .text

global \_start

\_start:

mov eax,msg1

call sprint

mov ecx,A

mov edx,30

call sread

mov eax,A

call atoi

mov [A],eax

xor eax,eax

mov eax,msg2

call sprint

mov ecx,B

mov edx,30

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]

jl check\_C

mov ecx, [B]

mov [min],ecx

check\_C:

mov eax,msg3

call sprint

mov ecx,C

mov edx,30

call sread

mov eax,C

call atoi

mov [C],eax

xor eax,eax

mov ecx,[min]

cmp ecx,[C]

jl fin

mov ecx,[C]

mov [min],ecx

fin:

mov eax, msg4

call sprint

mov eax,[min]

call iprintLF

call quit

1. Теперь записываю код для следующей функции: a-7 a>=7 | ax a<7 (рис. 20).

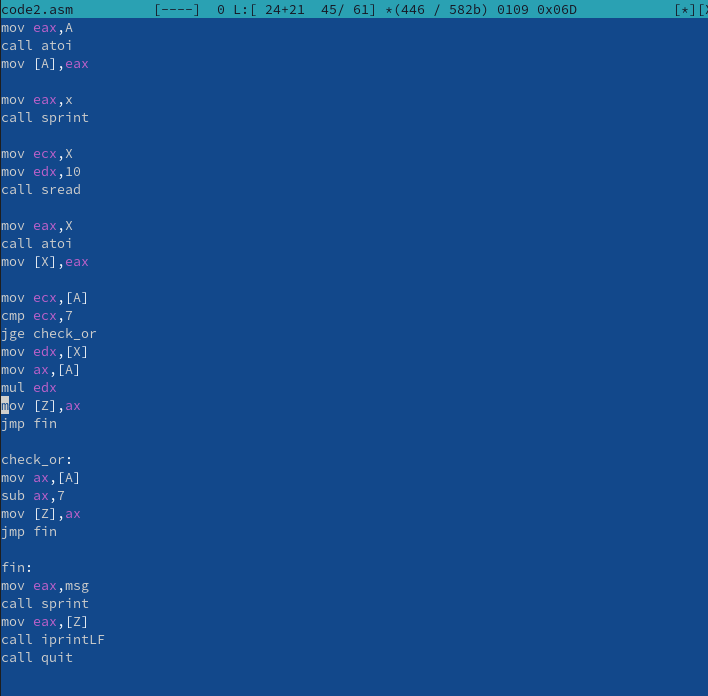


Рис. 20: Код второй

Проверяю на правильность написанного кода. Всё верно (рис. 21).

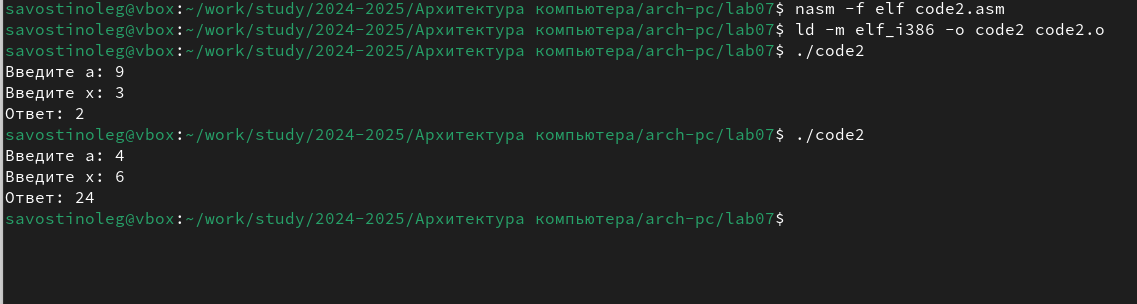


Рис. 21: Запуск исполняемого файла второго кода

КОД

%include ‘in\_out.asm’

section .data

a db ‘Введите a:’,0h

x db ‘Введите x:’,0h

msg db “Ответ:”,0h

section .bss

X RESB 10

A RESB 10

Z RESB 10

section .text

global \_start

\_start:

mov eax,a

call sprint

mov ecx,A

mov edx,10

call sread

mov eax,A

call atoi

mov [A],eax

mov eax,x

call sprint

mov ecx,X

mov edx,10

call sread

mov eax,X

call atoi

mov [X],eax

mov ecx,[A]

cmp ecx,7

jge check\_or mov edx,[X]

mov ax,[A]

mul edx

mov [Z],ax

jmp fin

check\_or:

mov ax,[A]

sub ax,7

mov [Z],ax

jmp fin

fin:

mov eax,msg

call sprint

mov eax,[Z]

call iprintLF

call quit

# 5 Вывод.

В заключении я изучил команды условного и безусловного переходов. Приобрел навыков написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.

# Список литературы

[Лабораторная работа №7](https://esystem.rudn.ru/pluginfile.php/2089545/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9..pdf)

::: ::