Отчёт по лабораторной работе 7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений

Тукаев Тимур Ильшатович НММбд-03-23

Содержание

# 1 Цель работы

Целью работы является 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. Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp.

Написал в файл lab7-1.asm текст программы из листинга 7.1.

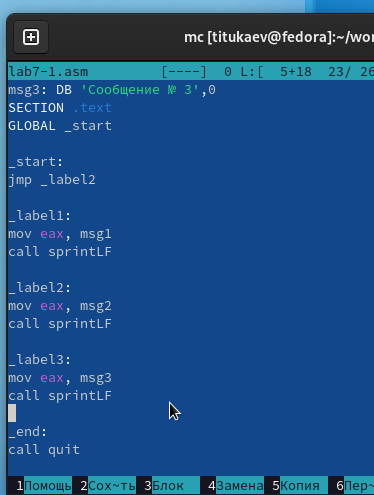


Figure 1: Программа в файле lab7-1.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
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  
  
\_label2:  
mov eax, msg2  
call sprintLF  
  
\_label3:  
mov eax, msg3  
call sprintLF  
  
\_end:  
call quit

Создал исполняемый файл и запустил его.

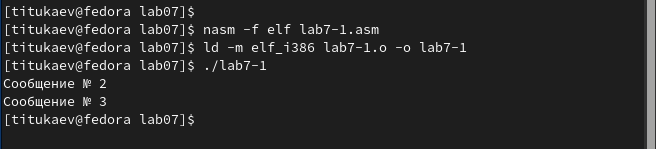


Figure 2: Запуск программы lab7-1.asm

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим 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).

Изменил текст программы в соответствии с листингом 7.2.

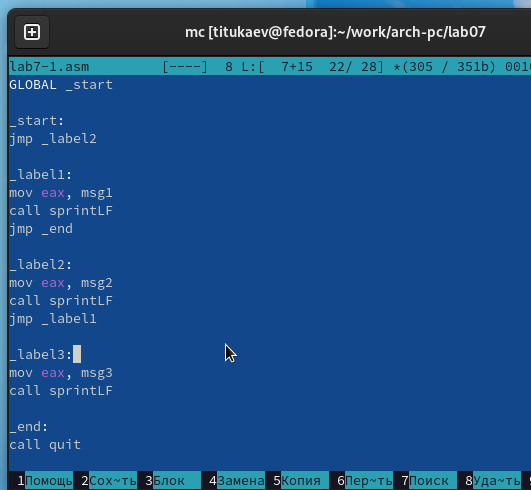


Figure 3: Программа в файле lab7-1.asm:

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
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  
  
\_label2:  
mov eax, msg2  
call sprintLF  
jmp \_label1  
  
\_label3:  
mov eax, msg3  
call sprintLF  
  
\_end:  
call quit

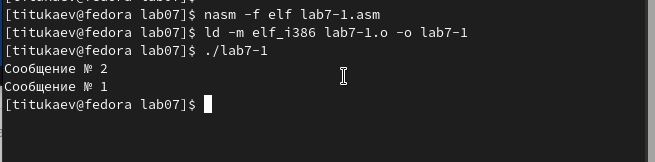


Figure 4: Запуск программы lab7-1.asm:

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим:

Сообщение № 3  
Сообщение № 2  
Сообщение № 1

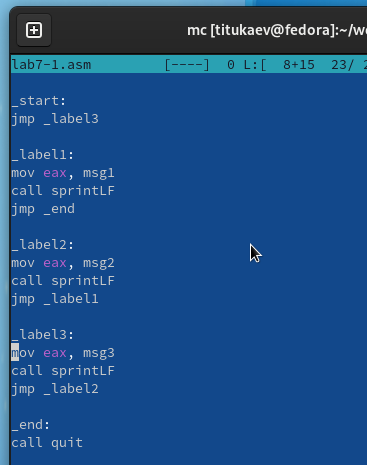


Figure 5: Программа в файле lab7-1.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
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  
  
\_label2:  
mov eax, msg2  
call sprintLF  
jmp \_label1  
  
\_label3:  
mov eax, msg3  
call sprintLF  
jmp \_label2  
  
\_end:  
call quit

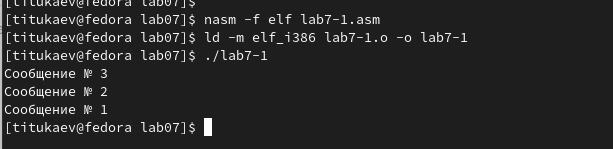


Figure 6: Запуск программы lab7-1.asm

1. Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.

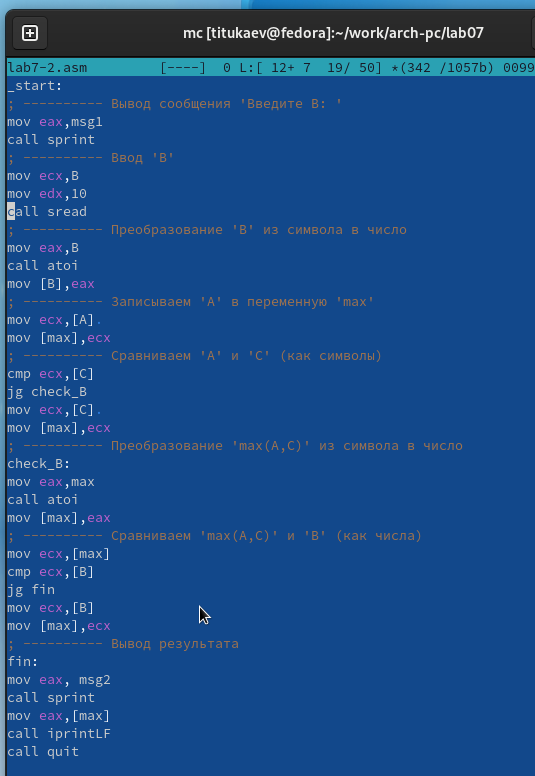


Figure 7: Программа в файле lab7-2.asm

Также размещаю код программы в отчете.

%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  
; ---------- Записываем 'A' в переменную 'max'  
mov ecx,[A]   
mov [max],ecx  
; ---------- Сравниваем 'A' и 'С' (как символы)  
cmp ecx,[C]  
jg check\_B  
mov ecx,[C]   
mov [max],ecx  
; ---------- Преобразование 'max(A,C)' из символа в число  
check\_B:  
mov eax,max  
call atoi  
mov [max],eax  
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)  
mov ecx,[max]  
cmp ecx,[B]  
jg fin  
mov ecx,[B]  
mov [max],ecx  
; ---------- Вывод результата  
fin:  
mov eax, msg2  
call sprint  
mov eax,[max]  
call iprintLF  
call quit

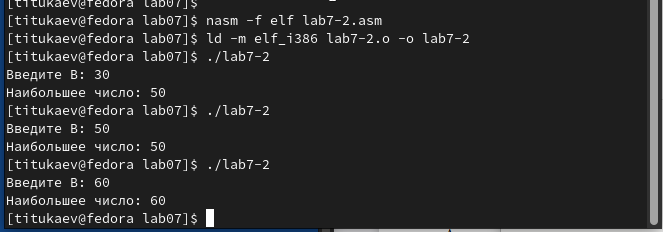


Figure 8: Запуск программы lab7-2.asm

1. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла lab7-2.asm

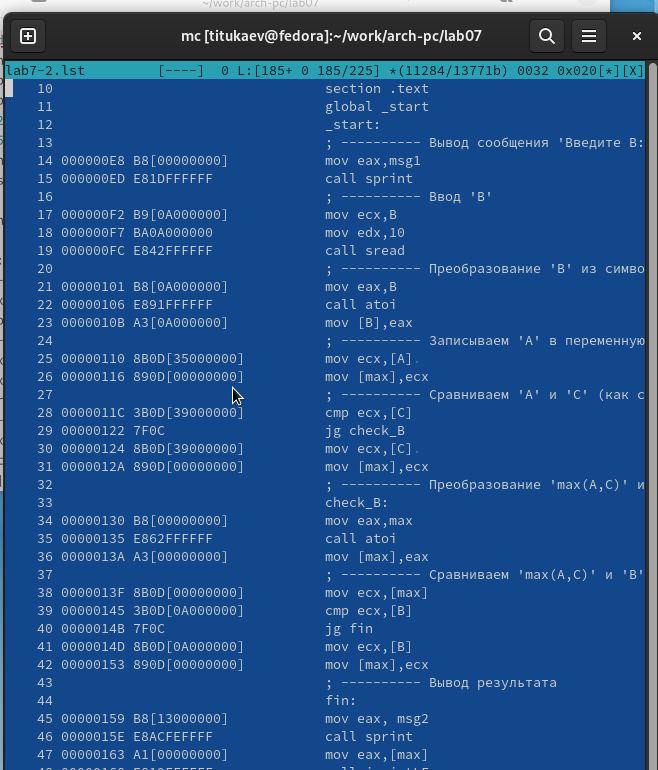


Figure 9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 21

* 21 - номер строки
* 00000101 - адрес
* B8[0A000000] - машинный код
* mov eax,B - код программы

строка 22

* 22 - номер строки
* 00000106 - адрес
* E891FFFFFF - машинный код
* call atoi- код программы

строка 23

* 23 - номер строки
* 0000010B - адрес
* A3[0A000000] - машинный код
* mov [B],eax - код программы

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.

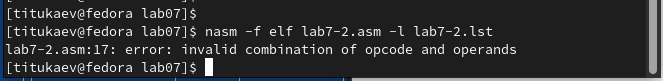


Figure 10: Ошибка трансляции lab7-2

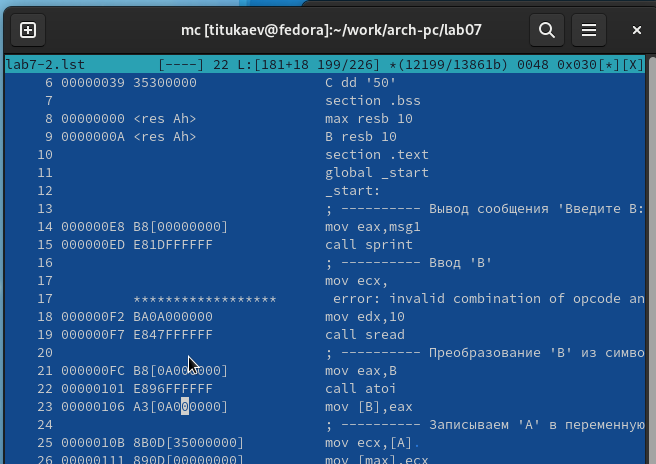


Figure 11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c.  Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 5 - 54, 62, 87

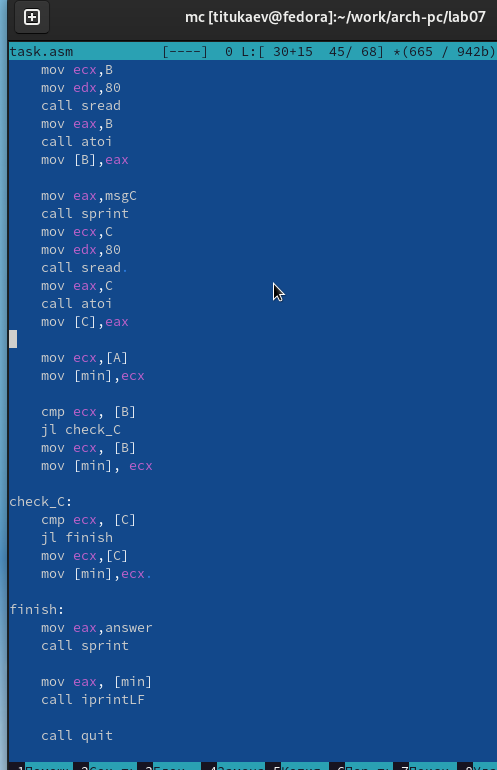


Figure 12: Программа в файле task.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
 msgA: DB 'Input A: ',0  
 msgB: DB 'Input B: ',0  
 msgC: DB 'Input C: ',0  
 answer: DB 'Smallest: ',0  
  
SECTION .bss  
 A: RESB 80  
 B: RESB 80  
 C: RESB 80  
 result: RESB 80  
 min: RESB 80  
  
SECTION .text  
 GLOBAL \_start  
  
\_start:  
 mov eax,msgA  
 call sprint  
 mov ecx,A  
 mov edx,80  
 call sread  
 mov eax,A  
 call atoi   
 mov [A],eax  
  
 mov eax, msgB  
 call sprint  
 mov ecx,B  
 mov edx,80  
 call sread  
 mov eax,B  
 call atoi  
 mov [B],eax  
  
 mov eax,msgC  
 call sprint  
 mov ecx,C  
 mov edx,80  
 call sread   
 mov eax,C  
 call atoi  
 mov [C],eax   
   
 mov ecx,[A]  
 mov [min],ecx  
  
 cmp ecx, [B]  
 jl check\_C  
 mov ecx, [B]  
 mov [min], ecx  
  
check\_C:  
 cmp ecx, [C]  
 jl finish  
 mov ecx,[C]  
 mov [min],ecx   
  
finish:  
 mov eax,answer  
 call sprint  
  
 mov eax, [min]  
 call iprintLF  
  
 call quit

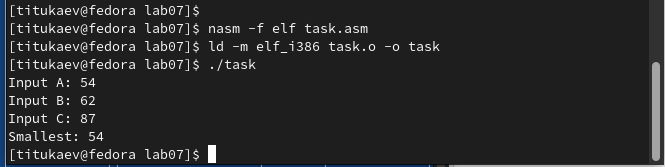


Figure 13: Запуск программы task.asm

1. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6.

для варианта 5

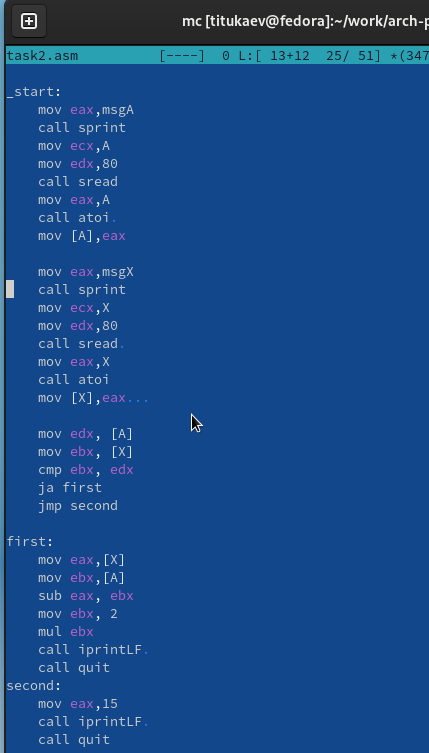


Figure 14: Программа в файле task2.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
 msgA: DB 'Input A: ',0  
 msgX: DB 'Input X: ',0  
  
SECTION .bss  
 A: RESB 80  
 X: RESB 80  
 result: RESB 80  
  
SECTION .text  
 GLOBAL \_start  
  
\_start:  
 mov eax,msgA  
 call sprint  
 mov ecx,A  
 mov edx,80  
 call sread  
 mov eax,A  
 call atoi   
 mov [A],eax  
  
 mov eax,msgX  
 call sprint  
 mov ecx,X  
 mov edx,80  
 call sread   
 mov eax,X  
 call atoi  
 mov [X],eax   
  
 mov edx, [A]  
 mov ebx, [X]  
 cmp ebx, edx  
 ja first  
 jmp second  
  
first:  
 mov eax,[X]  
 mov ebx,[A]  
 sub eax, ebx  
 mov ebx, 2  
 mul ebx  
 call iprintLF   
 call quit  
second:  
 mov eax,15  
 call iprintLF   
 call quit

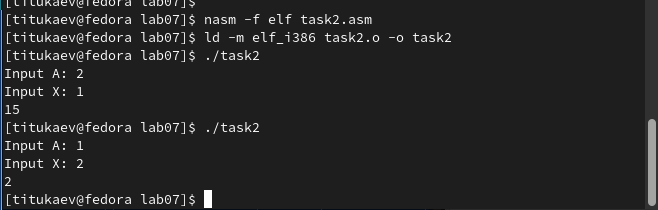


Figure 15: Запуск программы task2.asm

# 3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.