Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Кузьмина Мария Константиновна

Содержание

# 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

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

# 3 Выполнение лабораторной работы

## 3.1 Реализация переходов в NASM

Создаем директорию с помощью mkdir, переходим в нее и создаем файл lab7-1.asm (рис. 1):

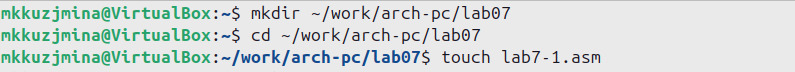


Рис. 1: снимок экрана

Вводим в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2):

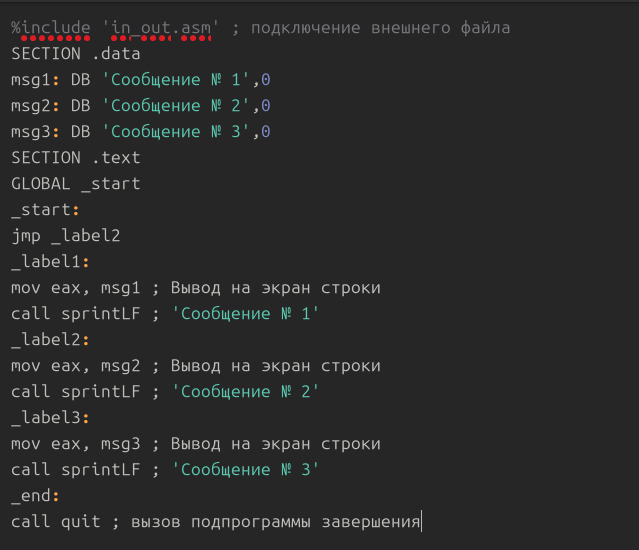


Рис. 2: снимок экрана

Создаем исполняемый файл и запускаем его. Результат работы данной программы будет следующим (рис. 3):

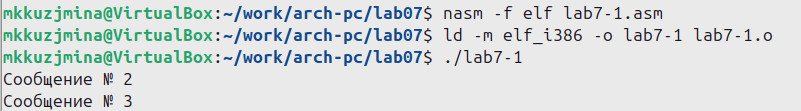


Рис. 3: снимок экрана

Изменяем 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). (рис. 4):

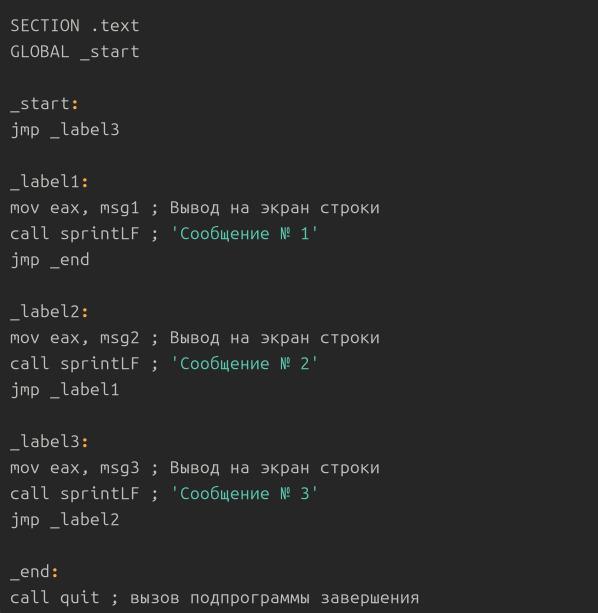


Рис. 4: снимок экрана

%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 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 1'  
jmp \_end  
  
\_label2:  
mov eax, msg2 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 2'  
jmp \_label1  
  
\_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintLF ; 'Сообщение № 3'  
jmp \_label2  
  
\_end:  
call quit ; вызов подпрограммы завершения

(рис. 5):

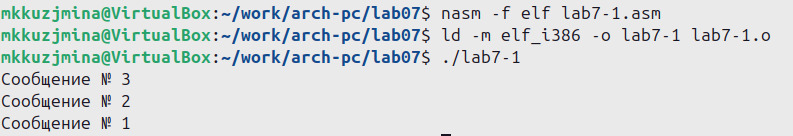


Рис. 5: снимок экрана

Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и вводим в lab7-2.asm текст из листинга 7.3. (рис. 6):

Рис. 6: снимок экрана

Рис. 6: снимок экрана

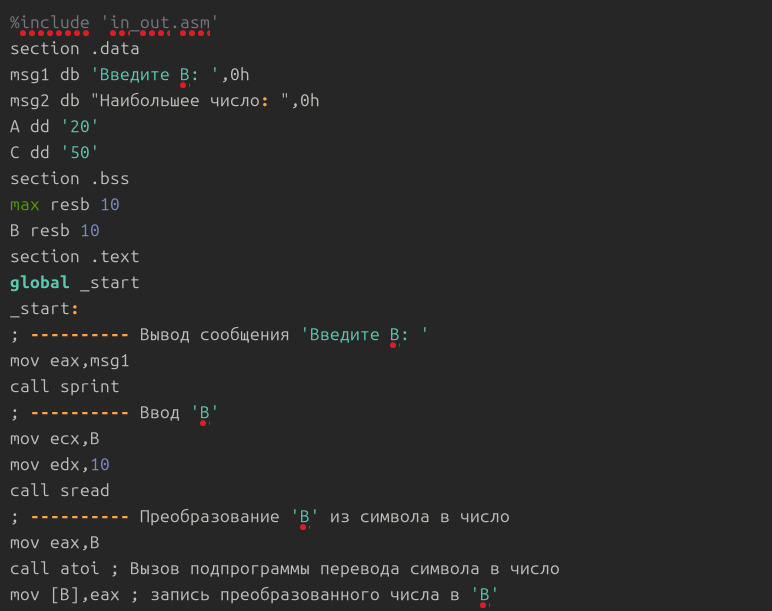


Рис. 7: снимок экрана

Вводим значение В и проверяем правильность выполнения (рис. 8):

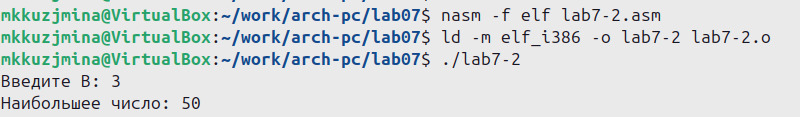


Рис. 8: снимок экрана

## 3.2 Изучение структуры файла листинга

Создаем файл листинга для программы из файла lab7-2.asm, открываем файл листинга lab7-2.lst с помощью текстового редактора mousepad и в инструкции с двумя операндами удаляем один операнд. (рис. 9):

Рис. 9: снимок экрана

Рис. 9: снимок экрана

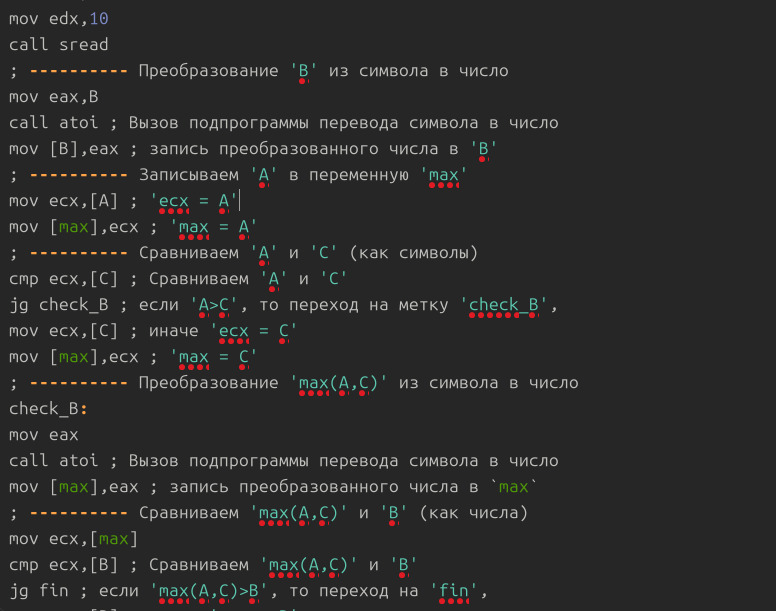


Рис. 10: снимок экрана

%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 'С' (как символы)  
cmp ecx,[C] ; Сравниваем 'A' и 'С'  
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  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [max],eax ; запись преобразованного числа в `max`  
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)  
mov ecx,[max]  
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'  
jg fin ; если 'max(A,C)>B', то переход на 'fin',  
mov ecx,[B] ; иначе 'ecx = B'  
mov [max],ecx  
; ---------- Вывод результата  
fin:  
mov eax, msg2  
call sprint ; Вывод сообщения 'Наибольшее число: '  
mov eax,[max]  
call iprintLF ; Вывод 'max(A,B,C)'  
call quit ; Выход

Выполняем трансляцию с получением файла листинга, получаем ошибку. (Подробно объяснить содержимое трёх строк файла листинга по выбору:) mov ecx, [A] - загружает значение переменной A в ecx. mov [max], ecx - копирует значение из регистра ecx в переменную max. cmp ecx, [C] - сравнивает значение в регистре ecx с числом, хранящимся в переменной C Если мы удалим операнд, это вызовет ошибку и выходной файл не создастся, а в листинге появится текст ошибки (рис. 11):

Рис. 11: снимок экрана

Рис. 11: снимок экрана

## 3.3 Выполнение заданий для самостоятельной работы.

Создаем исполняемый файл lab7-3.asm, в котором пишем программу нахождения наименьшей из 3 целочисленных переменных a,b и с (значения из 15 варианта) (рис. 12), и проверяем его работу, введя значение В (рис. 13):

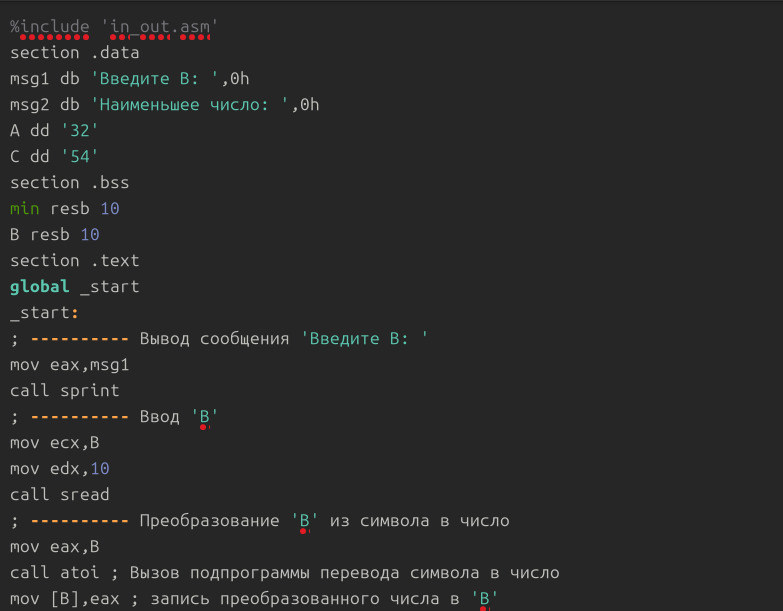


Рис. 12: снимок экрана

%include 'in\_out.asm'  
section .data  
msg1 db 'Введите В: ',0h  
msg2 db 'Наименьшее число: ',0h  
A dd '32'  
C dd '54'  
section .bss  
min resb 10  
B resb 10  
section .text  
global \_start  
\_start:  
; ---------- Вывод сообщения 'Введите В: '  
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 'min'  
mov ecx,[A] ; 'ecx = A'  
mov [min],ecx ; 'min = A'  
; ---------- Сравниваем 'A' и 'С' (как символы)  
cmp ecx,[C] ; Сравниваем 'A' и 'С'  
jg check\_B ; если 'A>C', то переход на метку 'check\_B',  
mov ecx,[C] ; иначе 'ecx = C'  
mov [min],ecx ; 'min = C'  
; ---------- Преобразование 'min(A,C)' из символа в число  
check\_B:  
mov eax,min  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [min],eax ; запись преобразованного числа в `min`  
; ---------- Сравниваем 'min(A,C)' и 'B' (как числа)  
mov ecx,[min]  
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'  
jb fin ; если 'min(A,C)<B', то переход на 'fin',  
mov ecx,[B] ; иначе 'ecx = B'  
mov [min],ecx  
; ---------- Вывод результата  
fin:  
mov eax,msg2  
call sprint ; Вывод сообщения 'Наименьшее число: '  
mov eax,[min]  
call iprintLF ; Вывод 'min(A,B,C)'  
call quit ; Выход

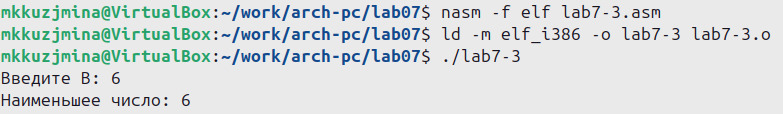


Рис. 13: снимок экрана

Создаем файл lab7-4.asm, пишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбраем из таблицы вариантов заданий (15ый). (рис. 14):

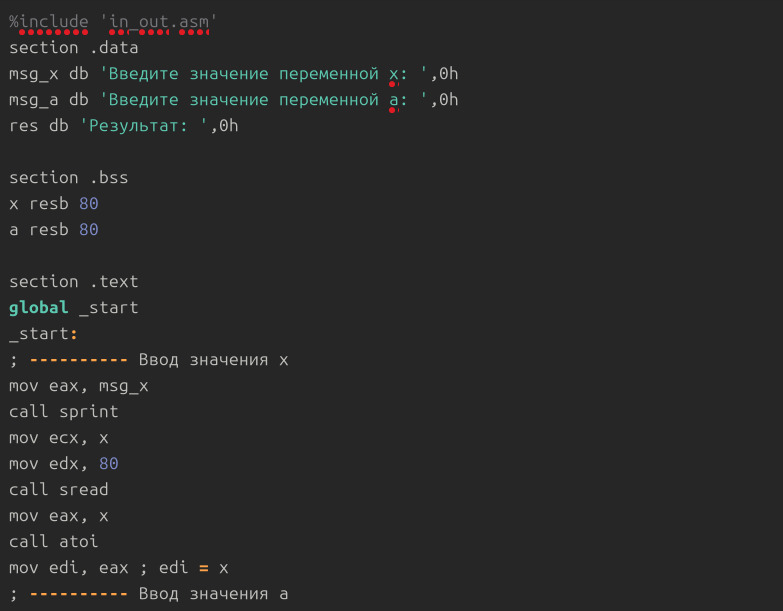


Рис. 14: снимок экрана

%include 'in\_out.asm'  
section .data  
msg\_x db 'Введите значение переменной x: ',0h  
msg\_a db 'Введите значение переменной a: ',0h  
res db 'Результат: ',0h  
  
section .bss  
x resb 80  
a resb 80  
  
section .text  
global \_start  
\_start:  
; ---------- Ввод значения x  
mov eax, msg\_x  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
mov edi, eax ; edi = x  
; ---------- Ввод значения a  
mov eax, msg\_a  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax, a  
call atoi  
mov esi, eax ; esi = a  
; ---------- Сравнение x и a  
cmp edi, esi  
jl less\_than\_a ; если x < a, переход на less\_than\_a  
; ---------- x >= a, вычисляем x + 10  
mov eax, edi  
add eax, 10  
jmp print\_result  
less\_than\_a:  
; ---------- x < a, вычисляем a + 10  
mov eax, esi  
add eax, 10  
; ---------- Вывод результата  
print\_result:  
mov edi, eax  
mov eax, res  
call sprint ; Вывод сообщения 'Результат: '  
mov eax, edi  
call iprintLF ; Вывод результата  
call quit ; Выход

Создаем исполняемый файл и проверяем его работу для значений x и a (рис. 15):

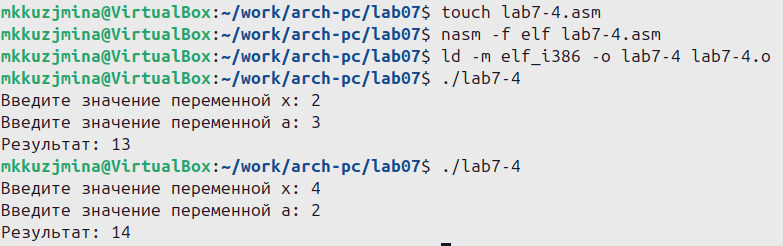


Рис. 15: снимок экрана

# 4 Выводы

Были изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов, ознакомились с назначением и структурой файла листинга.