Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Богданюк Анна Васильевна

Содержание

# Цель работы

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

# Задание

1. Выполнение лабораторной работы
2. Задания для самостоятельной работы

# 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 программы без каких-либо условий. Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление. Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре. 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. В следующей таблице указано положение битовых флагов в регистре флагов. 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. Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию. Структура листинга: • номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы); • адрес — это смещение машинного кода от начала текущего сегмента; • машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра); • исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

# Выполнение лабораторной работы

1.Выполнение лабораторной работы

Создаю каталог для программам лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. [1](#fig:001)).

Figure 1: Создание файла

Figure 1: Создание файла

В файл lab7-1.asm ввожу текст листинга(рис. [2](#fig:002)).

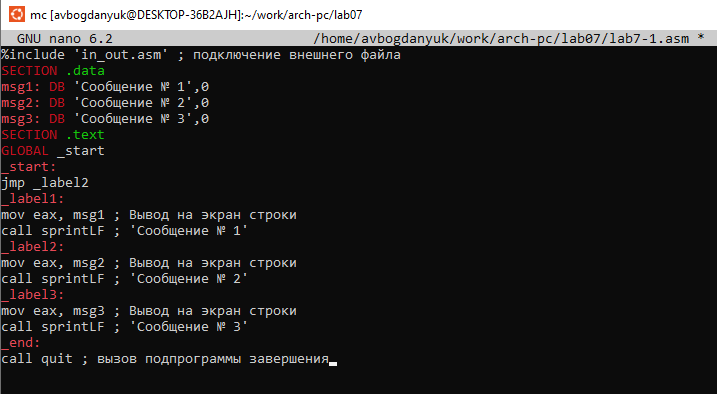


Figure 2: Создание файла

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

Создаю исполнительный файл и запускаю его (рис. [3](#fig:003)).

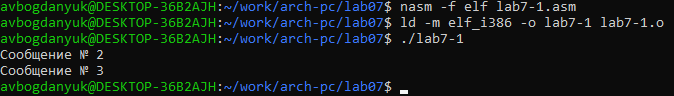


Figure 3: Создание и вывод

Изменяю текст файла lab7-1.asm (рис. [4](#fig:004)).

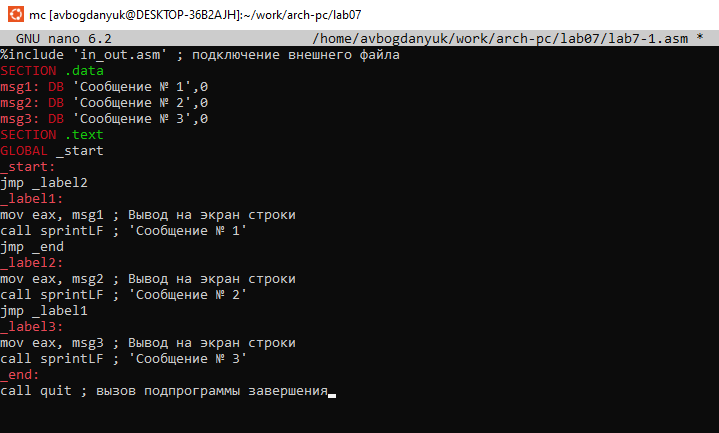


Figure 4: Файл lab7-1.asm

%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 ; 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'  
\_end:  
call quit ; вызов подпрограммы завершения

Изменяю текст файла так, чтобы выводилось 3 сообщения (рис. [5](#fig:005)).

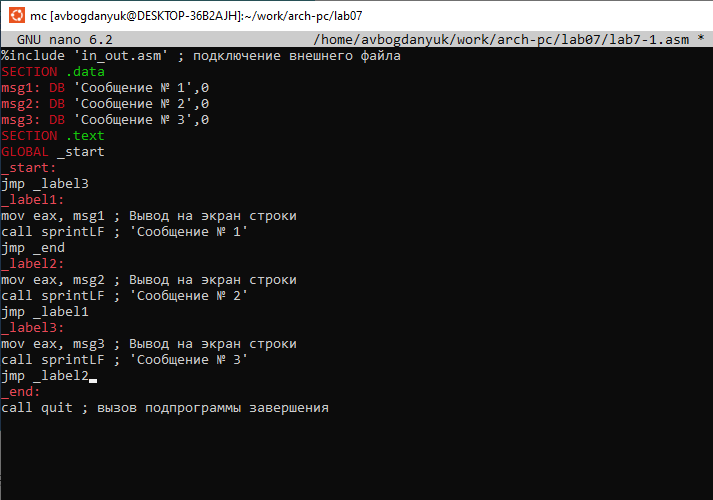


Figure 5: Файл lab7-1.asm

%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 ; 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

Создаю исполнительный файл и запускаю его (рис. [6](#fig:006)).

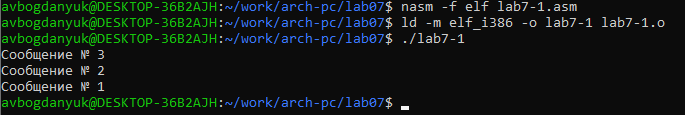


Figure 6: Создание и вывод

Создаю файл lab7-2.asm и ввожу текст листинга (рис. [7](#fig:007)).

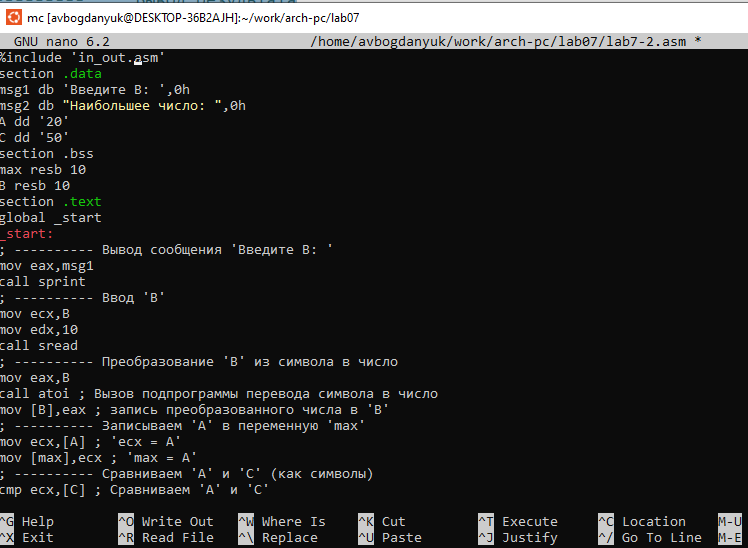


Figure 7: Файл lab7-2.asm

%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 '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,max  
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

Создаю исполнительный файл и запускаю его. Все работает корректно (рис. [8](#fig:008)).

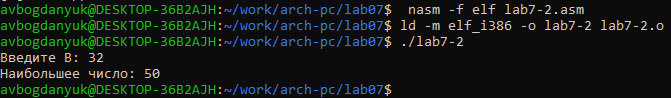


Figure 8: Создание и вывод

Создаю файл листинга для программы из файла lab7-2.asm (рис. [9](#fig:009)).

Figure 9: Создание

Figure 9: Создание

Открываю листинг(рис. [10](#fig:010)).

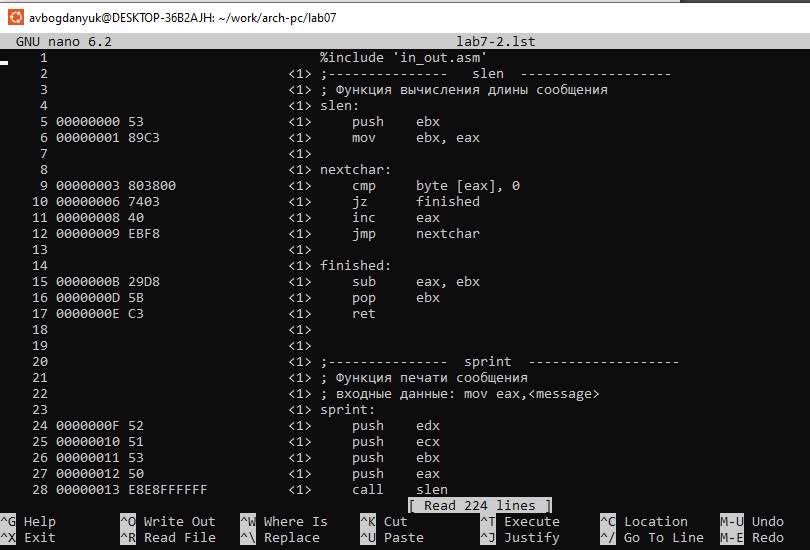


Figure 10: Листинг

‘21’ - это номер строки, ‘Функция печати сообщения’ - комментарий к коду, нет адреса и машинного кода ‘22’ - это номер строки, ‘входные данные: mov eax, ’ - это комментарий коду, также не имеет адреса и машинного кода ‘23’ - это номер строки, ‘sptint’ - название функции, не имеет адрема и машинного кода (рис. [11](#fig:011)).

Figure 11: Строки, которые были выбраны для объяснения

Figure 11: Строки, которые были выбраны для объяснения

Выбираю инструкцию с двумя операндами, удаляю ту, которая выделена (рис. [12](#fig:012)).

Figure 12: Файл lab7-2.lst

Figure 12: Файл lab7-2.lst

Пытаюсь создать файл листинг, но выдается ошибка, так как иструкция cmp не может работать, имея только один операнд (рис. [13](#fig:013)).

Figure 13: Файл lab7-1.asm

Figure 13: Файл lab7-1.asm

1. Задания для самостоятельной работы

Мой вариант из лабораторной работы №6 - 4. Мои значения а=8, b=88, c=68 (рис. [14](#fig:014)).



Figure 14: Файл lab7-3.asm

Создаю исполнительный файл и запускаю его. Результат корректный (рис. [15](#fig:015)).

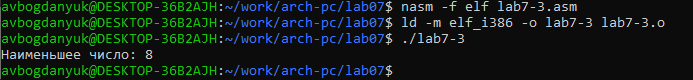


Figure 15: Создание и вывод

%include 'in\_out.asm'  
section .data  
msg db "Наименьшее число: ", 0h  
A dd '8'  
B dd '88'  
C dd '68'  
section .bss  
min resb 10  
section .text  
global \_start  
\_start:  
mov ecx, [A]  
mov [min],ecx  
  
cmp ecx, [C]  
jl check\_B  
mov ecx, [C]  
mov [min], ecx  
  
check\_B:  
mov eax, min  
call atoi  
mov [min], eax  
mov ecx, [min]  
cmp ecx, [B]  
jl fin  
mov ecx, [B]  
mov [min], ecx  
  
fin:  
mov eax, msg  
call sprint  
mov eax, [min]  
call iprintLF  
call quit

Вариант 4. 2x+1, если a=0, в противном случае 2x+a (рис. [16](#fig:016)).

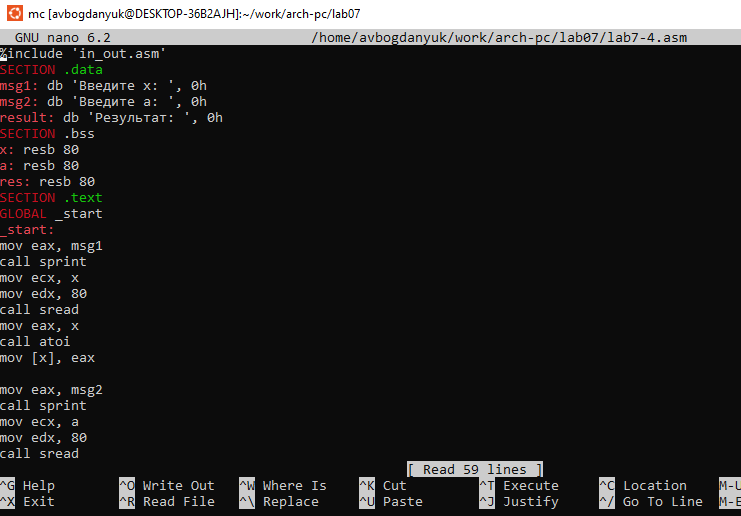


Figure 16: Файл lab7-4.asm

Создание исполнительного файла и запуск его. Результат корректный (рис. [17](#fig:017)).

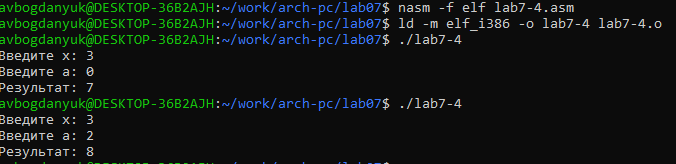


Figure 17: Создание и вывод

%include 'in\_out.asm'  
SECTION .data  
msg1: db 'Введите x: ', 0h  
msg2: db 'Введите a: ', 0h  
result: db 'Результат: ', 0h  
SECTION .bss  
x: resb 80  
a: resb 80  
res: resb 80  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg1  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
mov [x], eax  
  
mov eax, msg2  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax, a  
call atoi  
mov [a], eax  
  
mov ecx, [a]  
  
cmp ecx, 0  
je \_f1  
jne \_f2  
  
\_f1:  
mov eax, [x]  
mov ebx, 2  
imul eax, ebx  
mov ebx, 1  
add eax, ebx  
mov [res], eax  
jmp \_fin  
  
\_f2:  
mov eax, [x]  
mov ebx, 2  
imul eax, ebx  
add eax, ecx  
mov [res], eax  
jmp \_fin  
  
\_fin:  
mov eax, result  
call sprint  
mov eax, [res]  
call iprintLF  
call quit

# Выводы

В ходе проведения лабораторной работы были изучены команды условного и безусловного переходов, приобретены навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.

# Список литературы