Отчет по лабораторной работа №8

Группа НПИбд-02-22

Стариков Данила Андреевич

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 Основная часть

## 2.1 Выполнение лабораторной работы

### 2.1.1 Реализация переходов в NASM

Для выполнения лабораторной работы создали каталог ~/work/study/arch-pc/lab08. В нем создали файл lab7-8.asm и ввели текст из Листинга 1. Также положили в каталог файл in\_out.asm, использованный в лабораторной работе №6.

Листинг 1: Программа с использованием инструкции jmp

%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

Создали исполняемый файл и запустили его (Рис. 1). Инструкция jmp \_label2 изменяет порядок выполнения программы, пропуская вывод первого сообщения. Далее изменили текст программы (Листинг 2), выводится второе, а затем первое сообщения, третье игнорируется. ((Рис. 2)).

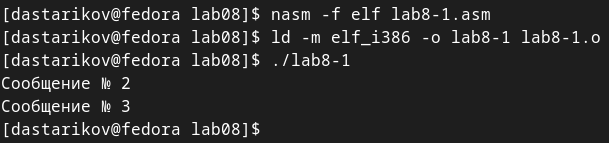


Рис. 1: Результат запуска файла lab8-1.

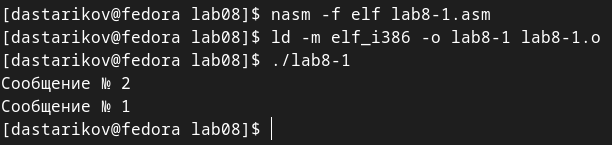


Рис. 2: Порядок вывода сообщений изменился.

Листинг 2: Программа с использованием инструкции jmp

%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

Затем изменили программу, что сообщения выводились в обратном порядке в соответствии с Листингом 3 (Рис. 3)

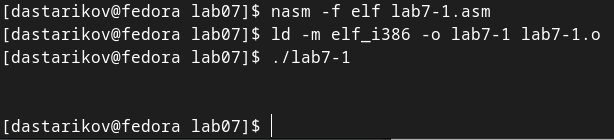


Рис. 3: Сообщения выводятся в обратном порядке.

Листинг 3: Программа с использованием инструкции jmp

%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

Рассмотрели команды условного перехода на примере программы lab8-2.asm, вычисляющей наибольшее из 3х целых чисел A, B и C (Листинг 4). A и C заданы заранее, а B вводится с клавиатуры, при этом A и С для демонстрации сравнивается как символы, а наибольшее из них конвертируется в число и с сравнивается с B как число (Рис. 4).

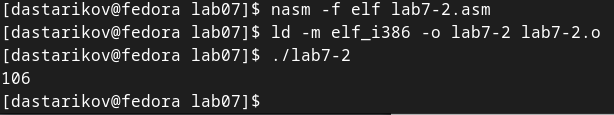


Рис. 4: Результат работы программы lab8-2 при вводе разных чисел.

Листинг 4: Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C.

%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

### 2.1.2 Изучение структуры файла листинга

Обычно nasm создает только объектный файл, для создания файла листинга необходимо указать ключ -l в командной строке.

nasm -f elf lab8-2.asm -l lab8-2.lst

Проанализировали полученный файл на примере 3 строк:

20 ; ---------- Ввод 'B'  
 21 000000F2 B9[0A000000] mov ecx,B  
 22 000000F7 BA0A000000 mov edx,10  
 23 000000FC E842FFFFFF call sread

Первый столбец содержит номер строки тексте программы: 21, 22, 23. Затем идет адрес команды в текущем сегменте кода: так инструкция mov ecx, B начинается по виртуальному адресу 000000F2. Виртуальный адрес - это число из абстрактного виртуального адресного пространства, характерной конкретной программе, и не всегда соответствующее адресу физической памяти. Далее идет машинное представление инструкции: mov ecx ассемблируется как B9, а так как переменная B обозначает виртуальный адрес, где хранится ее значение, то инструкция [0A000000] означает, что нужно взять данные по адресу 0000000A (адрес в сегменте .bss). Далее идет исходный текст программы. Далее 22 строка хранится по адресу 000000F7 - сдвиг на 5 байт от предыдущей, так как 1 байт занимает инструкция mov ecx и 4 байта - адрес памяти переменной B. BA - move edx и 0A000000 - число 10 в шестнадцатеричном представлении (Разрядность увеличивается справа налево). Затем строка 23: перешли еще на 5 байтов, E842FFFFFF соответствует вызову инструкции call sread.

Попробовали изменить программу, чтобы вызвать ошибку: команда mov всегда принимает 2 операнда (Рис. 5). При попытке трансляции с флагом создания файла листинга происходится ошибка, при это файл листинга создается с указанием, где произошла ошибка (Рис. 6).

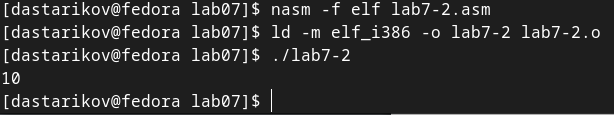


Рис. 5: Измененная часть программы.

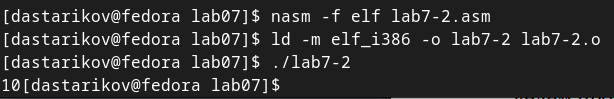


Рис. 6: Файл листинга указывает, где произошла ошибка.

## 2.2 Выполнение заданий для самостоятельной работы.

Для выполнения заданий выбран вариант 12, полученный при выполнении лабораторной работы №7.

### 2.2.1 Задание 1.

Написали программу (Листинг 5), сравнивающую 3 целых числа a, b и c. Чиссла вводятся с клавиатуры, в результаты выводится наибольшее из них. Проверили работу программы на числах 99, 29, 26 (Рис. 7).

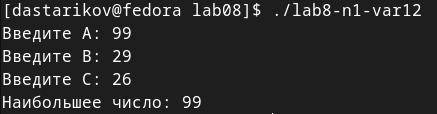


Рис. 7: Результат работы программы из задания 1 (Вариант 12).

Листинг 5: Программа находит наибольшее из 3 чисел.

%include 'in\_out.asm'  
  
section .data  
 msgA db 'Введите A: ',0h  
 msgB db 'Введите B: ',0h  
 msgC db 'Введите C: ',0h  
  
 msg2 db "Наибольшее число: ",0h  
  
section .bss  
 max resb 10   
 A resb 10  
 B resb 10  
 C resb 10  
  
section .text  
 global \_start  
  
\_start:  
; ---------- Получение переменной A  
 mov eax, msgA  
 call sprint  
  
 mov ecx,A  
 mov edx,10  
 call sread  
  
 mov eax,A  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [A],eax ; запись преобразованного числа в 'A'  
  
; ---------- Получение переменной B  
 mov eax, msgB  
 call sprint  
  
 mov ecx,B  
 mov edx,10  
 call sread  
  
 mov eax,B  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [B],eax ; запись преобразованного числа в 'B'  
  
; ---------- Получение переменной C  
 mov eax, msgC  
 call sprint  
  
 mov ecx,C  
 mov edx,10  
 call sread  
  
 mov eax,C  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [C],eax ; запись преобразованного числа в 'C'  
  
; ---------- Сравнение переменных  
  
 mov ecx,[A] ; 'ecx = A'  
   
 cmp ecx,[B] ; Сравниваем 'A' и 'B'  
 jg check\_2 ; если 'A>B', то переход на метку 'check\_2',  
 mov ecx,[B] ; иначе 'ecx = B'  
 mov [max],ecx ; 'max = B'  
  
check\_2:  
 mov [max], ecx   
  
; ---------- Сравниваем 'max(A,B)' и 'C'  
 mov ecx,[max]  
 cmp ecx,[C] ; Сравниваем 'max(A,B)' и 'C'  
 jg fin ; если 'max(A,B)>C', то переход на 'fin',  
 mov ecx,[C] ; иначе 'ecx = C'  
 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

### 2.2.2 Задание 2.

Написали программу (Листинг 6), вычисляющую значение выражения:

Проверили рабооту программы на двух примерах: (Рис. 8).

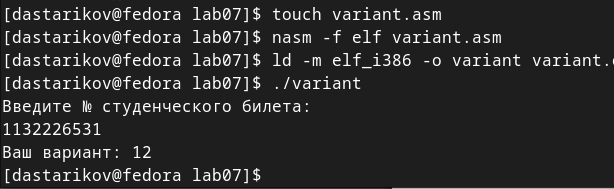


Рис. 8: Результат работы программы из задания 2 (Вариант 12)

Листинг 6: Программа вычисления выражения f(x)

%include 'in\_out.asm'  
  
section .data  
 msgf db 'f(x) = a\*x, x<5',10, 9, 'x-5, ','x>=5',0h  
 msgx db 'Введите x: ',0h  
 msga db 'Введите a: ',0h  
 msg2 db 'f(x) = ',0h  
  
section .bss  
 res resb 10  
 x resb 10  
 a resb 10  
  
section .text  
 global \_start  
  
\_start:  
; ---------- Вывод функции  
 mov eax, msgf  
 call sprintLF  
  
; ---------- Получение переменной x  
 mov eax, msgx  
 call sprint  
  
 mov ecx,x  
 mov edx,10  
 call sread  
  
 mov eax,x  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [x],eax ; запись преобразованного числа в 'x'  
  
; ---------- Получение переменной a  
 mov eax, msga  
 call sprint  
  
 mov ecx,a  
 mov edx,10  
 call sread  
  
 mov eax,a  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [a],eax ; запись преобразованного числа в 'a'  
  
; ---------- Проверка значения x (x<5 или part\_2: x>=5)  
 mov ecx,[x] ; 'ecx = x'  
   
 cmp ecx,5 ; Сравниваем 'x' и '5'  
 jge part\_2 ; если 'x>=5', то переход на метку 'part\_2',  
 mov eax,[x] ; иначе вычисляем выражение a\*x  
 mov ebx,[a]  
 mul ebx ; eax=eax\*ebx  
 mov [res], eax  
 jmp end  
part\_2: ; вычисляем выражение x-5  
 mov eax, [x]  
 sub eax, 5  
 mov [res], eax  
; ---------- Вывод результата вычислений  
end:  
 mov eax, msg2  
 call sprint ; Вывод сообщения 'f(x) = '  
 mov eax,[res]  
 call iprintLF ; Вывод результата  
 call quit ; Выход

Созданные файлы \*.asm скопировали в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab08/ и загрузили на Github.

# 3 Выводы

В рамках лабораторной работы получили практические навыки использования условных и безусловных переходов, анализа файла листинга.