Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений

Колонтырский Илья Русланович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Создадим папку lab07 и файл lab7-1.asm (рис. 2.1)

```
irkolontyrskiy@irkolontyrskiy:~$ mkdir ~/work/arch-pc/lab07
irkolontyrskiy@irkolontyrskiy:~$ cd ~/work/arch-pc/lab07
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 2.1: Создание папки и файла lab7-1.asm

Теперь вставим предложенный код в файл lab7-1.asm (рис. 2.2)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/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'
_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
```

Рис. 2.2: Вставка кода

Теперь соберём программу и проверим её рпботу (рис. 2.3)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o l
ab7-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 2.3: Сборка программы из файла lab7-1.asm и её запуск

Изменим файл lab7-1.asm, вставив предложенный код (рис. 2.4)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/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
```

Рис. 2.4: Изменение файла lab7-1.asm согласно листингу 7.2

Ещё раз скомпилируем программу, проверим её (рис. 2.5)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o l
ab7-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 2.5: Повторная компиляция программы, запуск для проверки

Нужно сделать так, чтобы программа выводила сообщения от третьего к первому. Изменим код следующим образом (рис. 2.6)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/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
```

Рис. 2.6: Изменение файла

Соберём и запустим программу (рис. 2.7)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o l
ab7-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 2.7: Сборка и запуск

Теперь создадим файл второй файл - lab7-2.asm (рис. 2.8)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ touch lab7-2.asm
```

Рис. 2.8: Создание файла lab7-2.asm

Вставим в файл lab7-2.asm следующий код (рис. 2.9)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/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 'C' (как символы)
```

Рис. 2.9: Запись кода в файл lab7-2.asm

Теперь скомпилируем его и запустим, провери для разных значений (рис. 2.10)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o l
ab7-2
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
```

Рис. 2.10: Компиляция программы и проверка работы

Попробуем пересобрать файл lab7-2.asm так, чтобы создался файл листинга и откроем файл листинга (рис. 2.11)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst la
b7-2.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 2.11: Создание листинга и его открытие

После его открытия мы видим следующее (рис. 2.12)

```
/home/ir~7-2.lst   [----]  0 L:[184+12 196/225] *(11987/14458b) 0032 0x020[*][X]
     9 0000000A <res Ah>                B resb 10
    10                                  section .text
    11                                  global _start
    12                                  _start:
    13                                  ; ---------- Вывод сообщения 'Введите B:
    14 000000E8 B8[00000000]            mov eax,msg1
    15 000000ED E81DFFFFFF              call sprint
    16                                  ; ---------- Ввод 'B'
    17 000000F2 B9[0A000000]            mov ecx,B
    18 000000F7 BA0A000000              mov edx,10
    19 000000FC E842FFFFFF              call sread
    20                                  ; ---------- Преобразование 'B' из симво
    21 00000101 B8[0A000000]            mov eax,B
    22 00000106 E891FFFFFF              call atoi ; Вызов подпрограммы перевода
    23 0000010B A3[0A000000]            mov [B],eax ; запись преобразованного чи
    24                                  ; ---------- Записываем 'A' в переменную
    25 00000110 8B0D[35000000]          mov ecx,[A] ; 'ecx = A'
    26 00000116 890D[00000000]          mov [max],ecx ; 'max = A'
    27                                  ; ---------- Сравниваем 'A' и 'C' (как с
    28 0000011C 3B0D[39000000]          cmp ecx,[C] ; Сравниваем 'A' и 'C'
    29 00000122 7F0C                    jg check_B ; если 'A>C', то переход на м
    30 00000124 8B0D[39000000]          mov ecx,[C] ; иначе 'ecx = C'
    31 0000012A 890D[00000000]          mov [max],ecx ; 'max = C'
    32                                  ; ---------- Преобразование 'max(A,C)' и
    33                                  check_B:
    34 00000130 B8[00000000]            mov eax,max
    35 00000135 E862FFFFFF              call atoi ; Вызов подпрограммы перевода
    36 0000013A A3[00000000]            mov [max],eax ; запись преобразованного
    37                                  ; ---------- Сравниваем 'max(A,C)' и 'B'
```

Рис. 2.12: Содержимое файла листинга

Теперь рассмотрим несколько строк файла листинга:

1. Строка 21 перемещает значение переменной B регистр eax
2. Строка 22 преобразовывает значение регистра eax в число
3. Строка 23 перемещает значение регистра eax в переменную B

Намеренно сделаем ошибку в коде, и уберём у команды mov один операнд (рис. 2.13)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/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
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 'C' (как символы)
```

Рис. 2.13: Создание ошибки в файле

Теперь соберём этот файл и создадим файл листинга (рис. 2.14)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst la
b7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
```

Рис. 2.14: Создание файла листинга и вывод ошибки

Откроем файл листинга и посмотрим на изменения (рис. 2.15)

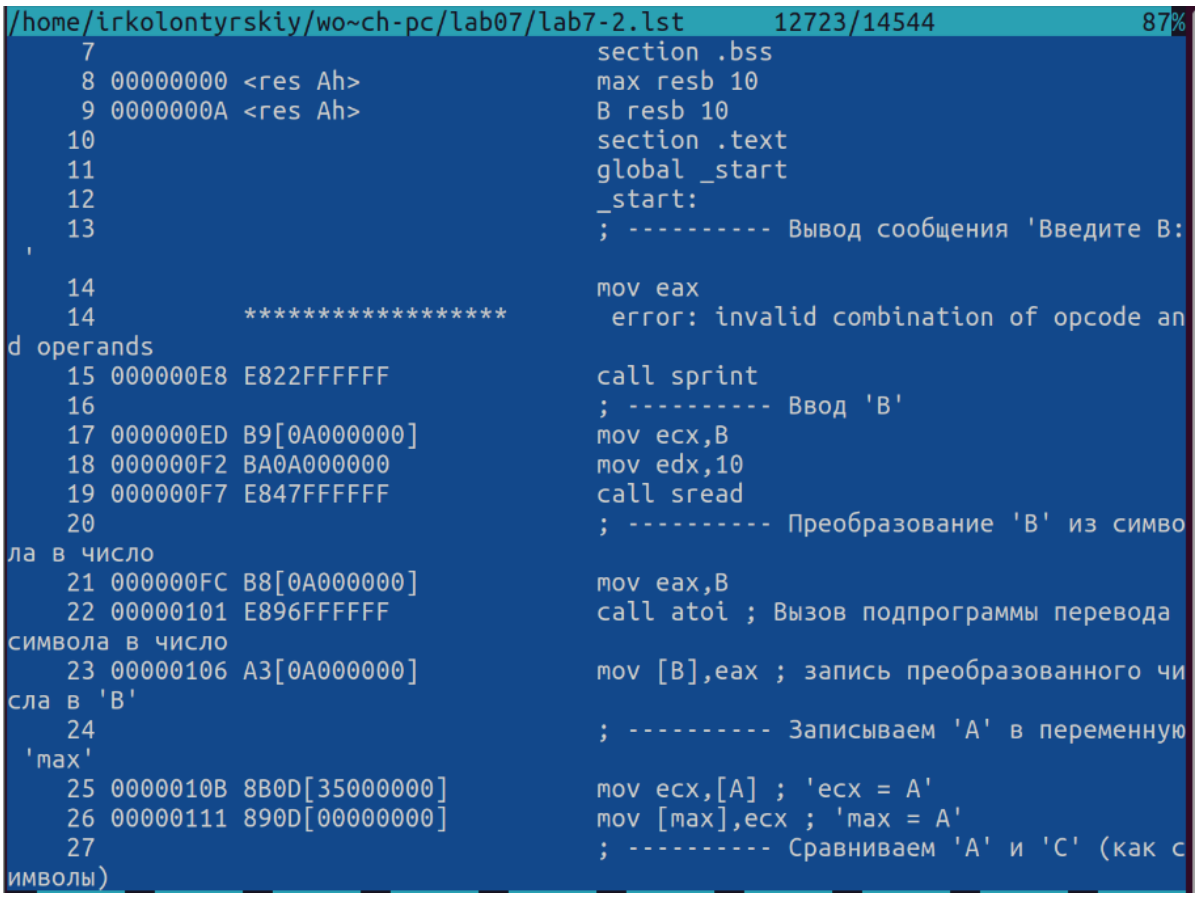

Рис. 2.15: Файл листинга с ошибкой

Заметим, что в листинге появилась ошибка

Самостоятельная работа

Создадим файлы для самостоятельной работы (19 вариант) (рис. 2.16)

Рис. 2.16: Создание файлов самостоятельной работы

Код для выполнения первого задания выглядит так (рис. 2.17)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/var19-1.asm
%include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '46'
B dd '32'
C dd '74'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,B
call atoi
mov [B],eax
mov eax,A
call atoi
mov [A],eax
mov eax,C
call atoi
mov [C],eax
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx
check_B:
mov ecx,[min]
cmp ecx,[B]
jl final
mov ecx,[B]
mov [min],ecx
final:
mov eax, msg
call sprint
mov eax,[min]
call iprintLF
call quit
```

Рис. 2.17: Код первого задания

Скомпилируем его и посмотрим на результат (рис. 2.18)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ nasm -f elf var19-1.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ld -m elf_i386 var19-1.o -o
var19-1
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab07$ ./var19-1
Наименьшее число: 32
```

Рис. 2.18: Компиляция файла и результат выполнения

Код для выполнения второго задания выглядит так (рис. 2.19)

```
 GNU nano 6.2   /home/irkolontyrskiy/work/arch-pc/lab07/var19-2.asm
%include 'in_out.asm'
section .data
msg1 DB "Введите X: ",0h
msg2 DB "Введите A: ",0h
msg3 DB "Ответ=",0h
section .bss
x: RESB 80
a: RESB 80
ans: 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 eax, [x]
cmp eax, [a]
jle xsa
mov eax, [a]
add eax, [x]
jmp ansv
xsa:
mov eax, [x]
ansv:
mov [ans],eax
mov eax,msg3
call sprint
mov eax,[ans]
call iprintLF
call quit
```

Рис. 2.19: Код второго файла самостоятельной работы

Скомпилируем его и посмотрим на результат (рис. 2.20)

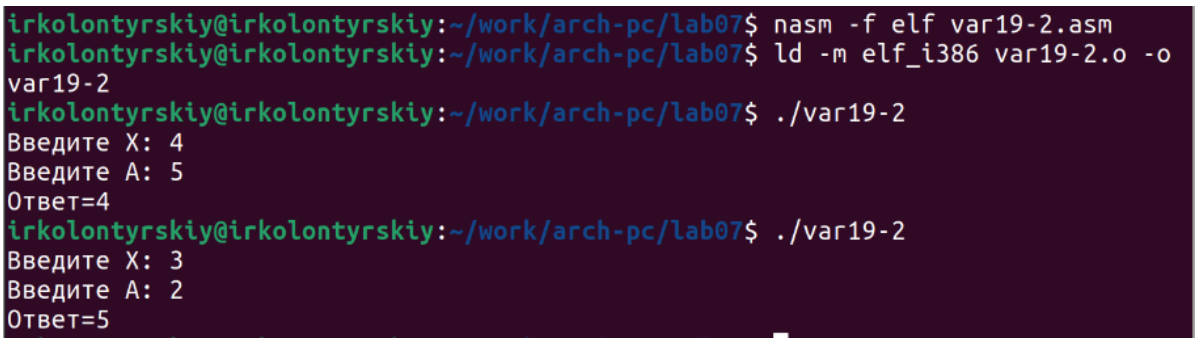

Рис. 2.20: Компиляция файла и результат выполнения

Программы считают всё верно

3 Выводы

Были изучены команды условных и безусловных переходов, а также они были применены на практике. Была рассмотрена работа с файлами листинга