# Лабораторная работа №7

**Команды безусловного и условного переходов в Nasm. Программирование ветвлений.**

Приспешкин Андрей Андреевич

# 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 Задание

1. Команды безусловного перехода
2. Команды условного перехода
3. Файлы листинга
4. Задания для самостоятельной работы

# 3 Выполнение лабораторной работы

Создадим новый каталог в котором будем делать лабораторную работу, в рабочем каталоге arch-pc, там-же создадим файл lab7-1.asm(Рис.1).

```
aaprispeshkin:[aaprispeshkin]:~$ mkdir ~/work/arch-pc/lab07
aaprispeshkin:[aaprispeshkin]:~$ cd ~/work/arch-pc/lab07
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1: Создание рабочего каталога и файла lab7-1.asm

В файл lab7-1.asm вставим код из листинга 7.1(Рис.2).

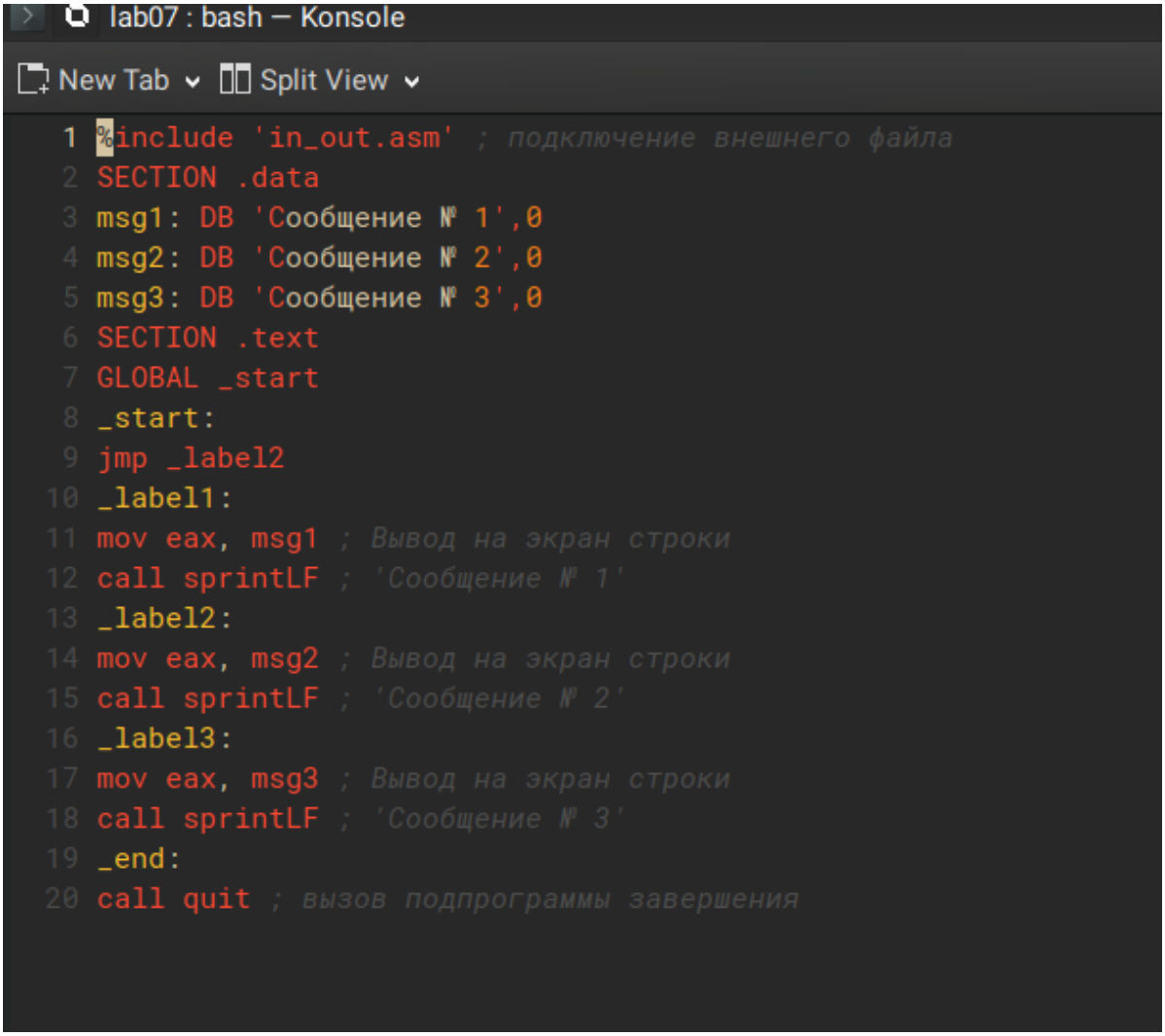


Рис. 3.2: Код из листинга 7.1

Создадим исполняемый файл и проверим результат работы кода(Рис.3).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Создание исполняемого файла и результат его работы

Изменим код в соответствии с листингом 7.2 и проверим результат(Рис.4).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.4: Результат после внесения изменений в код

Поменяем код так, чтобы порядок выводимых на экран сообщений был “3, 2, 1”(Рис.5).

```
%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. 3.5: Код после внесения изменений

Создадим исполняемый файл и проверим результат(Рис.6).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.6: Проверка результата работы исполняемого файла

Создадим файл lab7-2.asm(Рис.7).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ touch lab7-2.asm
```

Рис. 3.7: Создание файла lab7-2.asm

Вставим туда код в соответствии с листингом 7.3(Рис.8).

```
%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)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
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:
NORMAL  lab7-2.asm                                                    lab07  Top
```

Рис. 3.8: Код из листинга 7.3

Проверим правильность работы программы с разными входными данными(Рис.9).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 52
Наибольшее число: 52
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 23
Наибольшее число: 50
```

Рис. 3.9: Проверка работы программы

Создадим файла листинга для файла lab7-2.asm(Рис.10).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.10: Создание файла листинга

Откроем его текстовым редактором neovim(Рис.11).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nvim lab7-2.lst
```

Рис. 3.11: Открытие файла листинга в текстовом редакторе

Заметим, что файлы листинга устроены таким образом, что сначала там пишется номер строки, а затем адреса в памяти(Рис.12).


Рис. 3.12: Вид файла листинга

Попробуем удалить из файла lab7-2.asm операнд из строки 45(Рис.13).

```
44 fin:
45 mov eax
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
```

Рис. 3.13: Строка 45 после удаления операнда msg2

Заметим, что в результате при попытке создасть объектный файл мы получаем ошибку(Рис.14).

```
aaprispeshkin:[aaprispeshkin]:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:45: error: invalid combination of opcode and operands
```

Рис. 3.14: Ошибка при создании объектного файла

Напишем код, который будет вычислять выражение $ 2a - x $ в случае если $ x < a $ или выводить 8 в противном случае(Рис.15 и 16).

```
%include 'in_out.asm'

section .data
 msg1 db 'Введите x: ', 0h
 msg2 db 'Введита a: ',0h
 msg3 db 'Ответ: ', 0h

section .bss
 X resb 10
 A resb 10

section .text
 global _start

_start:
 mov eax, msg1
 call sprint
 mov ecx, X
 mov edx, 10
 call sread
 mov eax, X
 call atoi
 mov [X], eax

 mov eax, msg2
 call sprint
 mov ecx, A
 mov edx, 10
 call sread
 mov eax, A
 call atoi
 mov [A], eax

 cmp [X], eax
 jl Lesser
 mov eax, msg3
 call sprint
 mov eax, 8
 call iprintLF
 call quit

Lesser:
 mov eax, [A]
 mov ebx, 2
NORMAL  lab7-4.asm                                    lab07  67%
```

```
22    call atoi
23    mov [X], eax
24
25    mov eax, msg2
26    call sprint
27    mov ecx, A
28    mov edx, 10
29    call sread
30    mov eax, A
31    call atoi
32    mov [A], eax
33
34    cmp [X], eax
35    jl Lesser
36    mov eax, msg3
37    call sprint
38    mov eax, 8
39    call iprintLF
40    call quit
41
42 Lesser:
43    mov eax, [A]
44    mov ebx, 2
45    mul ebx
46    mov ebx, [X]
47    sub eax, ebx
48    mov ecx, eax
49    mov eax, msg3
50    call sprint
51    mov eax, ecx
52    call iprintLF
53    call quit
54
```

Создадим исполняемый файл и проверим результат его работы при x(1,2) и a(2,1) (Рис.17)


# Выводы

Я изучил команды безусловного и условного переходов и приобрёл навыки написания программ с их использованием.

# Список литературы

Лабораторная работа №7