

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ
НАРОДОВ**

**Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7.**

Дисциплина: Архитектура компьютера

Студент: Слабоспицкий Платон Сергеевич

Группа: НКАбд-05-25

МОСКВА 2025 г.

Оглавление

1. Цель работы:	3
2. Порядок выполнения лабораторной работы	4
2.1. Реализация переходов в NASM	4
2.2. Изучение структуры файлы листинга	10
2.3. Задание для самостоятельной работы.....	12
3. Вывод:.....	17

1. Цель работы:

Изучение команд условного и безусловного переходов.

Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2. Порядок выполнения лабораторной работы

2.1. Реализация переходов в NASM

Создадим каталог для программ лабораторной работы № 7, перейдём в него и создадим файл *lab7-1.asm*:

```
psslabospickiyj@dk7n03 ~ $ mkdir ~/work/arch-pc/lab07
psslabospickiyj@dk7n03 ~ $ cd ~/work/arch-pc/lab07
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рисунок 1 Создадим файл *lab7-1.asm*

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введём в файл *lab7-1.asm* текст программы из листинга 7.1.

```
%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
```

Рисунок 2 Листинг 7.1.

Создадим исполняемый файл и запустим его:

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm  
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o  
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ./lab7-1
```

Сообщение № 2

Сообщение № 3

Рисунок 3 Запустили файл lab7-1

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим 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).

Создадим файл lab7-01.asm и введём в него текст программы из листинга 7.2. :

```
GNU nano 8.6      /afs/.dk.sci.ptu.edu.ru/home/p/s/psslabospickiyj/wor  
%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
```

[Прочитано 22 строки]
^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выпол
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиц

Рисунок 4 Листинг 7.2.

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ touch lab7-01.asm
```

Рисунок 5 Создаём файл lab7-01.asm

Теперь запустим файл.

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ nasm -f elf lab7-01.asm
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-01 lab7-01.o
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ./lab7-01
Сообщение № 2
Сообщение № 1
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ █
```

Рисунок 6 Запустили файл lab7-01

Изменим текст программы изменив инструкции jmp, чтобы вывод программы был следующим:

```
user@dk4n31:~$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@dk4n31:~$
```

Для этого создадим копию файла с именем lab7-02.asm, исправим, оттранслируем и запустим.

```
psslabospickiyj@dk7n07 ~/work/arch-pc/lab07 $ touch lab7-02.asm
psslabospickiyj@dk7n07 ~/work/arch-pc/lab07 $ nasm -f elf lab7-02.asm
psslabospickiyj@dk7n07 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-02 lab7-02.o
psslabospickiyj@dk7n07 ~/work/arch-pc/lab07 $ ./lab7-02
Сообщение № 3
Сообщение № 2
Сообщение № 1
psslabospickiyj@dk7n07 ~/work/arch-pc/lab07 $
```

Рисунок 7 Запустили файл lab7-02.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      ; Первый переход к выводу сообщения №3

_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      ; Переход к выводу сообщения №1

_label3:
    mov eax, msg3    ; Вывод на экран строки
    call sprintLF   ; 'Сообщение № 3'
    jmp _label2      ; Переход к выводу сообщения №2

_end:
    call quit        ; вызов подпрограммы завершения

```

Рисунок 8 текст программы lab7-02.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создадим файл *lab7-2.asm* в каталоге *~/work/arch-pc/lab07*.

Внимательно изучим текст программы из листинга 7.3 и введём в *lab7-2.asm*.

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ touch lab7-2.asm
```

Рисунок 9 Создание файла *lab7-2.asm*

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 1
Наибольшее число: 50
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 2
Наибольшее число: 50
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 56
Наибольшее число: 56
```

Рисунок 10 Запустили файл *lab7-2* и проверили корректность выполнения программы, введя несколько различных значений

```
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- 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 ; ОШИБКА: удален второй операнд [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

```

Рисунок 11 Текст листинга 7.3.

2.2. Изучение структуры файлы листинга

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создадим файл листинга для программы из файла *lab7-2.asm*

```
psslabospickiyj@dk7n03 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рисунок 12 Сделали файл листинга для программы из файла *lab7-2.asm*

Откроем файл листинга *lab7-2.lst..* Подробно объясним содержимое трёх строк файла листинга:

77 00000054 50	<1>	push	eax
78 00000055 51	<1>	push	ecx
79 00000056 52	<1>	push	edx

Рисунок 13 Строки, которые мы будем объяснять

Строка 77:

- push eax - помещает значение регистра EAX в стек
- 56 - машинный код инструкции push eax
- 66666654 - адрес в памяти, где находится эта инструкция

Строка 78:

- push ecx - помещает значение регистра ECX в стек
- 51 - машинный код инструкции push ecx
- 00000055 - следующий адрес в памяти

Строка 79:

- push edx - помещает значение регистра EDX в стек
- 52 - машинный код инструкции push edx
- 66666656 - адрес в памяти

Откроем файл с программой *lab7-2.asm* и в инструкции с двумя operandами удалим один operand.

```
13 ; ----- Вывод сообщения 'Введите В: '
14     mov eax,msg1
15     call sprint
16 ; ----- Ввод 'В'
17     mov ecx
18     mov edx,10
19     call sread
20 ; ----- Преобразование 'В' из символа в число
21     mov eax,B
22     call atoi ; Вызов подпрограммы перевода символа в число
23     mov [B],eax ; запись преобразованного числа в 'В'
24 ; ----- Записываем 'A' в переменную 'max'
25     mov ecx,[A] ; 'ecx = A'
26     mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
```

Рисунок 14 Удалим operand B из 17 строки листинга

Если выполнить трансляцию после удаления operand'a, то листинг создается лишь частично (до строки с ошибкой), а объектный файл не создается вовсе.

2.3. Задание для самостоятельной работы

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выберем из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6 (У нас это вариант 20). Создадим файл *lab7-3.asm*, впишем в него наш код, преобразуем его в исполняемый файл и проверим корректность выполнения программы.

Таблица 7.5. Значения a, b, c для задания №1.

Номер варианта	Значения a, b, c	Номер варианта	Значения a, b, c
1	17,23,45	11	21,28,34
2	82,59,61	12	99,29,26
3	94,5,58	13	84,32,77
4	8,88,68	14	81,22,72
5	54,62,87	15	32,6,54
6	79,83,41	16	44,74,17
7	45,67,15	17	26,12,68
8	52,33,40	18	83,73,30
9	24,98,15	19	46,32,74
10	41,62,35	20	95,2,61

Рисунок 15 Таблица 7.5.

```
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ ./lab7-3
Введите a: 95
Введите b: 2
Введите c: 61
Наименьшее число: 2
```

Рисунок 16 Проверяем работу нашей программы

```

1 %include 'in_out.asm'
2
3 SECTION .data
4     msg_a db 'Введите а: ', 0
5     msg_b db 'Введите б: ', 0
6     msg_c db 'Введите с: ', 0
7     result_msg db 'Наименьшее число: ', 0
8
9 SECTION .bss
10    a resb 10
11    b resb 10
12    c resb 10
13    min resb 10
14
15 SECTION .text
16     global _start
17
18 _start:
19     ; Ввод переменной а
20     mov eax, msg_a
21     call sprint
22     mov ecx, a
23     mov edx, 10
24     call sread
25     mov eax, a
26     call atoi
27     mov [a], eax
28
29     ; Ввод переменной б
30     mov eax, msg_b
31     call sprint
32     mov ecx, b
33     mov edx, 10
34     call sread
35     mov eax, b
36     call atoi
37     mov [b], eax
38
39     ; Ввод переменной с
40     mov eax, msg_c
41     call sprint
42     mov ecx, c
43     mov edx, 10
44     call sread
45     mov eax, c
46     call atoi
47     mov [c], eax
48
49     ; Находим наименьшее число
50     mov eax, [a]      ; Предполагаем, что а - наименьшее
51     mov [min], eax
52
53     ; Сравниваем с б
54     mov ebx, [b]
55     cmp eax, ebx
56     jle check_c        ; Если а <= б, переходим к проверке с
57     mov [min], ebx    ; Иначе наименьшее = б
58     mov eax, ebx
59
60 check_c:
61     ; Сравниваем текущее наименьшее с с
62     mov ecx, [c]
63     cmp eax, ecx
64     jle print_result ; Если текущее наименьшее <= с, выводим результат
65     mov [min], ecx    ; Иначе наименьшее = с
66
67 print_result:
68     ; Вывод результата
69     mov eax, result_msg
70     call sprint
71     mov eax, [min]
72     call iprintLF
73
74     ; Завершение программы
75     call quit

```

Рисунок 17 Текст программы lab7-3.asm

2. Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выберем из таблицы 7.6 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6 (У нас это вариант 20). Создадим файл lab7-4.asm, преобразуем его в исполняемый файл и проверим корректность выполнения программы для значений x и a из таблицы 7.6.

$$20 \quad \begin{cases} x - a, & x \geq a \\ 5, & x < a \end{cases} \quad (1;2) \quad (2;1)$$

Рисунок 18 Вариант 20 из таблицы 7.6

```
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ touch lab7-4.asm
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ ./lab7-4
Функция: f(x) = x - a при x >= a, иначе f(x) = 5
Ведите x: 1
Ведите a: 2
Результат f(x) = 5
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ ./lab7-4
Функция: f(x) = x - a при x >= a, иначе f(x) = 5
Ведите x: 2
Ведите a: 1
Результат f(x) = 1
psslabospickiyj@dk5n14 ~/work/arch-pc/lab07 $ 
```

Рисунок 19 Выполнение программы lab7-4.asm

```
1 %include 'in_out.asm'
2
3 SECTION .data
4     msg_x db 'Введите x: ', 0
5     msg_a db 'Введите a: ', 0
6     result_msg db 'Результат f(x) = ', 0
7     func_desc db 'Функция: f(x) = x - a при x >= a, иначе f(x) = 5', 0
8
9 SECTION .bss
10    x resb 10
11    a resb 10
12    result resb 10
13
14 SECTION .text
15     global _start
16
17 _start:
18     ; Вывод описания функции
19     mov eax, func_desc
20     call sprintLF
21
22     ; Ввод переменной x
23     mov eax, msg_x
24     call sprint
25     mov ecx, x
26     mov edx, 10
27     call sread
28     mov eax, x
29     call atoi
30     mov [x], eax
31
32     ; Ввод переменной a
33     mov eax, msg_a
34     call sprint
35     mov ecx, a
36     mov edx, 10
37     call sread
38     mov eax, a
39     call atoi
40     mov [a], eax
41
42     ; Вычисление функции f(x)
43     mov eax, [x]
44     mov ebx, [a]
45
46     ; Сравниваем x и a
47     cmp eax, ebx
```

```
47    cmp eax, ebx
48    jl less_than    ; Если x < a, переходим к ветке "меньше"
49
50    ; Ветка x ≥ a: f(x) = x - a
51    sub eax, ebx
52    jmp store_result
53
54 less_than:
55    ; Ветка x < a: f(x) = 5
56    mov eax, 5
57
58 store_result:
59    mov [result], eax
60
61    ; Вывод результата
62    mov eax, result_msg
63    call sprint
64    mov eax, [result]
65    call iprintLF
66
67    ; Завершение программы
68    call quit
```

Рисунок 20 Листинг программы lab7-4.asm

3. Вывод:

Изучил команды условного и безусловного переходов. Приобрёл навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.