

# **Лабораторная работа №7**

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

Богданюк Анна Васильевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	23
	Список литературы	24

## Список таблиц

## Список иллюстраций

4.1	Создание файла . . . . .	9
4.2	Создание файла . . . . .	9
4.3	Создание и вывод . . . . .	10
4.4	Файл lab7-1.asm . . . . .	11
4.5	Файл lab7-1.asm . . . . .	12
4.6	Создание и вывод . . . . .	13
4.7	Файл lab7-2.asm . . . . .	14
4.8	Создание и вывод . . . . .	16
4.9	Создание . . . . .	16
4.10	Листинг . . . . .	16
4.11	Строки, которые были выбраны для объяснения . . . . .	17
4.12	Файл lab7-2.lst . . . . .	17
4.13	Файл lab7-1.asm . . . . .	17
4.14	Файл lab7-3.asm . . . . .	18
4.15	Создание и вывод . . . . .	18
4.16	Файл lab7-4.asm . . . . .	20
4.17	Создание и вывод . . . . .	20

# 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. Выполнение лабораторной работы
2. Задания для самостоятельной работы

### 3 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).



## 4 Выполнение лабораторной работы

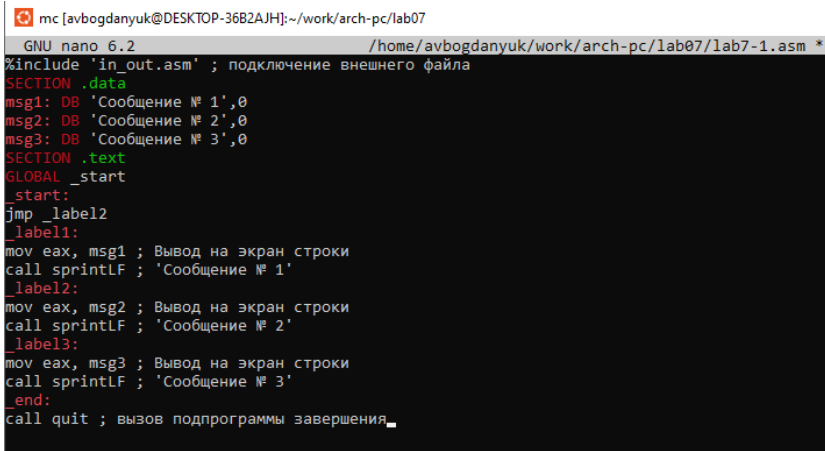
### 1.Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 4.1).

```
avbogdanyuk@DESKTOP-36B2AJH:~$ mkdir ~/work/arch-pc/lab07
avbogdanyuk@DESKTOP-36B2AJH:~$ cd ~/work/arch-pc/lab07
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание файла

В файл lab7-1.asm ввожу текст листинга(рис. 4.2).



```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения_
```

Рис. 4.2: Создание файла

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
SECTION .data
```

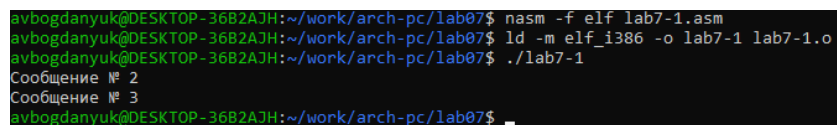
```
msg1: DB 'Сообщение № 1',0
```

```

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

```

Создаю исполняемый файл и запускаю его (рис. 4.3).



```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ _

```

Рис. 4.3: Создание и вывод

Изменяю текст файла lab7-1.asm (рис. 4.4).

```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения_
```

Рис. 4.4: Файл lab7-1.asm

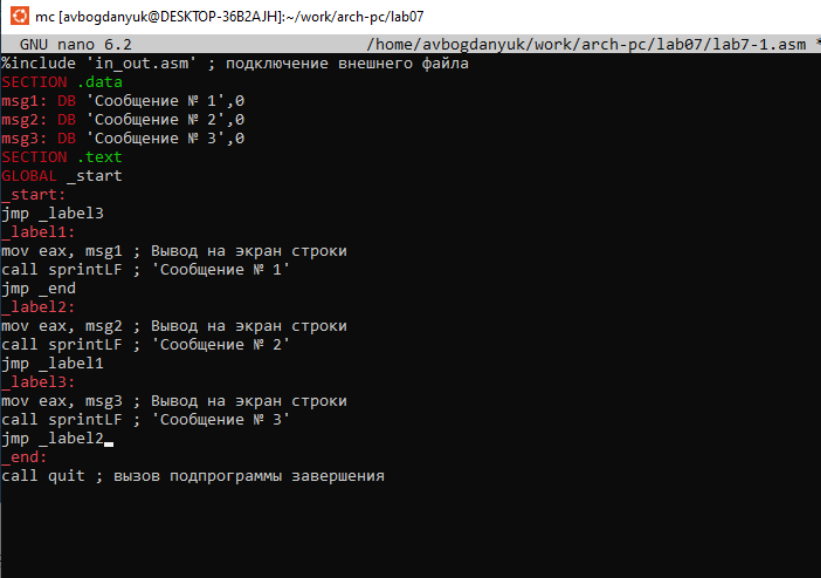
```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения_
```

```

mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Изменяю текст файла так, чтобы выводилось 3 сообщения (рис. 4.5).



```

mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.5: 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

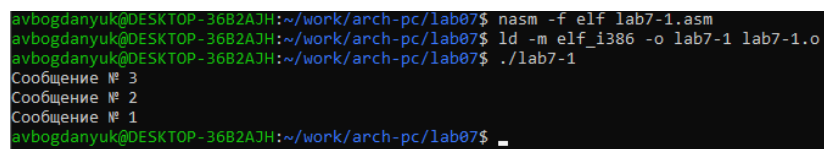
```

```

call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю его (рис. 4.6).



```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ _

```

Рис. 4.6: Создание и вывод

Создаю файл lab7-2.asm и ввожу текст листинга (рис. 4.7).

```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/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)
cmp ecx,[C] ; Сравниваем 'A' и 'C'

^G Help      ^O Write Out  ^W Where Is   ^X Cut        ^T Execute    ^C Location   ^M-U
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify    ^_ Go To Line  ^M-E
```

Рис. 4.7: 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 '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:
mov eax, msg2

```

```

call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Создаю исполняемый файл и запускаю его. Все работает корректно (рис. 4.8).

```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 32
Наибольшее число: 50
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$

```

Рис. 4.8: Создание и вывод

Создаю файл листинга для программы из файла lab7-2.asm (рис. 4.9).

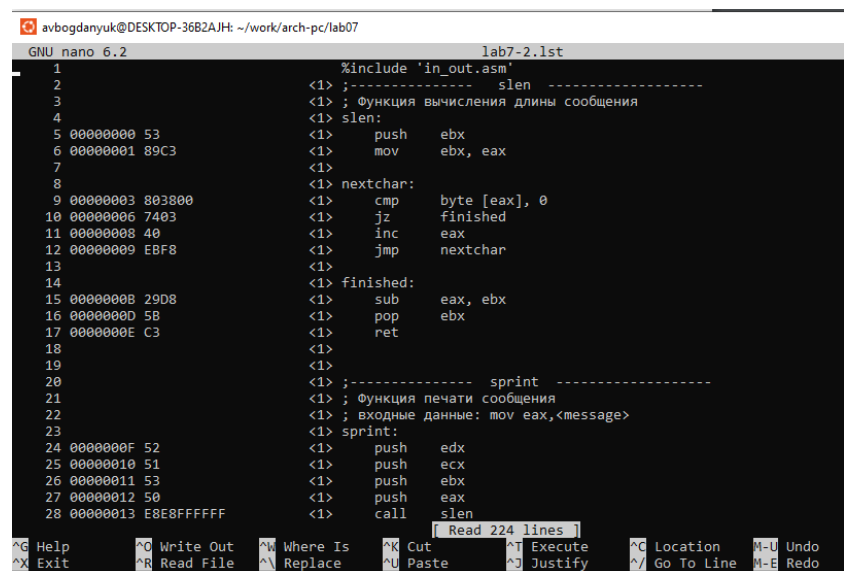
```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$

```

Рис. 4.9: Создание

Открываю листинг(рис. 4.10).



```

GNU nano 6.2 lab7-2.lst
1  %include 'in_out.asm'
2  <1> ;----- slen -----
3  <1> ; Функция вычисления длины сообщения
4  <1> slen:
5  00000000 53      <1> push    ebx
6  00000001 89C3    <1> mov     ebx, eax
7  <1>
8  <1> nextchar:
9  00000003 803800    <1> cmp     byte [eax], 0
10 00000006 7403     <1> jz      finished
11 00000008 40       <1> inc     eax
12 00000009 EBF8     <1> jmp     nextchar
13 <1>
14 <1> finished:
15 0000000B 29D8     <1> sub     eax, ebx
16 0000000D 5B       <1> pop     ebx
17 0000000E C3       <1> ret
18 <1>
19 <1>
20 <1> ;----- sprint -----
21 <1> ; Функция печати сообщения
22 <1> ; входные данные: mov eax,<message>
23 <1> sprint:
24 0000000F 52       <1> push    edx
25 00000010 51       <1> push    ecx
26 00000011 53       <1> push    ebx
27 00000012 50       <1> push    eax
28 00000013 E8E8FFFF <1> call    slen

```

Рис. 4.10: 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 к коду, также не имеет адреса и машинного кода ‘23’ - это номер строки, ‘sprintf’ - название функции, не имеет адреса и машинного кода (рис. 4.11).

```
21          <1> ; Функция печати сообщения
22          <1> ; входные данные: mov eax,<message>
23          <1> sprintf:
```

Рис. 4.11: Строки, которые были выбраны для объяснения

Выбираю инструкцию с двумя операндами, удаляю ту, которая выделена (рис. 4.12).

```
cmp eax,[B]; Сравниваем 'max(A,C)' и 'B'
```

Рис. 4.12: Файл lab7-2.lst

Пытаюсь создать файл листинг, но выдается ошибка, так как инструкция cmp не может работать, имея только один операнд (рис. 4.13).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:39: error: invalid combination of opcode and operands
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ _
```

Рис. 4.13: Файл lab7-1.asm

## 2. Задания для самостоятельной работы

Мой вариант из лабораторной работы №6 - 4. Мои значения a=8, b=88, c=68 (рис. 4.14).

```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/work/arch-pc/lab07/lab7-3.asm
%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]
jle check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]
cmp ecx, [B]
jle fin
mov ecx, [B]
mov [min], ecx

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
^C Location  ^_/ Go To Line ^M
```

Рис. 4.14: Файл lab7-3.asm

Создаю исполняемый файл и запускаю его. Результат корректный (рис. 4.15).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 8
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$
```

Рис. 4.15: 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. 4.16).

```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab07
GNU nano 6.2 /home/avbogdanyuk/work/arch-pc/lab07/lab7-4.asm
#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

Read 59 lines
Help Write Out Where Is Cut Execute Location M-L
Exit Read File Replace Paste Justify Go To Line M-E
```

Рис. 4.16: Файл lab7-4.asm

Создание исполнительного файла и запуск его. Результат корректный (рис. 4.17).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 0
Результат: 7
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 2
Результат: 8
```

Рис. 4.17: 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
```

## 5 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**