

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютеров и операционные системы

Никита Сахно НКАбд-05-23

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. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задания для самостоятельной работы.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

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

```
[nvsakhno@fedora arch-pc]$ cd lab07  
[nvsakhno@fedora lab07]$ touch lab7-1.asm  
[nvsakhno@fedora lab07]$
```

Figure 1: Создание файлов для лабораторной работы

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис.2).

```

GNU nano 7.2 /home/nvsakhno/work/arc
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 2'
78 Демидова А. В.
Архитектура ЭВМ
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение No 3'
_end:
call quit ; вызов подпрогр

```

Figure 2: Ввод текста программы из листинга 7.1

Создаю исполняемый файл и запускаю его. (рис.3).

```

[nvsakhno@fedora lab07]$ nasm -f elf lab7-1.asm
[nvsakhno@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[nvsakhno@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 3
[nvsakhno@fedora lab07]$

```

Figure 3: Запуск программного кода

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис.4).

```

GNU nano 7.2 /home/nvsakhno/work/arch-pc/lab07/lab
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit

```

Figure 4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис.5).

```

[nvsakhno@fedora lab07]$ nasm -f elf lab7-1.asm
[nvsakhno@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[nvsakhno@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 1
[nvsakhno@fedora lab07]$

```

Figure 5: Создание исполняемого файла

Затем изменяю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1`, (рис.6).

```
GNU nano 7.2 /home/nvsakhno/work/arch-pc/lab07
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:
call quit
```

Figure 6: Изменение текста программы

чтобы вывод программы был следующим: (рис.7).

```
[nvsakhno@fedora lab07]$ nasm -f elf lab7-1.asm
[nvsakhno@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[nvsakhno@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[nvsakhno@fedora lab07]$
```

Figure 7: Вывод программы

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис.8).

```
[nvsakhno@fedora lab07]$ touch lab7-2.asm
```

Figure 8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm., затем создаю исполняемый файл и проверьте его работу. (рис.9).

```
[nvsakhno@fedora lab07]$ nasm -f elf lab7-2.asm
[nvsakhno@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[nvsakhno@fedora lab07]$ ./lab7-2
Введите В: 70
Наибольшее число: 70
[nvsakhno@fedora lab07]$
```

Figure 9: Проверка работы файла

Файл работает корректно.

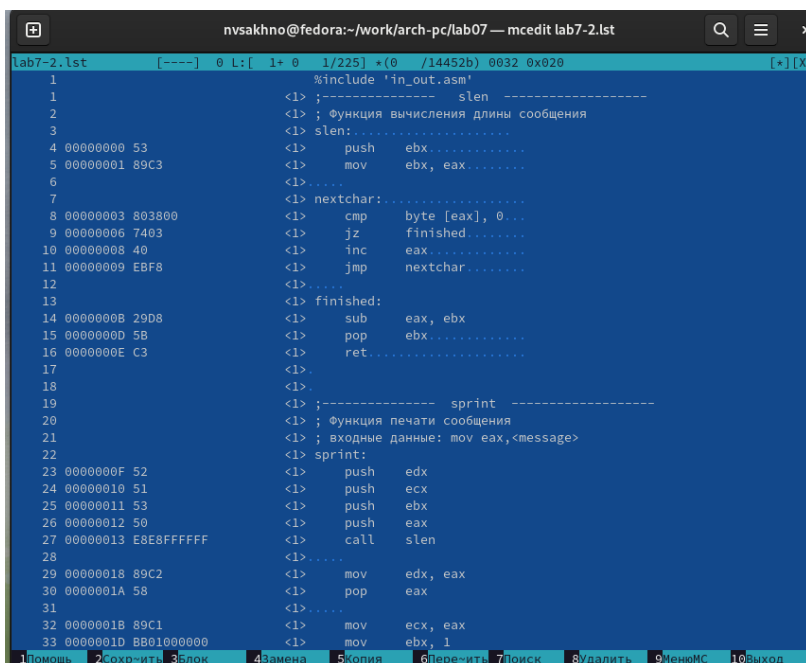
4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис.10).

```
[nvsakhno@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[nvsakhno@fedora lab07]$
```

Figure 10: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис.11).



```
lab7-2.lst  [----]  0 L: [ 1+ 0 1/225]  *(0 /14452b) 0032 0x020  [x] [X]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:
5          <1>     push    ebx
6          <1>     mov     ebx, eax
7          <1>     nextchar:
8          <1>     cmp     byte [eax], 0
9          <1>     jz      finished
10         <1>     inc     eax
11         <1>     jmp     nextchar
12         <1>     finished:
13         <1>     sub     eax, ebx
14         <1>     pop     ebx
15         <1>     ret
16         <1>
17         <1>
18         <1>
19         <1> ;----- sprint -----
20         <1> ; Функция печати сообщения
21         <1> ; входные данные: mov eax, <message>
22         <1> sprint:
23         <1>     push    edx
24         <1>     push    ecx
25         <1>     push    ebx
26         <1>     push    eax
27         <1>     call    slen
28         <1>
29         <1>     mov     edx, eax
30         <1>     pop     eax
31         <1>
32         <1>     mov     ecx, eax
33         <1>     mov     ebx, 1
```

Figure 11: Изучение файла листинга

В представленных трех строчках содержатся следующие данные: (рис.12).

```
2          <1> ; Функция вычисления длины сообщения
3          <1> slen:.....
4 00000000 53          <1>      push    ebx.....
5 00000001 89C3       <1>      mov     ebx, eax.....
```

Figure 12: Выбранные строки файла

Выполняю трансляцию с получением файла листинга. (рис.13).

```
[nvsakhno@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
[nvsakhno@fedora lab07]$
```

Figure 13: Получение файла листинга

На выходе я не получаю ни одного файла из-за ошибки:инструкция mov (единственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

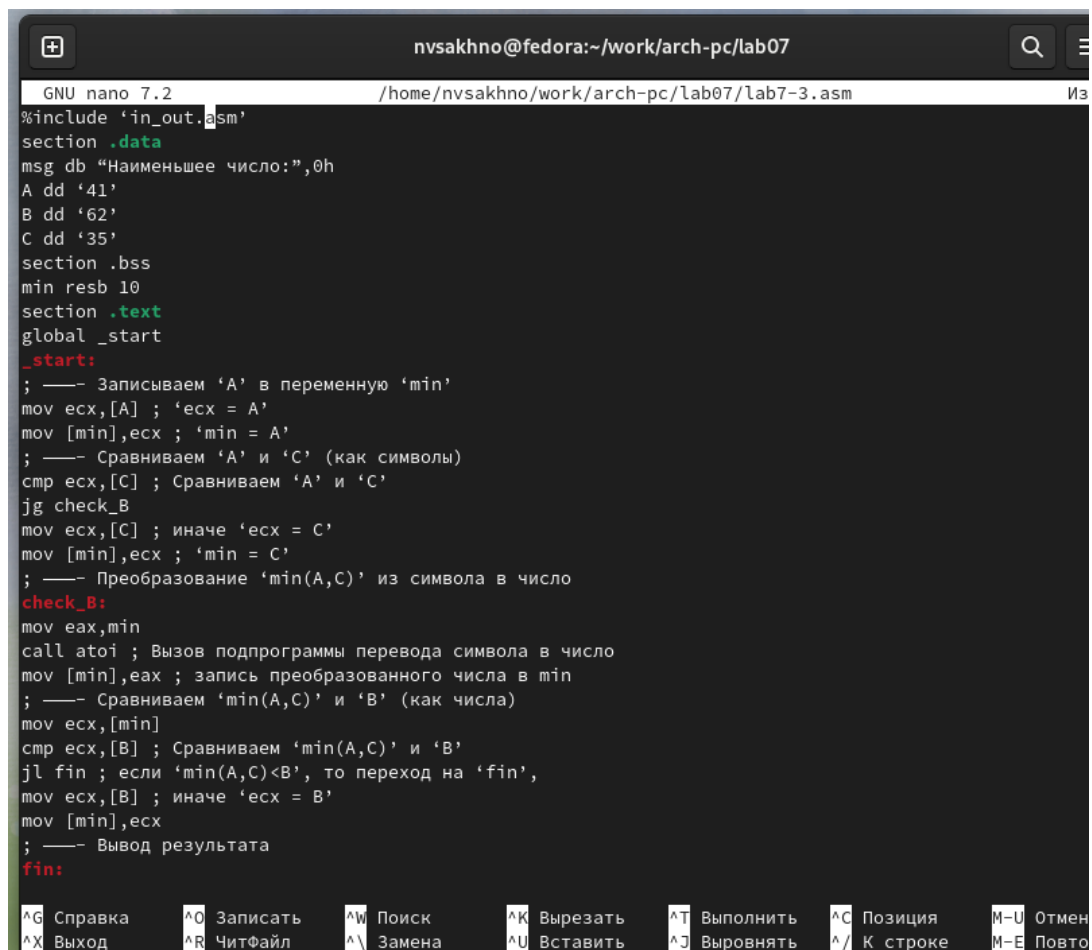
4.3 Задания для самостоятельной работы

Мой вариант под номером 10, поэтому мои значения - 41, 62 и 35. (рис.14).

```
[nvsakhno@fedora lab06]$ ./variant
Введите No студенческого билета:
1132230298
Ваш вариант: 10
[nvsakhno@fedora lab06]$
```

Figure 14: вариант

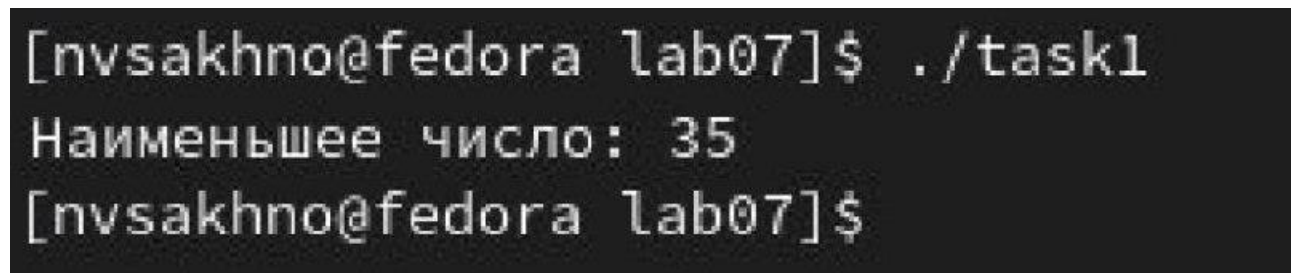
1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 (рис.15).



```
GNU nano 7.2 /home/nvsakhno/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
msg db "Наименьшее число:",0h
A dd '41'
B dd '62'
C dd '35'
section .bss
min resb 10
section .text
global _start
_start:
; --- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; --- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; --- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
; --- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; --- Вывод результата
fin:
```

Figure 15: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис.16).



```
[nvsakhno@fedora lab07]$ ./task1
Наименьшее число: 35
[nvsakhno@fedora lab07]$
```

Figure 16: Запуск файла и проверка его работы

Программа работает корректно.

5 Выводы

По итогам данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

