

Лабораторная работа №7

Архитектура компьютера и операционные системы

Бабенко Константин, НКАбд-01-23

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
2.1	Домашняя работа.....	Ошибка! Закладка не определена.
3	Выводы	6

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

Реализация переходов в NASM

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

```
abenkoka@LAPTOP-6USQFP0J:~$ mkdir ~/work/arch-pc/lab07
abenkoka@LAPTOP-6USQFP0J:~$ cd ~/work/arch-pc/lab07
abenkoka@LAPTOP-6USQFP0J:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Figure 1: Создаю файл.

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



```
GNU nano 6.2 /home/babenkoka/work/arch-pc/lab07/lab7-1.asm *
#include 'in_out.asm'
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
call sprintf
_label2:
mov eax, msg2
call sprintf
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

Figure 2: Ввожу текст программы.

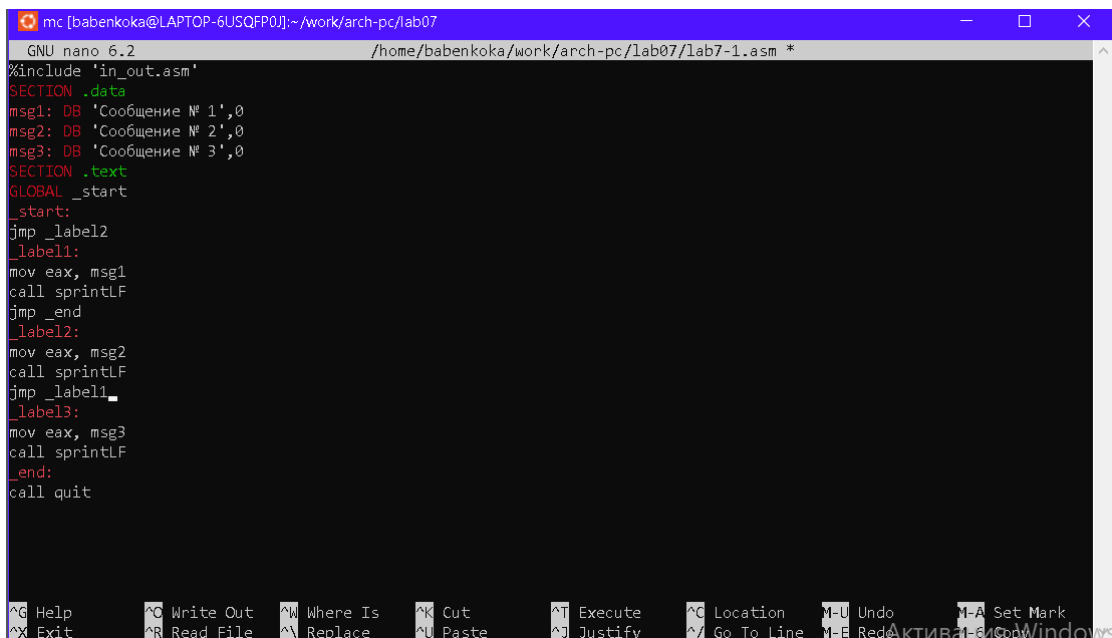
Создаю исполняемый файл и запускаю его:



```
root@LAPTOP-6USQFP0J: /home# ./lab7-1
Сообщение № 2
Сообщение № 3
```

Figure 3: Работа файла.

Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляю инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавляю инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).



```
GNU nano 6.2 /home/babenkoka/work/arch-pc/lab07/lab7-1.asm *
#include 'in_out.asm'
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
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

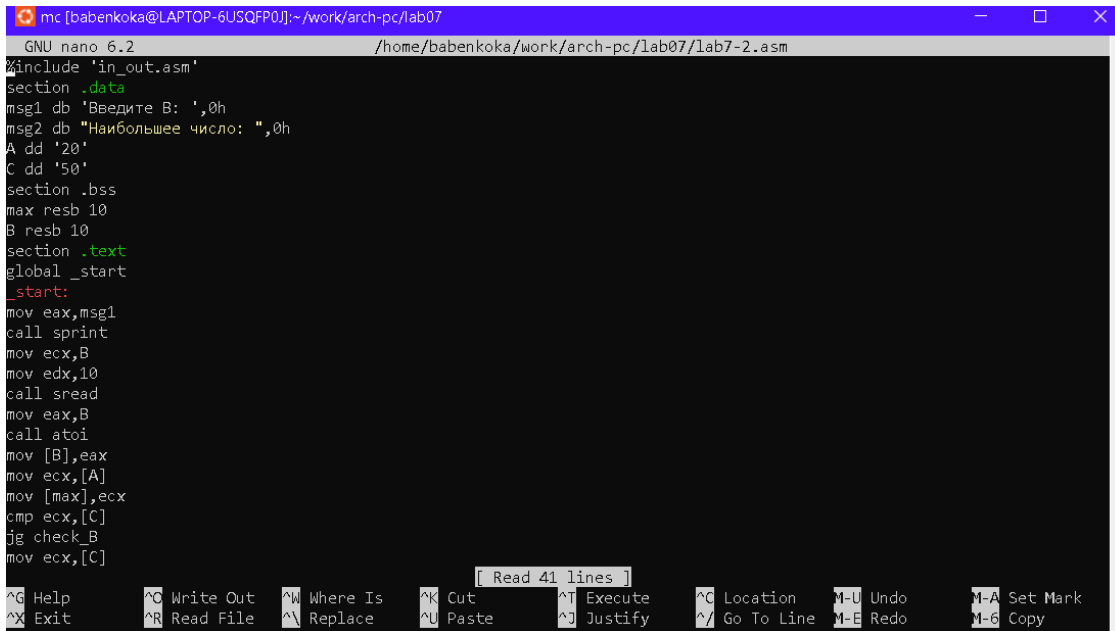
Figure 4: Измененный файл.

Создаю исполняемый файл и запускаю его:

```
root@LAPTOP-6USQFP0J:/home# ./lab7-1
Сообщение № 2
Сообщение № 1
```

Figure 5: Работа измененного файла.

3) Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучаю текст программы из листинга 7.3 и ввожу в lab7-2.asm.



```
GNU nano 6.2 /home/babenkoka/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:
mov eax,msg1
call sprint
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi
mov [B],eax
mov ecx,[A]
mov [max],ecx
cmp ecx,[C]
jg check_B
mov ecx,[C]
```

Figure 6: Введенный текст.

Создаю исполняемый файл и запускаю его:

```
root@LAPTOP-6USQFP0J:/home# ./lab7-2
Введите B: 1
Наибольшее число: 50
```

Figure 7: Работа файла.

Код 10 соответствует символу BS, он не отображается.

4) Создаю файл листинга для программы из файла lab7-2.asm:

```
babenkoka@LAPTOP-6USQFP0J:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Figure 8: Создание файла листинга.

Открываю файл листинга lab7-2.lst с помощью любого текстового редактора:

```
babenkoka@LAPTOP-6USQFP0J: ~/work/arch-pc/lab07
/home/babenkoka/work/arch-pc/lab07/lab7-2.lst [----] 0 L: [ 1+ 0 1/217] *(0 /13044b) 0032 0x020 [*][X]
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
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Figure 9: Файл листинга.

Описание строк 9,10,11:

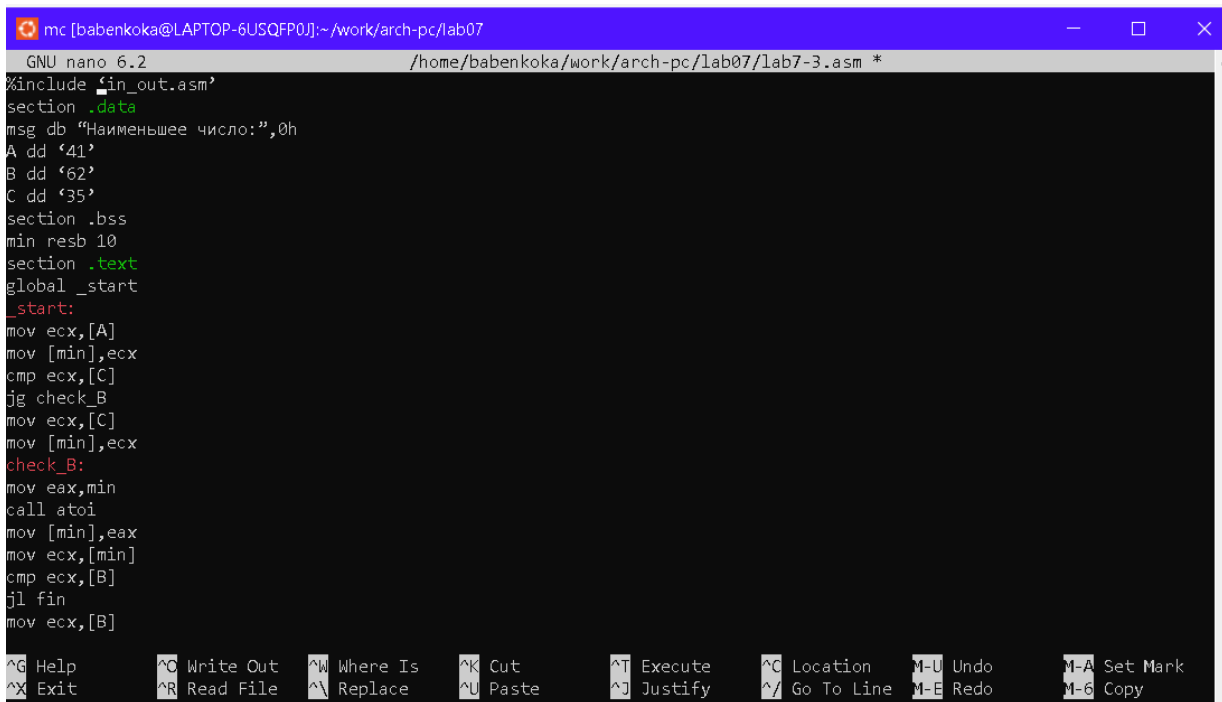
9 – Номер строки; 00000003 –адрес, 803800 –машинный код, cmp byte[eax],0 – тест программы.

10 – Номер строки; 00000006 –адрес, 7403 –машинный код, jz finished– тест программы.

11 – Номер строки; 00000008 –адрес, 40 –машинный код, inc eax– тест программы.

2.1 Домашняя работа

Ввожу программу нахождения наименьшей из 3 целочисленных переменных a, b и c :



```
mc [babenkoka@LAPTOP-6USQFP0J]~/work/arch-pc/lab07
GNU nano 6.2 /home/babenkoka/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:
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jg 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]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^N Replace    ^U Paste       ^J Justify    ^_ Go To Line  M-E Redo      M-6 Copy
```

Figure 10: Программа.

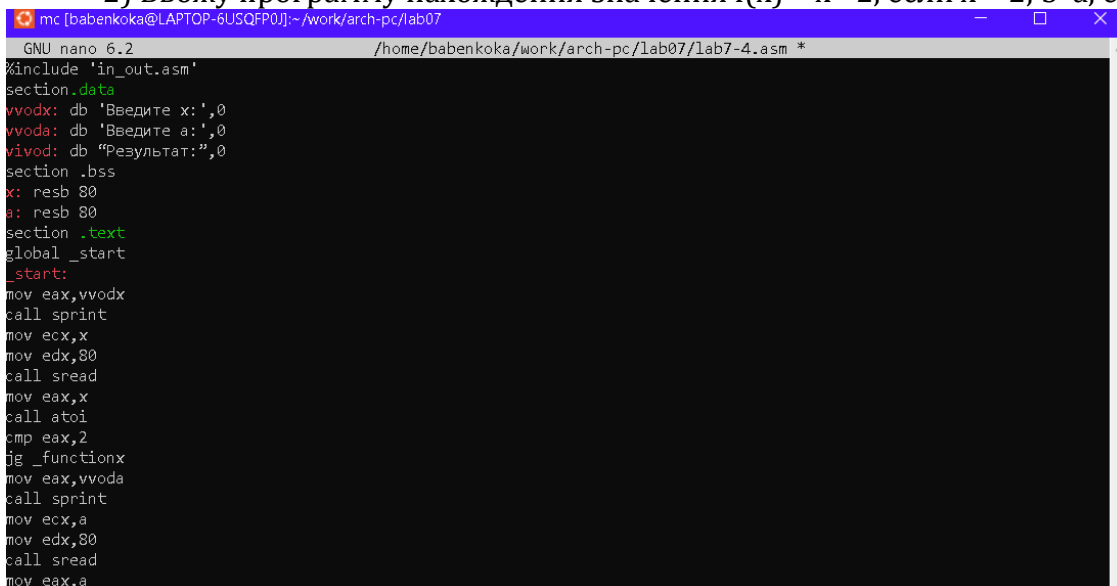
Создаю исполняемый файл и проверяю его работу:



```
root@LAPTOP-6USQFP0J:/home# chmod +x lab7-3
root@LAPTOP-6USQFP0J:/home# ./lab7-3
Наименьшее число:35
```

Figure 11: Работа файла.

2) Ввожу программу нахождения значения $f(x) = x - 2$, если $x > 2$; $3 \cdot a$, если $x \leq 2$:



```
mc [babenkoka@LAPTOP-6USQFP0J]~/work/arch-pc/lab07
GNU nano 6.2 /home/babenkoka/work/arch-pc/lab07/lab7-4.asm *
#include "in_out.asm"
section .data
vvodx: db "Введите x:",0
vvoda: db "Введите a:",0
vivod: db "Результат:",0
section .bss
x: resb 80
a: resb 80
section .text
global _start
_start:
mov eax,vvodx
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
cmp eax,2
jg _functionx
mov eax,vvoda
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
```

Figure 12: Программа.

Создаю исполняемый файл и проверяю его работу:

```
root@LAPTOP-6USQFP01:/home# ./lab7-4
Введите x:4
Результат:2
root@LAPTOP-6USQFP01:/home# ./lab7-4
Введите x:1
Введите a:3
Результат:9
root@LAPTOP-6USQFP01:/home#
```

Figure 13: Работа файла.

3 Выводы

Я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов, ознакомился с назначением и структурой файла листинга.