

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Орлов Илья Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	11
4.3	Задания для самостоятельной работы	13
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создание каталога и файла для программы	8
4.2	Сохранение программы	8
4.3	Запуск программы	8
4.4	Изменение программы	9
4.5	Запуск измененной программы	9
4.6	Изменение программы	9
4.7	Проверка изменений	10
4.8	Сохранение новой программы	10
4.9	Проверка программы из листинга	10
4.10	Проверка файла листинга	11
4.11	Удаление операнда из программы	12
4.12	Просмотр ошибки в файле листинга	12
4.13	Первая программа самостоятельной работы	13
4.14	Проверка работы первой программы	14
4.15	Вторая программа самостоятельной работы	15
4.16	Проверка работы второй программы	17

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. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной 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.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. -fig. 4.1).

```
rutnixya@rutnix-VirtualBox:~$ mkdir ~/work/arch-pc/lab07
rutnixya@rutnix-VirtualBox:~$ cd ~/work/arch-pc/lab07
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. -fig. 4.2).

```
GNU nano 7.2 /home/rutnixya/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
```

Рис. 4.2: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. -fig. 4.3).

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o in_out.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. -fig. 4.4).

```
GNU nano 7.2 /home/rutnixya/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
```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 4.5).

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o in_out.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 4.5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig. 4.6).

```
GNU nano 7.2 /home/rutnixya/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 _label3
_label1:
mov eax, msg1
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
jmp _label2
_end:
call quit
```

Рис. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. -fig. 4.7).

```

rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1.o in_out.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. -fig. 4.8).

```

GNU nano 7.2 /home/rutnixya/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]
mov [max],ecx
check_B:
mov eax,max
call atoi
mov [max],eax
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintf
call quit

```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. -fig. 4.9).

```

rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ls
in_out.asm  in_out.o  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ mc

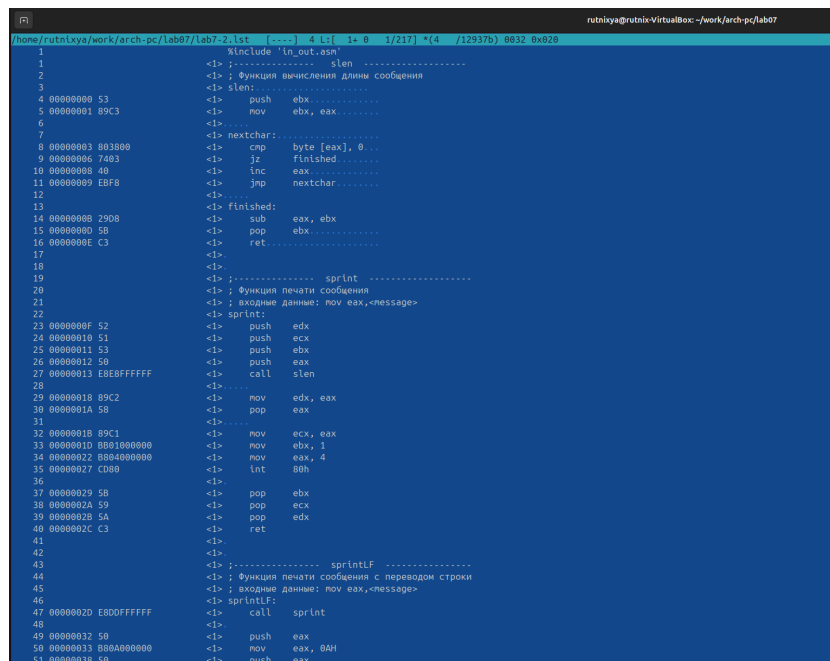
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2.o in_out.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 15
Наибольшее число: 50
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 4.9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. -fig. 4.10).



```
1      ;include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; функция вычисления длины сообщения
4      00000000 53      <1> slen
5      00000001 89C3    <1> push ebx
6      <1> mov     ebx, eax
7      <1> nextchar:
8      00000003 803800  <1> cmp     byte [eax], 0
9      00000006 7403    <1> jz      finished
10     00000008 40      <1> inc     eax
11     00000009 EBF8    <1> jmp     nextchar
12
13     <1> finished:
14     0000000B 2908    <1> sub     eax, ebx
15     0000000D 5B      <1> pop     ebx
16     0000000E C3      <1> ret
17
18     <1>
19     <1> ;----- sprintf -----
20     <1> ; функция печати сообщения
21     <1> ; входные данные: mov eax, <message>
22     <1> sprintf:
23     0000000F 52      <1> push    edx
24     00000010 51      <1> push    ecx
25     00000011 53      <1> push    ebx
26     00000012 56      <1> push    eax
27     00000013 EBEBFFFF <1> call    slen
28     <1> .....
29     00000018 89C2    <1> mov     edx, eax
30     0000001A 5B      <1> pop     eax
31     <1> .....
32     0000001D 89C1    <1> mov     ecx, eax
33     0000001D BB01000000 <1> mov     ebx, 1
34     00000022 8B40000000 <1> mov     eax, 4
35     00000027 CD80    <1> int     80h
36
37     00000029 5B      <1> pop     ebx
38     0000002A 59      <1> pop     ecx
39     0000002B 5A      <1> pop     edx
40     0000002C C3      <1> ret
41
42     <1>
43     <1> ;----- sprintfL -----
44     <1> ; функция печати сообщения с переводом строки
45     <1> ; входные данные: mov eax, <message>
46     <1> sprintfL:
47     0000002D E8D0FFFFF <1> call    sprintf
48     <1> .....
49     00000032 5B      <1> push    eax
50     00000033 B8B0A0000000 <1> mov     eax, 0Ah
51     0000003B 59      <1> push    eax
```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. -fig. 4.11).

```

GNU nano 2.2                               rutnixya@rutnix-VirtualBox: ~/work/arch-pc/lab07
#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,1
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]
mov [max],ecx
check_B:
mov eax,max
call atoi
mov [max],eax
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
fin:
mov eax,msg2
call sprint
mov eax,[max]
call sprintf
call quit

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. -fig. 4.12).

```

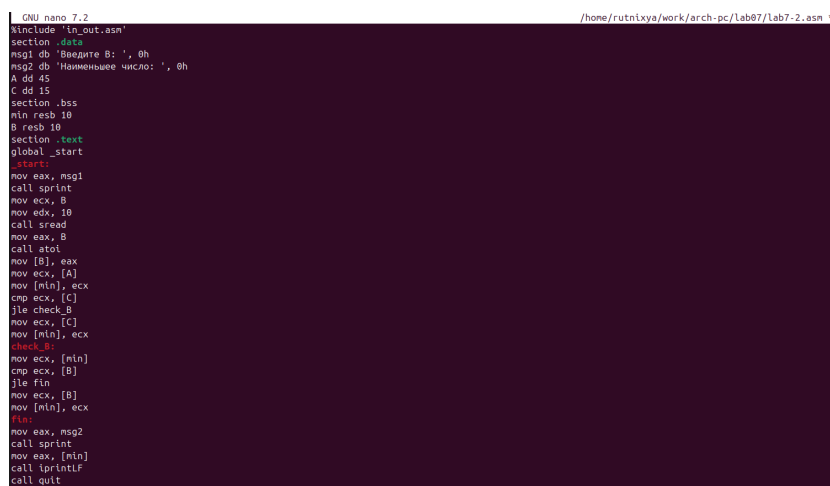
/home/rutnixya/work/arch-pc/lab07/lab7-2.lst [B---] 90 L:[167+22 189/218] *(11570/130236) 0010 0x00A
166                                <!-- quit:
167 00000000 B000000000          <!-- mov     ebx, 0.....
168 000000E0 B001000000          <!-- mov     eax, 1.....
169 000000E5 CD00                <!-- int     00h
170 000000E7 C3                 <!-- ret
2                                section .data
3 00000000 D0920082D0085D0084D0-  msg1 db "Введите B: ",0h
3 00000009 B00102008520423A20-
3 00000012 00.....           msg2 db "Наибольшее число: ",0h
4 00000012 D090000000000000100-
4 0000001C B00000019CD18000005-
4 00000025 D005200107D00000101-
4 0000002E D000000E3A2000..... A dd '20'
5 00000035 32300000            C dd '50'
6 00000039 35300000            section .bss
7                                max resb 10
8 00000000 <res Ah>            B resb 10
9 0000000A <res Ah>            section .text
10                                _start:
11                                global _start
12                                _start:
13                                mov     eax,
13                                error: Invalid combination of opcode and operands
14 000000E0 B022FFFFFF          call sprint
15 000000ED B9[0A000000]         mov     ecx,B
16 000000F2 BA0A000000          mov     edx,10
17 000000F7 E047FFFFFF          call sread
18 000000FC B0[0A000000]         mov     eax,B
19 00000101 E096FFFFFF          call atoi
20 00000106 A3[0A000000]         mov     [B],eax
21 00000108 0000[35000000]       mov     ecx,[A]
22 00000111 0900[00000000]       mov     [max],ecx
23 00000117 3000[39000000]       cmp     ecx,[C]
24 0000011D 7F0C                jg     check_B
25 0000011F 8000[39000000]       mov     ecx,[C]
26 00000125 0900[00000000]       mov     [max],ecx
27                                check_B:
28 0000012B B0[00000000]         mov     eax,max
29 00000130 E067FFFFFF          call atoi
30 00000135 A3[00000000]         mov     [max],eax
31 0000013A 8000[00000000]       cmp     ecx,[max]
32 00000140 7F0C                jg     fin

```

Рис. 4.12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Буду использовать свой вариант - седьмой - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. -fig. 4.13).



```
GNU nano 7.2 /home/rutnxya/work/arch-pc/lab07/lab7-2.asm *
#include "in_out.asm"
section .data
msg1 db "Введите B: ", 0h
msg2 db "Наименьшее число: ", 0h
A dd 45
C dd 15
section .bss
min 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 [min], ecx
mov ecx, [min]
cmp ecx, [B]
jle check_B
jle fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintf
call quit
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd 45
C dd 15

section .bss
min resb 10
B resb 10

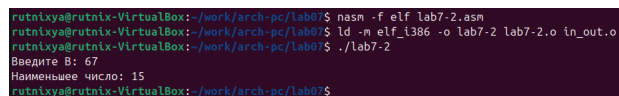
section .text
global _start
_start:
mov eax, msg1
```

```

call sprint
mov ecx, B
mov edx, 10
call sread
mov eax, B
call atoi
mov [B], eax
mov ecx, [A]
mov [min], ecx
cmp ecx, [C]
jle check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov ecx, [min]
cmp ecx, [B]
jle fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

```

Проверяю корректность написания первой программы (рис. -fig. 4.14).



```

rutnixya@rutnix-VirtualBox: /work/arch-pc/lab0$ nasm -f elf lab7-2.asm
rutnixya@rutnix-VirtualBox: /work/arch-pc/lab0$ ld -m elf_i386 -o lab7-2 lab7-2.o in_out.o
rutnixya@rutnix-VirtualBox: /work/arch-pc/lab0$ ./lab7-2
Введите B: 67
Наименьшее число: 15
rutnixya@rutnix-VirtualBox: /work/arch-pc/lab0$

```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. -fig. 4.15).



```

GNU nano 7.2                               /home/rutnixya/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите x: ', 0
msg2 db 'Введите a: ', 0
msg3 db 'Результат: ', 0
errMsg db 'Ошибка ввода', 0
section .bss
x resd 1
a resd 1
section .text
global _start
_start:
mov eax, msg1
call sprint
mov ecx, x
mov edx, 4
call sread
mov eax, x
call atoi
mov [x], eax
mov eax, msg2
call sprint
mov ecx, a
mov edx, 4
call sread
mov eax, 0
call atoi
mov [a], eax
mov ebx, [x]
mov ebx, [a]
cmp eax, ebx
je equal
add eax, ebx
jmp print_result
equal:
mov eax, 6
mul ebx
print_result:
push eax
mov eax, msg3
call sprint
pop eax
call iprintf
call quit

```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```

#include 'in_out.asm'

section .data
msg1 db 'Введите x: ', 0
msg2 db 'Введите a: ', 0
msg3 db 'Результат: ', 0
errMsg db 'Ошибка ввода', 0

section .bss
x resd 1
a resd 1

section .text
global _start
_start:
mov eax, msg1

```

```

call sprint
mov ecx, x
mov edx, 4
call sread
mov eax, x
call atoi
mov [x], eax
mov eax, msg2
call sprint
mov ecx, a
mov edx, 4
call sread
mov eax, a
call atoi
mov [a], eax
mov eax, [x]
mov ebx, [a]
cmp eax, ebx
je equal
add eax, ebx
jmp print_result
equal:
mov eax, 6
mul ebx
print_result:
push eax
mov eax, msg3
call sprint
pop eax

```



```
call iprintLF
```

```
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. -fig. 4.16).

```
rutniya@rutnix-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
rutniya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o in_out.o
./lab7-3
Введите х: 1
Введите а: 1
Результат: 6
rutniya@rutnix-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите х: 2
Введите а: 1
Результат: 3
rutniya@rutnix-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 4.16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.