

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Алексеев Илья Сергеевич

Содержание

Список иллюстраций

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

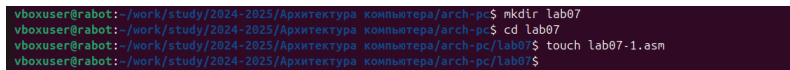
3 Теоретическое введение

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. 1).



```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab07
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab07
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab07-1.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 2).

File Edit Search View Document Help

```
%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. 2: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

```
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab07-1.asm
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Сообщение № 2
Сообщение № 3
```

Рис. 3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

~/work/study/2024-2025/Архитектура компью

File Edit Search View Document Help

```
%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: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

```
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab07-1.asm
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Сообщение № 2
Сообщение № 1
```

Рис. 5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).

```
%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 sprintfLF
```

```
jmp _end
```

```
_label2:
```

```
mov eax, msg2
```

```
call sprintfLF
```

```
jmp _label1
```

```
_label3:
```

```
mov eax, msg3
```

```
call sprintfLF
```

```
jmp _label2
```

```
|
```

```
_end:
```

```
call quit
```

Рис. 6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).

```
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab07-1.asm
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
vboxuser@rabort: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).


```
File Edit Search View Document Help
%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:
```

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

```
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab07-1.asm
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Введите B: 50
Наибольшее число: 50
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Введите B: 10
Наибольшее число: 50
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Введите B: 90
Наибольшее число: 90
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-1
Введите B: 35
Наибольшее число: 50
```

Рис. 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 mouserpad (рис. 10).

file	Edit	Search	View	Document	Help
1					
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					<1> finished:
13					<1> sub eax, ebx
14	0000000B 29D8				<1> pop ebx
15	0000000D 5B				<1> ret
16	0000000E C3				<1>
17					<1>
18					<1>
19					<1> ;----- sprint -----
20					<1> ; Функция печати сообщения
21					<1> ; входные данные: mov eax,<message>
22					<1> sprint:
23	0000000F 52				<1> push edx
24	00000010 51				<1> push ecx
25	00000011 53				<1> push ebx
26	00000012 50				<1> push eax
27	00000013 E8E8FFFFFF				<1> call slen
28					<1>
29	00000018 89C2				<1> mov edx, eax
30	0000001A 5B				<1> pop eax
31					<1>
32	0000001B 89C1				<1> mov ecx, eax
33	0000001D B801000000				<1> mov ebx, 1
34	00000022 B804000000				<1> mov eax, 4

Рис. 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. 11).

*~/work/study/2024-2025/Архитект

File Edit Search View Document

```
5
mov eax,msg1
5 call sprint
J
J mov ecx,B
J mov edx,10
call sread
5
mov eax,B
5 call atoi
J mov [B],eax

5 mov ecx,[A]
mov [max],ecx
5
J cmp ecx,[C]
J jg check_B
J mov ecx,[C]
J mov [max],ecx
J
J check_B:
mov eax,|
5 call atoi
mov [max],eax
5
J mov ecx,[max]
cmp ecx,[B]
5 jg fin
```

Рис. 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. 12).

```
35                                     check_B:
36                                     mov eax,
37      *****                      error: invalid combination of opcode and operands
37 00000130 E867FFFFFF               call atoi
38 00000135 A3[00000000]             mov [max],eax
39
```

Рис. 12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

При выполнении лабораторной №7, мною не было замечено кода который выдаст мне вариант для самостоятельной работы, поэтому я использую вариант №20 из лабораторной №6. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).

```
~/WORK/Study/2024-2025  
File Edit Search View Document Help  
%include 'in_out.asm'  
  
SECTION .data  
msg1 db 'Введите B: ', 0h  
msg2 db 'Наименьшее число: ', 0h  
A dd '95'  
C dd '61'  
  
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]
```

Рис. 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 '95'
C dd '61'

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

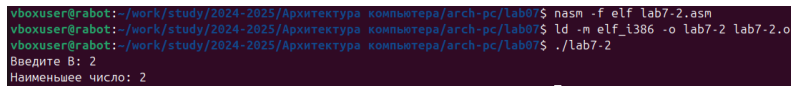
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb 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. 14).



```
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 2
Наименьшее число: 2
```

Рис. 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. 15).

```

#include 'in_out.asm'
SECTION .data
msg_x db 'Введите значение переменной x: ', 0
msg_a db 'Введите значение переменной a: ', 0
msg_result db 'Результат: ', 0
SECTION .bss
x resb 10
a resb 10
result resd 1
SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 10
call sread
mov eax, x
call atoi
mov edi, eax
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 10
call sread
mov eax, a
call atoi
mov esi, eax
cmp edi, esi
jge computeFunction
mov eax, 5

```

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

```

#include 'in_out.asm'
SECTION .data
msg_x db 'Введите значение переменной x: ', 0
msg_a db 'Введите значение переменной a: ', 0
msg_result db 'Результат: ', 0
SECTION .bss
x resb 10
a resb 10
result resd 1
SECTION .text
GLOBAL _start
_start:
mov eax, msg_x

```

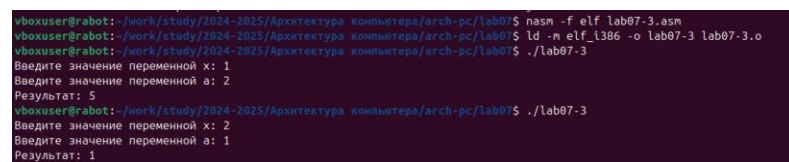


```

call sprint
mov ecx, x
mov edx, 10
call sread
mov eax, x
call atoi
mov edi, eax
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 10
call sread
mov eax, a
call atoi
mov esi, eax
cmp edi, esi
jge computeFunction
mov eax, 5
jmp storeResult
computeFunction:
sub edi, esi
mov eax, edi
storeResult:
mov [result], eax
mov eax, msg_result
call sprint
mov eax, [result]
call iprintLF
call quit

```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 16).



```

vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab07-3.asm
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab07-3 lab07-3.o
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат: 5
vboxuser@rabort: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab07-3
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат: 1

```

Рис. 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.