

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Машков Илья Евгеньевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файла листинга	11
3.3	Выполнение заданий для самостоятельной работы	12
4	Выводы	19
5	Список литературы	20

Список иллюстраций

3.1	Директория lab07.	6
3.2	Создание рабочего файла.	6
3.3	Код программы.	7
3.4	Результат выполнения программы.	7
3.5	Изменённый код программы.	8
3.6	Результат второго выполнения программы.	8
3.7	Создание файла lab7-2.asm.	8
3.8	Код программы.	10
3.9	Результат выполнения программы.	11
3.10	Строчки программы в листинге.	11
3.11	Ошибки при трансляции.	12
3.12	Код программы.	13
3.13	Результаты выполнений программы.	14
3.14	Создание .asm файла.	14
3.15	Код программы.	15
3.16	Результаты выполнений программы.	17

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 файла листинга
3. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Для начала в папке локального репозитория я создаю директорию **lab07** для дальнейшей работы в ней (рис. [3.1]).

```
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab07
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ ls
CHANGELOG.md  lab04  lab07  LICENSE  presentation  README.md
config        lab05  labs   Makefile  README.en.md  template
COURSE        lab06  lamkdir  prepare  README.git-flow.md
```

Рис. 3.1: Директория lab07.

Далее я перехожу в эту директорию и создаю файл **lab7-1.asm** с помощью команды **'touch'**, а также копирую файл **in_out.asm** (Рис. [3.2]).

```
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ls
lab7-1.asm
iemashkov@iemashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 3.2: Создание рабочего файла.

Затем я ввожу код в .asm файл (Рис. [3.3]).

```

#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. 3.3: Код программы.

Создаю исполняемый файл и запускаю программу (Рис. [3.4]). В выводе программы я получаю символ **‘Сообщение №2’** и **‘Сообщение №3’**.

```

lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -n elf_i386 -o lab7-1 lab7-1.o
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ 

```

Рис. 3.4: Результат выполнения программы.

Теперь я меняю в программе пару строк, чтобы получился такой вывод: **‘Сообщение №3’**, **‘Сообщение №2’**, **‘Сообщение №1’**. (Рис. [3.5]).

```

#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. 3.5: Изменённый код программы.

Создаю исполняемый файл и запускаю его (Рис. [3.6]). В выводе получаю именно то, что мне и надо было.

```

ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ 

```

Рис. 3.6: Результат второго выполнения программы.

Создаю файл **lab7-2.asm** (Рис. [3.7]).

```

ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ 

```

Рис. 3.7: Создание файла lab7-2.asm.

Ввожу код программы (Рис. [3.8]).

```

%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
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

```

Рис. 3.8: Код программы.

Создаю исполняемый файл и запускаю программу (Рис. [3.9]). В выводе получаю число **50**, т.к. это и есть максимальное из всех значений, но если в **'В'** ввести число, которое будет больше 50-ти, то программа выведет введенное с клавиатуры число.

```
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите В: 1
Наибольшее число: 50
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 3.9: Результат выполнения программы.

3.2 Изучение структуры файла листинга

С помощью команды **'nasm -f elf -l lab7-2.lst lab7-2.asm'** создаю файл листинга, внимательно изучаю его структуру. После чего я выбрал три строчки к которым буду писать пояснение (Рис. [3.10]).

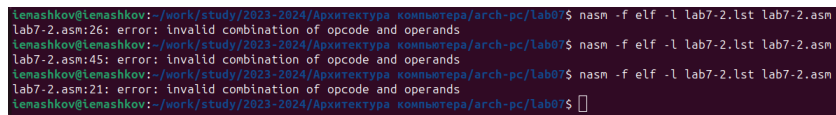
```
mov ecx,[max]
cmp ecx,[B]
jg fin
```

Рис. 3.10: Строчки программы в листинге.

1. Командой **'mov ecx,[max]'** мы перемещаем в регистр **ecx** значения из **max**, где лежат результаты сравнения **А** и **С**.
2. Команда **'cmp ecx,[B]'** позволяет нам провести сравнение **В** со значениями **А** и **С**.

3. Если ' $\max(A, C) > B$ ' то команда '**lg**' позволяет нам перейти на финальный этап программы, обозначенный меткой '**fin**'. Команда '**lg**' позволяет перейти к следующей метке только если мы получаем такое сравнение: ' $A > B$ ' - иначе будет использоваться '**jl**', '**jle**', '**jge**' и т.д..

Далее по заданию мне нужно было убрать из любой инструкции с двумя операндами один операнд и выполнить трансляцию с получением файла листинга. Я сделал это в разных местах, но при трансляции вылетала ошибка, а изменённый файл листинга не создавался, т.к. программа не может работать при отсутствии одного операнда. (Рис. [3.11]).



```
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:26: error: invalid combination of opcode and operands
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:45: error: invalid combination of opcode and operands
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:21: error: invalid combination of opcode and operands
ienashkov@ienashkov:~/work/study/2023-2024/Архитектура_компьютера/arch-pc/lab07$
```

Рис. 3.11: Ошибки при трансляции.

3.3 Выполнение заданий для самостоятельной работы

1. Создаю файл **lab7-3.asm** и ввожу код программы, которая будет находить минимальное из значений **A**, **B** и **C**. Для этого я немного поменял код программы '**lab7-2.asm**' (Рис. [3.12]).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db "Наименьшее число: ",0h
4 A dd '54'
5 B dd '62'
6 C dd '87'
7 section .bss
8 min resb 10
9 section .text
10 global _start
11 _start:
12 ; ----- Записываем 'A' в переменную 'min'
13 mov ecx,[A] ; 'ecx = A'
14 mov [min],ecx ; 'min = A'
15 ; ----- Сравниваем 'A' и 'C' (как символы)
16 cmp ecx,[C] ; Сравниваем 'A' и 'C'
17 jl check_B ; если 'A<C', то переход на метку 'check_B',
18 mov ecx,[C] ; иначе 'ecx = C'
19 mov [min],ecx ; 'min = C'
20 ; ----- Преобразование 'min(A,C)' из символа в число
21 check_B:
22 mov eax,min
23 call atoi ; Вызов подпрограммы перевода символа в число
24 mov [min],eax ; запись преобразованного числа в 'min'
25 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
26 mov ecx,[min]
27 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
28 jl fin ; если 'min(A,C)<B', то переход на 'fin',
29 mov ecx,[B] ; иначе 'ecx = B'
30 mov [min],ecx
31 ; ----- Вывод результата
32 fin:
33 mov eax, msg1
34 call sprint ; Вывод сообщения 'Наименьшее число: '
35 mov eax,[min]
36 call iprintLF ; Вывод 'min(A,B,C)'
37 call quit ; Выход

```

Рис. 3.12: Код программы.

Создаю исполняемый файл и запускаю его со значениями из 5-го варианта: '54', '62', '87' (Рис. [3.13]).

```

Наименьшее число: 0
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-3.lst lab7-3.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -n elf_i386 -o lab7-3 lab7-3.o
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-3
Введите В: 62
Наименьшее число: 54
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ █

```

Рис. 3.13: Результаты выполнения программы.

P.s.: Изначально я запущал вариант программы, где значение **В** запрашивалось с клавиатуры, но, как видно из (Рис. [3.12]), я поменял программу так, чтобы ничего с клавиатуры не запрашивалось и в выводе получал число '54', что и является правильным ответом.

2. Создаю файл **lab7-4.asm** (Рис. [3.14]).

```

lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-4.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst lab7-3.asm lab7-3.o
lab7-1 lab7-1.o lab7-2.asm lab7-3 lab7-3.lst lab7-4.asm
lenashkov@lenashkov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ █

```

Рис. 3.14: Создание .asm файла.

Ввожу код программы, которая будет запрашивать значения **х** и **а** с клавиатуры и после сравнения их значений вычислять значения функции, которые указаны напротив результатов сравнения в варианте 5 (Рис. [3.16]).

```

%include 'in_out.asm'
SECTION .data
msg1 DB 'Введите x: ', 0h
msg2 DB 'Введите a: ', 0h
msg3 DB 'Результат: ', 0h

SECTION .bss
x resb 11
a resb 11
res resb 12

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: ' и ввод 'x' с клавиатуры
mov eax, msg1
call sprint
mov ecx, x
mov edx, 10
call sread

; ----- Вывод сообщения 'Введите a: ' и ввод 'a' с клавиатуры
mov eax, msg2
call sprint
mov ecx, a
mov edx, 10
call sread

mov eax, x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x], eax ; запись преобразованного числа в 'x'
mov eax, a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a], eax ; запись преобразованного числа в 'a'

; ----- Сравниваем 'x' и 'a' (как числа)
mov eax, [x]
mov ecx, [a]
cmp eax, ecx ; Сравниваем 'a' и 'x'
jg _label1 ; если 'a>x', то переход на метку '_label1'
mov ebx, 15
mov eax, ebx ; иначе 'eax = 15'
mov [res], eax ; 'res = 15'
jmp _end

; ----- Вычисление выражения '2*(x - a)'
_label1:
mov ebx, 2
sub eax, ecx ; 'eax = eax + ecx = x - a'
mul ebx ; 'eax = ebx*eax = 2*(x-a)'
mov [res], eax ; 'res = 2*(x-a)'
jmp _end

; ----- Вывод результата
_end:
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax, [res]
call iprintLF ; Вывод
call quit ; Вызов подпрограммы завершения

```

Рис. 3.15: Код программы.

Далее создаю исполняемый файл и запускаю программу в первый раз со значениями **(1;2)**, а потом и с **(2;1)** (Рис. [??])


```

%include 'in_out.asm'
SECTION .data
msg1 DB 'Введите x: ', 0h
msg2 DB 'Введите a: ', 0h
msg3 DB 'Результат: ', 0h

SECTION .bss
x resb 11
a resb 11
res resb 12

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: ' и ввод 'x' с клавиатуры
mov eax, msg1
call sprint
mov ecx, x
mov edx, 10
call sread

; ----- Вывод сообщения 'Введите a: ' и ввод 'a' с клавиатуры
mov eax, msg2
call sprint
mov ecx, a
mov edx, 10
call sread

mov eax, x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x], eax ; запись преобразованного числа в 'x'
mov eax, a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a], eax ; запись преобразованного числа в 'a'

; ----- Сравниваем 'x' и 'a' (как числа)
mov eax, [x]
mov ecx, [a]
cmp eax, ecx ; Сравниваем 'a' и 'x'
jg _label1 ; если 'a>x', то переход на метку '_label1'
mov ebx, 15
mov eax, ebx ; иначе 'eax = 15'
mov [res], eax ; 'res = 15'
jmp _end

; ----- Вычисление выражения '2*(x - a)'
_label1:
mov ebx, 2
sub eax, ecx ; 'eax = eax + ecx = x - a'
mul ebx ; 'eax = ebx*eax = 2*(x-a)'
mov [res], eax ; 'res = 2*(x-a)'
jmp _end

; ----- Вывод результата
_end:
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax, [res]
call iprintLF ; Вывод
call quit ; Вызов подпрограммы завершения

```

Рис. 3.16: Результаты выполнений программы.

Результаты я проверил, поэтому могу сказать, что программа отработала верно.

4 Выводы

При выполнении данной лабораторной работы я освоил переходы в NASM, а также научился создавать и разбираться в файле листинга.

5 Список литературы

Архитектура ЭВМ