

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Жернаков Данила Иванович

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	10
4.3	Выполнение заданий для самостоятельной работы (вар. 16)	11
5	Выводы	14

Список иллюстраций

4.1	Создание каталога и файла в ней	8
4.2	Исполнение программы	8
4.3	Исполнение программы	8
4.4	Редактирование файла	9
4.5	Исполнение программы	9
4.6	Исполнение программы для разных значений В	10
4.7	Исполнение программы	12
4.8	Написание программы	13
4.9	Исполнение программы	13

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 файла листинга
3. Задание для самостоятельной 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 для лабораторной работе №7, перехожу в него и создаю файл lab7-1.asm (рис. 4.1).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ cd labsasm/  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ mkdir lab07  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ cd lab07
```

Рис. 4.1: Создание каталога и файла в ней

Ввожу в файл lab7-1.asm текст программы с использованием функции jmp (рис. ??). Создаю исполняемый файл и запускаю его (рис. 4.2).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ cp ~/3арпыски/in_out.asm in_out.asm  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-1.asm  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-1  
Сообщение № 2  
Сообщение № 3  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $
```

Рис. 4.2: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу (рис. 4.3).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-1.asm  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-1  
Сообщение № 2  
Сообщение № 1  
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $
```

Рис. 4.3: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение No 3', потом 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу (рис. 4.4).

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл и проверяю корректность работы программы (рис. 4.5). Программа отработала корректно.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $
```

Рис. 4.5: Исполнение программы

Создаю файл lab7-2.asm Ввожу в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 4.6). Программа сработала корректно.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-2.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-2
Введите B: 4
Наибольшее число: 50
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-2
Введите B: 36
Наибольшее число: 50
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-2
Введите B: 103
Наибольшее число: 103
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $
```

Рис. 4.6: Исполнение программы для разных значений B

4.2 Изучение структуры файла листинга

Создание файла листинга и его просмотр в текстовом редакторе(рис. ??).

```
5 00000001 89C3      <1>      mov     ebx, eax
6                                     <1>
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>
13                                     <1> finished:
14 0000000B 29D8      <1>      sub     eax, ebx
15 0000000D 5B         <1>      pop     ebx
16 0000000F C3         <1>      ret
17                                     <1>
18                                     <1>
19                                     <1> ;----- sprint -----
20                                     <1> ; Функция печати сообщения
21                                     <1> ; входные данные: mov eax, <message>
```

1. В строке 5 содержится собственно номер строки[5], адрес[00000001], машинный код[89C3] и содержимое строки кода[mov ebx, eax].
2. В строке 11 содержится собственно номер строки[11], адрес[00000009], машинный код[EBF8] и содержимое строки кода[jmp nextchar].
3. В строке 14 содержится собственно номер строки[14], адрес[0000000B], машинный код[29D8] и содержимое строки кода[sub eax, ebx].

Открываю файл lab7-2.asm и удаляю в инструкции cmp вторгг операнд (рис. ??). Открытие файла листинга после трансляции (рис. ??). Если в коде появляется ошибка, то её описание появится в файле листинга.

```
zhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
zhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ mcedit lab7-2.lst
```

```
28      *****      cmp ecx ; Сравниваем 'A' и 'C'
28      *****      error: invalid combination of opcode
29 0000011C 7F0C      jg check_B ; если 'A>C', то переход на
```

4.3 Выполнение заданий для самостоятельной работы

(вар. 16)

Создаю файл lab7-3.asm, пишу в нём программу для нахождения наименьшей из трёх целочисленных переменных a, b и c.

Текст программы в файле lab7-3.asm:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число из 44, 74 и 17 - это ",0h
A dd 44
B dd 74
C dd 17

section .bss
max resb 10

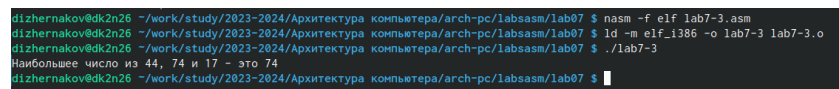
section .text
global _start
_start:
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'B'
cmp ecx, [B]
jg check_C ; если 'A>B', то переход на метку 'check_C',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx ; 'max = B'
check_C:
mov eax,max
mov ecx,[max]
```

```

cmp ecx,[C] ; Сравниваем 'max(A,B)' и 'C'
jg fin ; если 'max(A,B)>C', то переход на 'fin',
mov ecx,[C] ; иначе 'ecx = C'
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

```

Создаю исполняемый файл и проверяю его работу (рис. 4.8). Программа отработала корректно.



```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-3
Наибольшее число из 44, 74 и 17 - это 74
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $

```

Рис. 4.7: Исполнение программы

Создаю файл lab7-4.asm, пишу в нём программу, которая для введённых с клавиатуры значений x и a вычисляет значение функции $f(x)$, которая равна xa при $x \geq 4$, когда $x < 4$, то $x+4$ и выводит результат вычислений.

```

lab7-3.asm [-----] 5 L: [ 1+12 13/ 34] *(231 / 960b) 0045 0x02D
1 include "in_out.asm"
2 section .data
3 msg1 db "Введите B: ",0h
4 msg2 db "Наибольшее число из 44, 74 и 17 - это ",0h
5 A dd 44
6 B dd 74
7 C dd 17
8 section .bss
9 max resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Записываем 'A' в переменную 'max'
14 mov ecx,[A] ; 'ecx' = A'
15 mov [max],ecx ; 'max' = A'
16 ; ----- Сравниваем 'A' и 'B'
17 cmp ecx, [B]
18 jg check_C ; если 'A>B', то переход на метку 'check_C',
19 mov ecx,[B] ; иначе 'ecx' = B'
20 mov [max],ecx ; 'max' = B'
21 check_C:
22 mov eax,max
23 mov ecx,[max]
24 cmp ecx,[C] ; Сравниваем 'max(A,B)' и 'C'
25 jg fin ; если 'max(A,B)>C', то переход на 'fin',
26 mov ecx,[C] ; иначе 'ecx' = C'
27 mov [max],ecx
28 ; ----- Вывод результата
29 fin:
30 mov eax, msg2
31 call sprint ; Вывод сообщения 'Наибольшее число: '
32 mov eax,[max]
33 call iprintf ; Вывод 'max(A,B,C)'
34 call quit ; Выход

```

Рис. 4.8: Написание программы

Создаю исполняемый файл и проверяю его работу для пар x и a (1,1) и (7,1)
(рис. 4.9). Программа отработала верно.

```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ nasm -f elf lab7-4.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-4
Введите a: 1
Введите x: 7
Результат выполнения функции: 7
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-4
Введите a: 2
Введите x: 8
Результат выполнения функции: 16
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $ ./lab7-4
Введите a: 1
Введите x: 1
Результат выполнения функции: 5
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab07 $

```

Рис. 4.9: Исполнение программы

5 Выводы

В ходе выполнения лабораторной работы я освоил принципы условного и безусловного перехода в NASM.