

Отчёта по лабораторной работе №7

Дисциплина: архитектура компьютера

Худдыева Дженнет

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 файлы листинга	12
4.3	Задания для самостоятельной работы	14
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Создание файла	8
4.2	Ввод текста	8
4.3	Запуск исполняемого кода	9
4.4	Изменение текста программы	9
4.5	Создание исполняемого файла	10
4.6	изменение текста программы	10
4.7	Вывод программы	11
4.8	Создание файла	11
4.9	Ввод текста программы	12
4.10	Проверка работы файла	12
4.11	Создание файла листинга	12
4.12	Изучение файла листинга	13
4.13	Выбранные строки файла	13
4.14	Удаление выделенного операнда из кода	14
4.15	Получение файла листинга	14
4.16	Написание программы	15
4.17	Запуск файла и проверка	15

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 файлы листинга. 3.Задания для самостоятельной работы.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды переходов. Можно выделить 2 типа переходов:

- условный переход-выполнение или не выполнение перехода в определённую точку программы в зависимости от проверки условия.
- безусловный переход-выполнение передачи управления в определённую точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp`. Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команды вычитания.

Листинг (в аппарате NASM) - это один из выходных файлов, так как кроме строк самой программы он содержит дополнительную информацию

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программы лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm (рис. [4.1]).

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

Рис. 4.1: Создание файла

Ввожу файл lab7-1.asm текст программы из листинга 7.1. (рис. [4.2]).

```
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 _label2
10
11 _label1:
12 mov eax, msg1 ; Вывод на экран строки
13 call sprintf ; Сообщение №1
14
15 _label2:
16 mov eax, msg2 ; Вывод на экран строки
17 call sprintf ; Сообщение №2
18
19 _label3:
20 mov eax, msg3 ; Вывод на экран строки
21 call sprintf ; Сообщение №3
22
23 _end:
24 call quit
```

Рис. 4.2: Ввод текста

Создаю исполняемый файл и запускаю его (рис. [4.3]).

```
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0 $ gedit lab7-1.asm
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0 $ nasm -f elf lab7-1.asm
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0 $ ld -n elf_i386 -o lab7-1 lab7-1.o
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab0 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.3: Запуск исполняемого кода

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение №2', потом 'Сообщение №1' и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2 (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 _label2
10
11 _label1:
12 mov eax, msg1      ; Вывод на экран строки
13 call sprintfLF     ; Сообщение №1
14 jmp _end
15
16 _label2:
17 mov eax, msg2      ; Вывод на экран строки
18 call sprintfLF     ; Сообщение №2
19 jmp _label1
20
21 _label3:
22 mov eax, msg3      ; Вывод на экран строки
23 call sprintfLF     ; Сообщение №3
24
25 _end:
26 call quit
```

Рис. 4.4: Изменение текста программы

Создаю исполняемый файл и проверяю его.(рис. [4.5]).

```

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

```

Рис. 4.5: Создание исполняемого файла

Изменяю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1` (рис. [4.6]).

```

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
11 _label1:
12 mov eax, msg1      ; Вывод на экран строки
13 call sprintfLF     ; Сообщение №1
14 jmp _end
15
16 _label2:
17 mov eax, msg2      ; Вывод на экран строки
18 call sprintfLF     ; Сообщение №2
19 jmp _label1
20
21 _label3:
22 mov eax, msg3      ; Вывод на экран строки
23 call sprintfLF     ; Сообщение №3
24 jmp _label2
25
26 _end:
27 call quit

```

Рис. 4.6: изменение текста программы

Вывод программы будет таким образом:(рис. [4.7]).

```

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

```

Рис. 4.7: Вывод программы

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В, С. Значение для А и С задаются в программе, значение В вводится с клавиатуры. Создаю файл lab7-2.asm (рис. [4.8]).

```

dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$

```

Рис. 4.8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm (рис. [4.9]).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в max
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)

```

Рис. 4.9: Ввод текста программы

Создаю исполняемый файл и проверяю его (рис. [4.10]).

```

dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$

```

Рис. 4.10: Проверка работы файла

4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. [??]).

```

dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dkhuddheva@dkhuddheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ncdit lab7-2.lst

```

Рис. 4.11: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое.((рис. [4.12]).

```

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
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
14          <1> finished:-----
15 0000000B 29D8       <1>      sub     eax, ebx
16 0000000D 5B         <1>      pop     ebx
17 0000000E C3         <1>      ret
18
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
28 00000013 E8E8FFFF   <1>      call    slen
29
30 00000018 89C2       <1>      mov     edx, eax
31 0000001A 58         <1>      pop     eax
32
33 0000001B 89C1       <1>      mov     ecx, eax
34 0000001D BB01000000   <1>      mov     ebx, 1
35 00000022 B804000000   <1>      mov     eax, 4
36 00000027 CD80       <1>      int     80h

```

Рис. 4.12: Изучение файла листинга

В представленных трёх строчках содержатся следующие данные: (рис. [4.13]).

```

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

```

Рис. 4.13: Выбранные строки файла

“3”-номер строки кода, ;Функция вычисления длинны сообщения - комментарий к коду,не имеет адреса и машинного кода.

“4”-номер строки кода,“slen”- название функции, не имеет адреса и машинного кода.

“5”-номер строки кода,“00000000” -адрес строки,“53” - машинный код,“push

ebx”- исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываю файл с программой lab7-2.asm и выбранной мной инструкции с двумя операндами удаляю выделенный операнд.(рис. [4.14]).

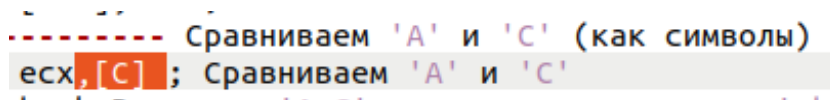


Рис. 4.14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла с листинга (рис. [4.15]).

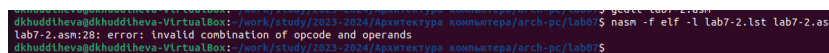


Рис. 4.15: Получение файла листинга

4.3 Задания для самостоятельной работы

1.Пишу программу нахождения наименьшей из 3 целочисленных переменных а,b,c.Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы №7.Мой вариант под номером 17. (рис. [4.17]).

```

1 %include 'in_out.asm'
2 section .data
3 msg db "Наименьшее число: ",0h
4 A dd '12'
5 B dd '26'
6 C dd '68'
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, msg
34 call sprint
35 mov eax,[min]
36 call iprintLF
37 call quit

```

Рис. 4.16: Написание программы

Создаю исполняемый файл и проверяю его работу (рис. [??]).

```

Наименьшее число: 26
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ nasm -f elf task1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ ld -m elf_i386 -o task1 task1.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ ./task1
Наименьшее число: 12

```

Рис. 4.17: Запуск файла и проверка

5 Выводы

Я изучила команды условного и безусловного переходов и приобрела навыки написания программ с использованием переходов, а также познакомилась с назначением и структурой файла листинга.

Список литературы

Лабораторная работа №7