

Отчёт по лабораторной работе №7

дисциплина: Архитектура компьютера

Гайдук Софья Сергеевна

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Задания для самостоятельной работы	14
4 Выводы	16

Список иллюстраций

2.1	image1	6
2.2	image2	7
2.3	image3	7
2.4	image4	8
2.5	image5	8
2.6	image6	9
2.7	image7	9
2.8	image8	9
2.9	image9	10
2.10	image10	10
2.11	image11	10
2.12	image12	11
2.13	image13	11
2.14	image14	12
2.15	image15	12
3.1	image16	14
3.2	image17	14
3.3	image18	15
3.4	image19	15
3.5	image20	15
3.6	image21	15

Список таблиц

1 Цель работы

Изучить команды условного и безусловного переходов. Освоить навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm. Проверим его наличие (рис. 2.1).

```
ssgayjduk1@dk6n14 ~/work $ cd ~
ssgayjduk1@dk6n14 ~ $ mkdir ~/work/arch-pc/lab07
ssgayjduk1@dk6n14 ~ $ cd ~/work/arch-pc/lab07
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ touch lab7-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ls
lab7-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ █
```

Рисунок 2.1: image1

Введем в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2.2).

```
.../.dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab07/lab7-1.asm
%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:
_start:
jmp _label2
_label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 1'
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'
_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 3'
_end:
    call quit ; вызов подпрограммы завершения
```

Рисунок 2.2: image2

Создадим исполняемый файл и запустим его (рис. 2.3).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $
```

Рисунок 2.3: image3

Изменим текст программы в соответствии с листингом 7.2 (рис. 2.4).

```
GLOBAL _start
_start:
    jmp _label2
_label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 1'
    jmp _end
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'
    jmp _label1
_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 3'
_end:
```

Рисунок 2.4: image4

Создадим исполняемый файл и проверим его работу (рис. 2.5).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $
```

Рисунок 2.5: image5

Изменим текст программы добавив и изменив инструкции jmp, чтобы вывод программы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1 (рис. 2.6).

```

.../.dk.sc1.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab07/lab7-1.asm Изменение
GLOBAL _start
_start:
    jmp label3
label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 1'
    jmp _end
label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'
    jmp _label1
label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 3'
    jmp _label2
_end:

```

Справка ^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выполнить M-U Отмена
Выход ^X ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор

Рисунок 2.6: image6

Создадим исполняемый файл и проверим его работу (рис. 2.7).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $

```

Рисунок 2.7: image7

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Введем в lab7-2.asm текст из листинга 7.3 (рис. 2.8).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ touch lab7-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $

```

Рисунок 2.8: image8

```

.../.dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab07/lab7-2.asm Изменение
%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
resB 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
inc

```

Справка Запись Поиск Вырезать Выполнить Отмена
Выход ЧитФайл Замена Вставить Позиция Повтор

Рисунок 2.9: image9

Создадим исполняемый файл и проверим его работу для разных значений B (рис. 2.10).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1000
Наибольшее число: 1000
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 30
Наибольшее число: 50
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ 

```

Рисунок 2.10: image10

Создадим файл листинга для программы из файла lab7-2.asm, указав ключ -l и задав имя файла листинга lab7-2.lst (рис. 2.11).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ 

```

Рисунок 2.11: image11

Откроем файл листинга lab7-2.lst с помощью текстового редактора mcedit (рис. 2.12).

The screenshot shows the mcedit text editor with the file 'lab7-2.lst' open. The code is assembly language with comments. The assembly code includes instructions like push, mov, inc, jmp, cmp, jz, sub, pop, ret, and call. There are also comments for 'slen', 'nextchar', 'finished', 'sprint', and 'message'. The file path is shown as '1/225] *(0 /14458b) 0032 0x020 [*]' at the top. The menu bar at the bottom has items 1-10: Помощь, Сохран, Блок, Замена, Копия, Переть, Поиск, Удаить, МенюМС, Выход.

Рисунок 2.12: image12

Объясним содержимое трёх строк (17, 18, 19) файла листинга lab7-2.lst (рис. 2.13).

The screenshot shows the mcedit text editor with the file 'lab7-2.lst' open, focusing on lines 16 through 20. The assembly code includes instructions like mov, call, and a comment. The comments explain the input ('В') and the conversion of 'В' to ASCII. The file path is shown as '1/225] *(0 /14458b) 0032 0x020 [*]' at the top.

Рисунок 2.13: image13

17 - это номер строки

000000F2 - это адрес (смещение машинного кода от начала текущего сегмента)

В9[0A000000] - машинный код (ассемблированную исходную строку в виде шестнадцатеричной последовательности)

mov ecx, В - исходный текст программы

mov - это команда пересылки данных на языке ассемблера.

ecx - это регистр (сверхбыстрая оперативная память небольшого объёма, для хранения переменной (результата), максимальная длина которой – 32 бита). В него мы

запишем адрес В.

„В“ - это введенная нами переменная.

18 - это номер строки

000000F7 - это адрес (смещение машинного кода от начала текущего сегмента)

BA0A000000 - машинный код (ассемблированную исходную строку в виде шестнадцатеричной последовательности)

mov edx, 10 - исходный текст программы

mov - это команда пересылки данных на языке ассемблера.

edx - это регистр (сверхбыстрая оперативная память небольшого объёма, для хранения переменной (результата), максимальная длина которой – 32 бита). В него мы запишем длину В.

10 - это длина переменой В.

19 - это номер строки

000000FC - это адрес (смещение машинного кода от начала текущего сегмента)

E842FFFFFF - машинный код (ассемблированную исходную строку в виде шестнадцатеричной последовательности)

call sread - это вызов подпрограммы из внешнего файла для считывания ввода с клавиатуры.

Откроем файл с программой lab7-2.asm и в инструкции с двумя operandами удалим один operand (удалим строку mov edx, 10). Выполним трансляцию с получением файла листинга (рис. 2.14).

```
; ----- Ввод 'В'
mov ecx, B
[REDACTED]
call sread
. ----- Преобразование 'R' из символов в число
```

Рисунок 2.14: image14

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ [REDACTED]
```

Рисунок 2.15: image15

В листинге изменится машинный код так, как мы удалили одну строку, следовательно, код программы меняется.

3 Задания для самостоятельной работы

1. Создадим файл lab7-3.asm. Напишем программу нахождения наименьшей из 3 целочисленных переменных A, B, C. Значения переменных будут равны A=79, B=83, C=41 из таблицы 7.5 вариант 6, полученным при выполнении лабораторной работы №6. Создадим исполняемый файл и проверим его работу (рис. 3.1).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ touch lab7-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ 
```

Рисунок 3.1: image16

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-3
Введите B: 83
Наименьшее число: 41
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ 
```

Рисунок 3.2: image17

2. Создадим файл lab7-4.asm. Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ и значения x и a выберем из таблицы 7.6 вариант 6, полученным при выполнении лабораторной работы №6. Создадим исполняемый файл и проверим его работу. При $x=a$ вывести $x+a$, при $x \neq a$ вывести $5x$ (рис. 3.3).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ touch lab7-4.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $
```

Рисунок 3.3: image18

$$6 \quad \begin{cases} x + a, & x = a \\ 5x, & x \neq a \end{cases} \quad (2;2) \quad (2;1)$$

Рисунок 3.4: image19

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-4  
Введите а: 2  
Введите х: 2  
Итого: 4  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $ ./lab7-4  
Введите а: 1  
Введите х: 2  
Итого: 10  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab07 $
```

Рисунок 3.5: image20

Отправим файлы на Github (рис. 3.6).

```
07/report $ git add .  
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab  
07/report $ git commit -am 'feat(aim): add lab-7'  
[master fd836f1] feat(aim): add lab-7  
 2 files changed, 0 insertions(+), 0 deletions(-)  
 create mode 100644 labs/lab07/report/arch-pc--lab07--report.docx  
 create mode 100644 labs/lab07/report/arch-pc--lab07--report.pdf  
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab  
07/report $ git push  
Перечисление объектов: 11, готово.  
Подсчет объектов: 100% (11/11), готово.  
При сжатии изменений используется до 6 потоков  
Сжатие объектов: 100% (7/7), готово.  
Запись объектов: 100% (7/7), 1.34 МиБ | 9.24 МиБ/с, готово.  
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.  
To github.com:SofiaGayduk/study_2025-2026_arh-pc.git  
 defce48..fd836f1 master -> master  
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab  
07/report $
```

Рисунок 3.6: image21

4 Выводы

Мы изучили команды условного и безусловного переходов. Научились писать программы с использованием переходов. Познакомились с назначением и структурой файла листинга.