

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Мартынова Милана Александровна

Группа: НКАбд-04-25

МОСКВА

2025 г.

Содержание

1. Цель работы.....	4
2. Порядок выполнения лабораторной работы.....	4
2.1 Реализация переходов в NASM.....	4
2.2 Изучение структуры файлы листинга.....	7
2.3 Задания для самостоятельной работы.....	8
3. Выводы.....	9

Список иллюстраций

1. Создание каталога и файла для программы.....	4
2. Сохранение программы.....	4
3. Запуск программы.....	4
4. Изменение программы.....	5
5. Запуск измененной программы.....	5
6. Повторное изменение программы.....	5
7. Проверка изменений.....	6
8. Сохранение новой программы.....	6
9. Проверка программы листинга.....	6
10. Создание файла листинга.....	7
11. Проверка файла листинга.....	7
12. Удаление операнда из программы.....	7
13. Просмотр ошибки в файле листинга.....	8
14. Первая программа самостоятельной работы.....	8
15. Проверка работы первой программы.....	8
16. Вторая программа самостоятельной работы.....	9
17. Проверка работы второй программы.....	9

1. Цель работы

Освоение применения команд условных и безусловных переходов при программировании.
Формирование практических умений разработки программ с использованием механизма
переходов. Ознакомление с назначением, форматом и содержанием файла листинга.

2. Порядок выполнения лабораторной работы

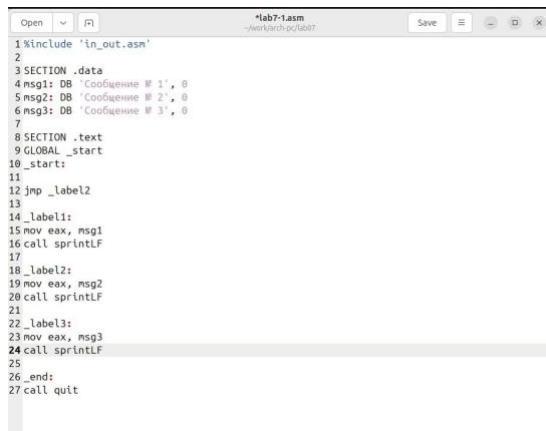
2.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7(рис. 1).

```
mamartynova@VirtualBox:~$ mkdir ~/work/arch-pc/lab07
mamartynova@VirtualBox:~$ cd ~/work/arch-pc/Lab07
mamartynova@VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 1: Создание каталога и файла для программы

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



```
Open ▾ F Save
lab7-1.asm
1;include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10_start:
11 jmp _label2
12
13 _label1:
14 mov eax, msg1
15 call sprintF
16
17 _label2:
18 mov eax, msg2
19 call sprintF
20
21 _label3:
22 mov eax, msg3
23 call sprintF
24
25
26_end:
27 call quit
```

Рис 2: Сохранение программы

При запуске программы убедилась, что неусловный переход действительно изменяет порядок выполнения инструкций(рис. 3).

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

Рис. 3: Запуск программы

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

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1
16 call sprintLF
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintLF
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintLF
27
28 _end:
29 call quit

```

Рис. 4: Изменение программы

Запускаю программу и проверяю верность применённых изменений(рис. 5).

```

mamartynova@VirtualBox:~/work/arch-pc/lab07$ gedit lab7-1.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 5: Запуск измененной программы

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

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1
16 call sprintLF
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintLF
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintLF
27 jmp _label2
28
29 _end:
30 call quit

```

Рис. 6: Повторное изменение программы

Изменение выполнено корректно, программа выводит сообщения в нужном мне порядке(рис. 7).

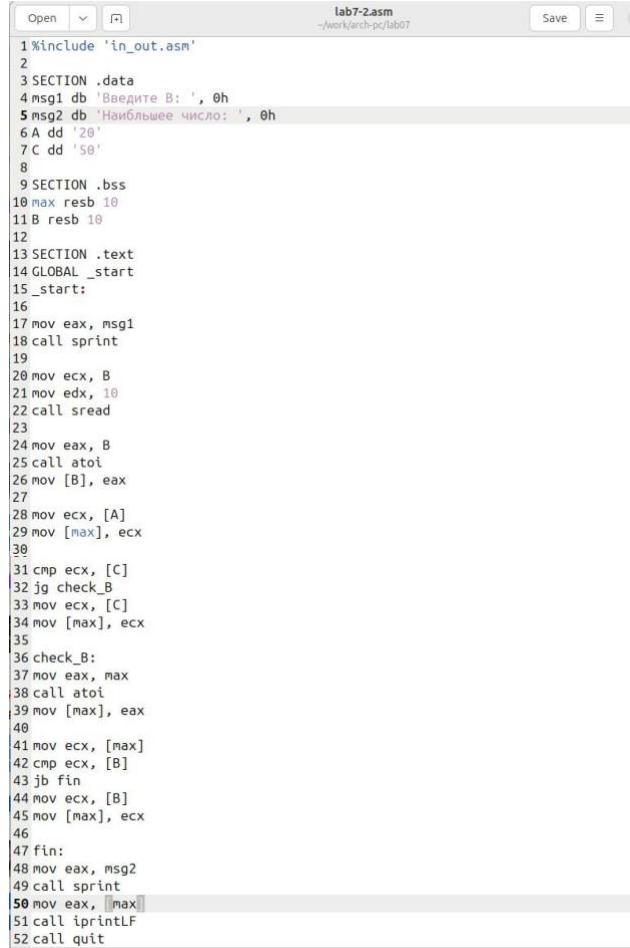
```

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

```

Рис. 7: Проверка изменений

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



```

Open Save lab7-2.asm -/work/arch-pc/lab07
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наибольшее число: ', 0h
6 A dd '20'
7 C dd '50'
8
9 SECTION .bss
10 max resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [max], ecx
35
36 check_B:
37 mov eax, max
38 call atoi
39 mov [max], eax
40
41 mov ecx, [max]
42 cmp ecx, [B]
43 jb fin
44 mov ecx, [B]
45 mov [max], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [max]
51 call iprintf
52 call quit

```

Рис. 8: Сохранение новой программы

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

```

mamartynova@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 25
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 50
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 10
mamartynova@VirtualBox:~/work/arch-pc/lab07$ 

```

Рис. 9: Проверка программы листинга

2.2 Изучение структуры файлы листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора gedit (рис. 10, 11).

```
mamartynova@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm  
mamartynova@VirtualBox:~/work/arch-pc/lab07$ gedit lab7-2.lst
```

Рис. 10: Создание файла листинга

```
Open ▾ ⌂ lab7-2.lst -/wolk/arch-pc/lab07 Save ⌄ ⌅ ⌇ Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 INS
1 1 %include 'in_out.asm'
2 1 <1> ;----- slen -----
3 2 <1> ; Функция вычисления длины сообщения
4 3 <1> slen:
5 4 00000000 53 <1> push ebx
6 5 00000001 89C3 <1> mov ebx, eax
7 6 <1>
8 7 <1> nextchar:
9 8 00000003 803B00 <1> cmp byte [eax], 0
10 9 00000006 7403 <1> jz finished
11 10 00000008 40 <1> inc eax
12 11 00000009 EBF8 <1> jmp nextchar
13 12 <1>
14 13 <1> finished:
15 14 0000000B 29D8 <1> sub eax, ebx
16 15 0000000D 5B <1> pop ebx
17 16 0000000E C3 <1> ret
18 17 <1>
19 18 <1>
20 19 <1> ;----- sprint

21 20 <1> ; Функция печати сообщения
22 21 <1> ; входные данные: mov eax,<message>
23 22 <1> sprint:
24 23 0000000F 52 <1> push edx
25 24 00000010 51 <1> push ecx
26 25 00000011 53 <1> push ebx
27 26 00000012 50 <1> push eax
28 27 00000013 EBEBFFFFFF <1> call slen
29 28 <1>
30 29 00000012 80C7 <1> mov adv eax
```

Рис. 11: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 12).

Рис.12: 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. 13).

```

194 19
195 20 000000F2 B9[0A000000]      mov ecx, B
196 21 000000F7 BA0A000000       mov edx, 10
197 22 000000FC E842FFFFFF      call sread
198 23
199 24 00000101 B8[0A000000]      mov eax, B
200 25 00000106 E891FFFFFF      call atoi
201 26 0000010B A3[0A000000]      mov [B], eax
202 27
203 28 00000110 880D[36000000]    mov ecx, [A]
204 29 00000116 890D[00000000]    mov [max], ecx
205 30
206 31 0000011C 3B0D[3A000000]    cmp ecx, [C]
207 32 00000122 7F0C             jg check_B
208 33 00000124 880D[3A000000]    mov ecx, [C]
209 34 0000012A 890D[00000000]    mov [max], ecx
210 35
211 36
212 37
213 37 ***** error: invalid combination of opcode and
214 38 00000130 E867FFFFFF      call atoi
215 39 00000135 A3[00000000]      mov [max], eax
216 40
217 41 0000013A 880D[00000000]    mov ecx, [max]

```

Рис. 13: Просмотр ошибки в файле листинга

2.3 Задания для самостоятельной работы

Возвращаю operand к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 14).

```

1%include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '20'
7 C dd '50'
8
9 SECTION .bss
10 max resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [min], ecx
35
36 check_B:
37 mov eax, min
38 call atoi
39 mov [min], eax
40
41 mov ecx, [min]
42 cmp ecx, [B]
43 jb fin
44 mov ecx, [B]
45 mov [min], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [min]
51 call iprintF
52 call quit

```

Рис. 14: Первая программа самостоятельной работы

Проверяю корректность написания первой программы (рис. 15).

```

mamartynova@VirtualBox:~/work/arch-pc/lab07$ gedit lab7-2.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наименьшее число: 5

```

Рис. 15: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных а и х (рис. 16).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, 80
25 call sread
26 mov eax, a
27 call atoi
28 mov est, eax
29
30 cmp edi, esi
31 jle add_values
32 mov eax, esi
33 jmp print_result
34
35 add_values:
36 mov eax, edi
37 add eax, esi
38
39 print_result:
40 mov edi, eax
41 mov eax, res
42 call sprint
43 mov eax, edi
44 call iprintLF
45 call exit
```

Рис. 16: Вторая программа самостоятельной работы

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 17).

```
mamartynova@VirtualBox:~/work/arch-pc/lab07$ touch lab7-3.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ gedit lab7-3.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ gedit lab7-3.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 3
Введите значение переменной a: 4
Результат: 9
mamartynova@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 4
Результат: 5
```

Рис. 17: Проверка работы второй программы

3. Выводы

В рамках лабораторной работы успешно освоено применение команд условных и безусловных переходов для управления логикой выполнения программ. Приобретен практический опыт написания и отладки кода с использованием ветвлений и циклов. Также проведен анализ файла листинга, что позволило понять взаимосвязь между исходным кодом и результирующим машинным кодом, а также использовать листинг для поиска и исправления ошибок.

Список литературы

1. GDB: The GNU Project Debugger. — URL:<https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL:
<https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Кулjas О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL:
<https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL:
http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).