

Отчёт по лабораторной работе №7

Архитектура компьютера

Алехин Давид Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создание lab07/lab7-1.asm	8
4.2	in_out.asm	8
4.3	Программа 7.1	9
4.4	./lab7-1	9
4.5	lab7-1.2.asm	10
4.6	Запуск lab7-1.2.asm	10
4.7	Откорректированный lab7-1.2.asm	11
4.8	Запуск откорректированного lab7-1.2.asm	11
4.9	lab7-2.asm	12
4.10	Запуск lab7-2.asm	12
4.11	lab7-2.lst	13
4.12	Расшифровка 3х строк листинга	13
4.13	mov eax	14
4.14	Ошибка при создании	14
4.15	Ошибка в листинге	14
4.16	Код 1 задания	15
4.17	./lab7-1s.asm	15
4.18	Код 2 задания	16
4.19	./lab7-2s.asm	17

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

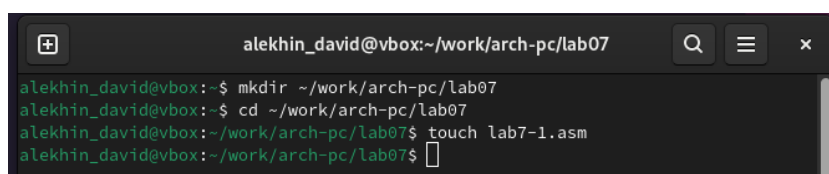
1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

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.

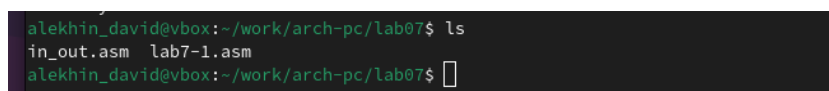
4 Выполнение лабораторной работы

Создаю папку для лабораторной работы №7, в ней создаб файл lab7-1.asm (рис. 4.1). Переношу в эту папку файл in_out.asm. (рис. 4.2).



```
alekhin_david@vbox:~/work/arch-pc/lab07
alekhin_david@vbox:~$ mkdir ~/work/arch-pc/lab07
alekhin_david@vbox:~$ cd ~/work/arch-pc/lab07
alekhin_david@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
alekhin_david@vbox:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание lab07/lab7-1.asm



```
alekhin_david@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm
alekhin_david@vbox:~/work/arch-pc/lab07$
```

Рис. 4.2: in_out.asm

Ввожу в файл lab7-1.asm текст программы 7.1. Программа с использованием инструкции jmp. (рис. 4.3).


```

alekhin_david@vbox:~/work/arch-pc/lab07$ cat 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:
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. 4.3: Программа 7.1

Компелирую файл и запускаю его.(рис. 4.4).

```

alekhin_david@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab5-1 lab5-1.o
ld: невозможно найти lab5-1.o: Нет такого файла или каталога
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
alekhin_david@vbox:~/work/arch-pc/lab07$ █

```

Рис. 4.4: ./lab7-1

Создаю файл lab7-1.2.asm с текстом команды 7.2. Программа с использованием инструкции jmp. (рис. 4.5). Компелирую и запускаю его. (рис. 4.6).

```

lab7-1.2.asm      [-M--] 31 L:[ 1+11 12/ 22] *(358 / 670b) 0010 0x
#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 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:
call quit ; вызов подпрограммы завершения

```

Рис. 4.5: lab7-1.2.asm

```

alekhin_david@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.2.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1.2 lab7-1.2.o
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-1.2
Сообщение № 2
Сообщение № 1
alekhin_david@vbox:~/work/arch-pc/lab07$ 

```

Рис. 4.6: Запуск lab7-1.2.asm

Корректирую текст команды чтобы она выводила Сообщение №1, Сообщение №2, Сообщение №3 в обратном порядке. (рис. 4.7). Компилирую и запускаю исправленный файл. (рис. 4.8).

```

lab7-1.2.asm [----] 11 L: [ 1+20 21/ 23] *(607 / 682b) 0010 0x00A
#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. 4.7: Откорректированный lab7-1.2.asm

```

alekhin_david@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.2.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1.2 lab7-1.2.o
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-1.2
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 4.8: Запуск откорректированного lab7-1.2.asm

Слздаю файл lab7-2.asm с текстом команды 7.3. Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. (рис. 4.9).

```

alekhin_david@vbox:~/work/arch-pc/lab07$ cat lab7-2.asm
#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,B
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 iprintf ; Вывод 'max(A,B,C)'
alekhin_david@vbox:~/work/arch-pc/lab07$

```

Рис. 4.9: lab7-2.asm

Компелирую и запускаю lab7-2.asm. (рис. 4.10).

```

alekhin_david@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 111
Наибольшее число: 111
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 12

```

Рис. 4.10: Запуск lab7-2.asm

Создаю и открываю листинг lab7-2.asm. (рис. 4.11).

```

lab7-2.lst      [----]  0 L:  1+ 0  1/225) +(0  /14458b) 0032 0x020  [*)(X)
1      1      %include 'in_out.asm'
2      2      ;----- slen -----
3      3      <1> ; Функция вычисления длины сообщения
4      4      00000000 53      <1> push ebx.....
5      5      00000001 89C3    <1> mov ebx, eax.....
6      6      00000000 00      <1> .....
7      7      00000003 803800    <1> nextchar: .....
8      8      00000006 7403    <1> cmp byte [eax], 0...
9      9      00000008 40      <1> jz finished.....
10     10     00000008 40      <1> inc eax.....
11     11     00000009 EBF8    <1> jmp nextchar.....
12     12     00000000 00      <1> .....
13     13     00000008 20D8    <1> finished: .....
14     14     00000000 5B      <1> sub eax, ebx.....
15     15     00000000 5B      <1> pop ebx.....
16     16     0000000E C3      <1> ret.....
17     17     00000000 00      <1> .....
18     18     00000000 00      <1> .....
19     19     ;----- sprint -----
20     20     <1> ; Функция печати сообщения
21     21     <1> ; входные данные: mov eax, <message>
22     22     0000000F 52      <1> sprint: .....
23     23     00000010 51      <1> push edx.....
24     24     00000010 51      <1> push ecx.....
25     25     00000011 53      <1> push ebx.....
26     26     00000012 50      <1> push eax.....
27     27     00000013 EB8FFFFFFF <1> call slen.....
28     28     00000018 89C2    <1> .....
29     29     0000001A 5B      <1> mov edx, eax.....
30     30     0000001A 5B      <1> pop eax.....
31     31     0000001B 89C1    <1> .....
32     32     0000001B 89C1    <1> mov ecx, eax.....
33     33     0000001D B801000000 <1> mov ebx, 1.....
34     34     00000022 B804000000 <1> mov eax, 4.....
35     35     00000027 CD80    <1> int 80h.....
36     36     00000029 5B      <1> .....
37     37     00000029 5B      <1> pop ebx.....
38     38     0000002A 59      <1> pop ecx.....
39     39     0000002B 5A      <1> pop edx.....
40     40     0000002C C3      <1> ret.....
41     41     00000000 00      <1> .....
42     42     00000000 00      <1> .....

```

Рис. 4.11: lab7-2.lst

Расшифровка 3х строк листинга: 1)Строка 5: Эта строка находится на 5 месте, ее адрес “00000035”, Машинный код - [32300000], A dd ‘20’ - исходный текст программы, присваивающий переменной A значение 20. 2)Строка 6: Эта строка находится на 6 месте, ее адрес “00000039”, Машинный код - [35300000], A dd ‘50’ - исходный текст программы, присваивающий переменной B значение 50. 3)Строка 8 : Эта строка находится на 8 месте, ее адрес “00000000”, Машинный код - res Ah, max resb 10 - исходный текст программы, означающий, что максимальное резервирование может быть 10 байт. (рис. 4.12).

```

5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7                          section .bss
8 00000000 <res Ah>      max resb 10

```

Рис. 4.12: Расшифровка 3х строк листинга

У даляю в строке mov eax, max операнду max.(рис. 4.13).



Рис. 4.13: mov eax

В результате получаю ошибку при создании asm файла и lst файла (рис. 4.14), а в фай с листингом получаю ошибку в 34 строке.(рис. 4.15).

```
alekhin_david@vbox: ~/work/arch-pc/lab07$ nano lab7-2.asm
alekhin_david@vbox: ~/work/arch-pc/lab07$ nano -f elf lab7-2.asm
Указанный rcfile не существует
alekhin_david@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
alekhin_david@vbox: ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
```

Рис. 4.14: Ошибка при создании

```
34                                     mov eax
34      *****                      error: invalid combination of opcode and operands
```

Рис. 4.15: Ошибка в листинге

Приступаю к самостоятельной работе (1 вариант). Создаю файл lab7-1s.asm и пишу код для первого задания. (рис. 4.16).

```

#include 'in_out.asm'

SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
otv DB 'Наименьшее число: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax, A
call atoi
mov [A],eax

xor eax,eax

```

Рис. 4.16: Код 1 задания

Компеллирую и запускаю lab7-1s.asm, проверяю работоспособность.(рис. 4.17).

```

alekhin_david@vbox: ~/work/arch-pc/lab07$ touch lab7-1s.asm
alekhin_david@vbox: ~/work/arch-pc/lab07$ mcedit lab7-1s.asm

alekhin_david@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-1s.asm
alekhin_david@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1s lab7-1s.o
alekhin_david@vbox: ~/work/arch-pc/lab07$ ./lab7-1s
Введите число A: 17
Введите число B: 23
Введите число C: 45
Наименьшее число: 17
alekhin_david@vbox: ~/work/arch-pc/lab07$ 

```

Рис. 4.17: ./lab7-1s.asm

Создаю файл lab7-2s.asm и пишу код для второго задания. (рис. 4.18).

```
/home/alekhin_david/work/arch-pc/lab07/lab7-2s.asm
%include 'in_out.asm'

SECTION .data
prim1 DB '2a-x ,x<a' ,0
prim2 DB '8, x=>a',0
X1 DB 'Введите значение X:',0
A1 DB 'Введите значение a:',0
otv DB 'Ответ: ',0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax,prim1
call sprintLF
mov eax,prim2
call sprintLF

mov eax,X1
call sprint

mov ecx,X
mov edx,10
call sread

1Помощь 2Разверн 3Выход 4Нех 5Перейти
```

Рис. 4.18: Код 2 задания

Компеллирую и запускаю lab7-2s.asm, проверяю работоспособность. (рис. 4.19).


```
alekhin_david@vbox:~/work/arch-pc/lab07$ touch lab7-2s.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ mcedit lab7-2s.asm

alekhin_david@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2s.asm
alekhin_david@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2s lab7-2s.o
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-2s
2a-x ,x<a
8, x=>a
Введите значение X:1
Введите значение a:2
Ответ: 3
alekhin_david@vbox:~/work/arch-pc/lab07$ ./lab7-2s
2a-x ,x<a
8, x=>a
Введите значение X:2
Введите значение a:1
Ответ: 8
```

Рис. 4.19: ./lab7-2s.asm

5 Выводы

Сделав лабораторную работу я выполнил её главную цель: “Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.”

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 с. — (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 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).