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: Безходарнова А.В

Группа: НКАбд-01-25

МОСКВА

2025 г.

СОДЕРЖАНИЕ

1.Цель работы 3

2. Выполнение лабораторной работы 4

2.1 Реализация переходов в NASM 4
2.2 Изучение структуры файла листинга 10
3. Задание для самостоятельной работы 14

4. Выводы 17

1.Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2. Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создаю каталог для программам лабораторной работы №7, перейдя в него создаю файл lab7-1.asm:.(рис.2.1.1)

Рис. 2.1.1. Создание файла

В созданном файле ввожу программу из листинга. (рис.2.1.3)

Open • lab07-1.asm
~/work/arch-pc/lab07

```
%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'
_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
```

Рис. 2.1.2 Загрузка текста из листинга

Далее запускаю исходный код. (рис.2.1.3)

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf lab07-1.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ld -m elf_386i -o lab07-1 lab07-1.o
ld: unrecognised emulation mode: elf_386i
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i386pep i386pe elf64bpf
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ./lab07-1
Сообщение № 2
Сообщение № 3
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$
```

Рис. 2.1.3 Запуск программы

Убеждаюсь в том, что, использование инструкции jmp меняет порядок исполнения инструкций.

Далее я изменяю программу для получения другого результата. (Рис.2.1.4) и (рис.2.1.5)

```
%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.2.1.4 Изменение текста

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf lab07-1.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ./lab07-1
Сообщение № 2
Сообщение № 1
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$
```

Рис.2.1.5 Запуск программы

Теперь изменяю файл так, чтобы сообщения выводились в обратном порядке. (рис.2.1.6) и (рис.2.1.7)

Open • lab07-1.asm
~/work/arch-pc/lab07

```
%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 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:
call quit ; вызов подпрограммы завершения
```

Рис.2.1.6 Изменение текста

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf lab07-1.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab07-1 lab07-1.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ./lab07-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$
```

Рис.2.1.7 Вывод результата

Программа ввела сообщения в обратном порядке, как и было нужно.

Далее создаю новый файл и вставляю в него код из другого листинга. (рис.2.1.8)

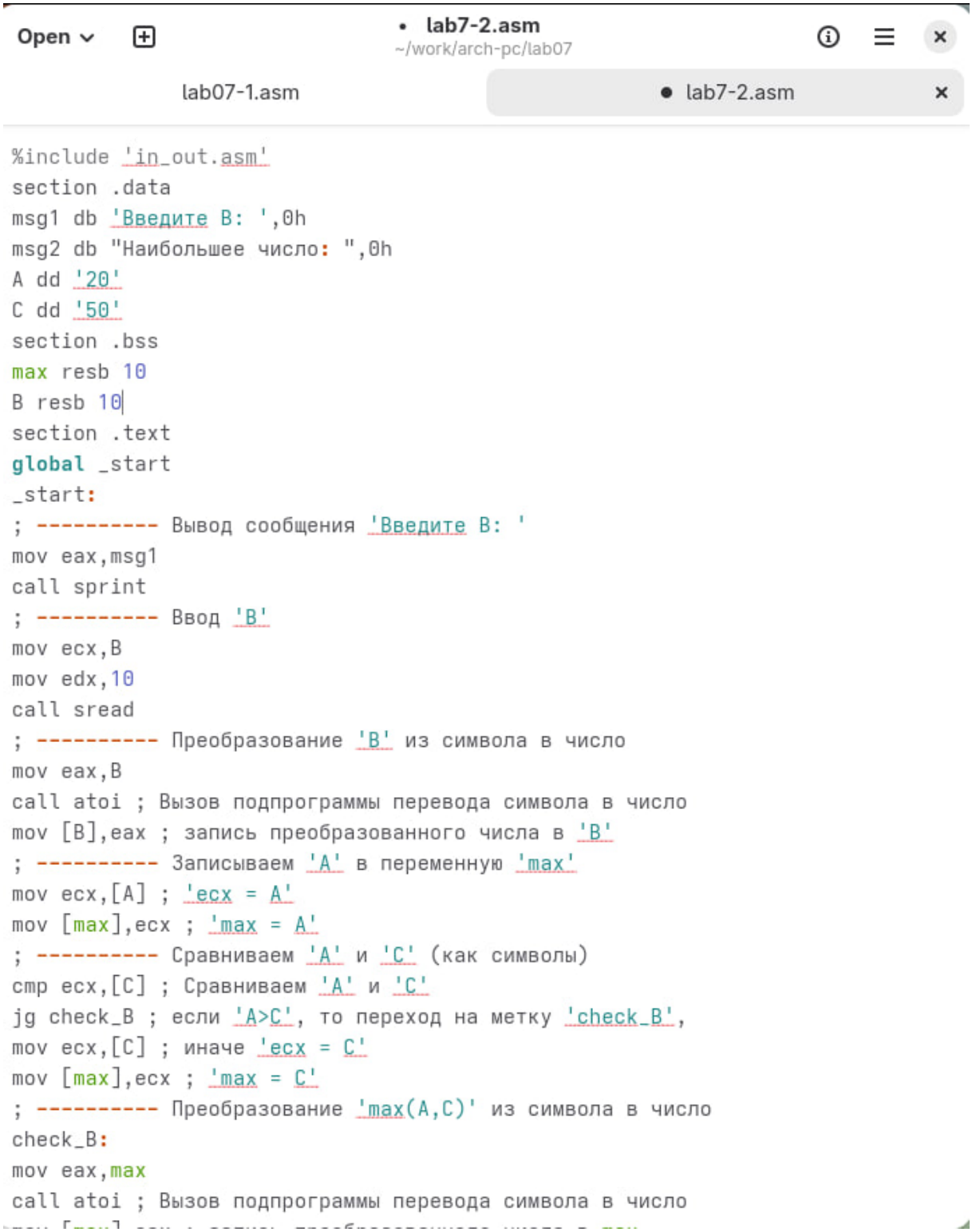

Рис.2.1.8 Код из листинга

Запускаю программу и получаю результат. (рис.2.1.9)

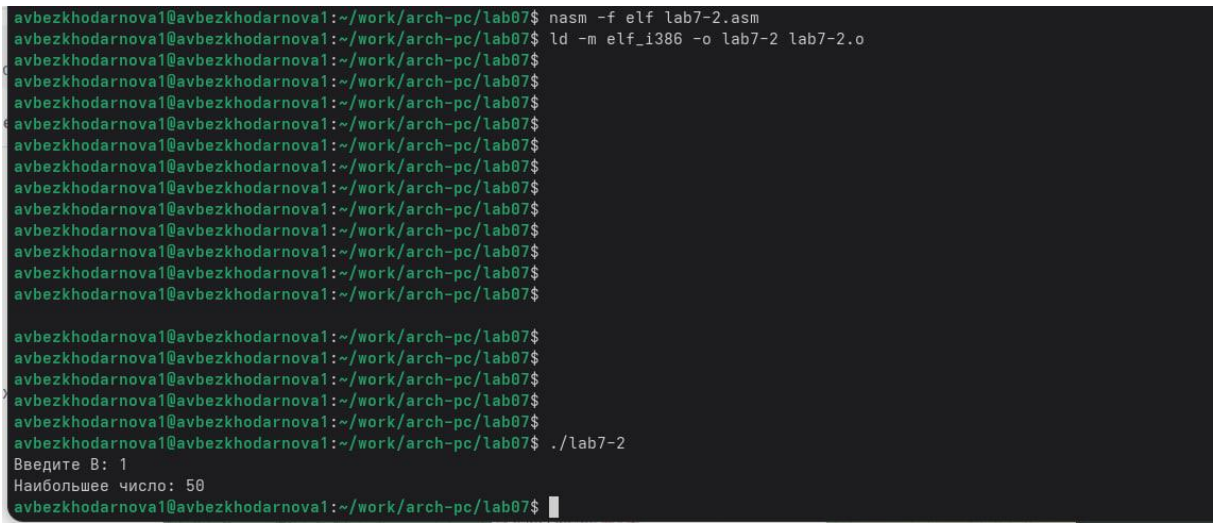

Рис.2.1.9 запуск программы

Проверяю, чтобы программа работала верно.

2.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его (рис. 2.2.1)

```
1                                       %include 'in_out.asm'
1                                   <1> ;---------------   slen  -------------------
2                                   <1> ; Функция вычисления длины сообщения
3                                   <1> slen:
4 00000000 53                       <1>     push    ebx
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 0000000E C3                      <1>     ret
17                                  <1>
18                                  <1>
19                                  <1> ;---------------  sprint  -------------------
20                                  <1> ; Функция печати сообщения
21                                  <1> ; входные данные: mov eax,<message>
22                                  <1> sprint:
23 0000000F 52                      <1>     push    edx
24 00000010 51                      <1>     push    ecx
25 00000011 53                      <1>     push    ebx
26 00000012 50                      <1>     push    eax
27 00000013 E8E8FFFFFF              <1>     call    slen
28                                  <1>
29 00000018 89C2                    <1>     mov     edx, eax
30 0000001A 58                      <1>     pop     eax
31                                  <1>
32 0000001B 89C1                    <1>     mov     ecx, eax
33 0000001D BB01000000              <1>     mov     ebx, 1
34 00000022 B804000000              <1>     mov     eax, 4
35 00000027 CD80                    <1>     int     80h
36                                  <1>
37 00000029 5B                      <1>     pop     ebx
38 0000002A 59                      <1>     pop     ecx
39 0000002B 5A                      <1>     pop     edx
40 0000002C C3                      <1>     ret
41                                  <1>
```

Рис.2.2.1 Открытие файла.

В листинге 4 строка – это порядковый номер строки. Нули отвечают за смещение в байтах от начала, а 53 означает команду puch ebx. Далее 6

строка идет пустой, это из-за того, что там в исходном файле там находится пустая строка или строка с комментарием. Потом, к примеру, 8 строка. Цифра 8, так же как и цифра 4 означает порядковый номер строки. Цифра 3 после нулей – насколько смещена строка от начала, а остальные цифры обозначают команды.

Удаляю один из операндов. (Рис.2.2.2) Потом я запускаю программу (Рис.2.2.3) и нахожу ошибку. (Рис.2.2.4)

Open • lab7-2.asm ~/work/arch-pc/lab07

lab07-1.asm | lab7-2.asm | lab7-2.lst

```
mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi ;
mov [B],eax ;

mov ecx ;
mov [max],ecx ;

cmp ecx,[C] ;
jg check_B ;
mov ecx,[C] ;
mov [max],ecx ;

check_B:
mov eax,max
call atoi ;
mov [max],eax ;

mov ecx,[max]
cmp ecx,[B] ;
jg fin ;
mov ecx,[B] ;
mov [max],ecx

fin:
mov eax, msg2
call sprint ;
mov eax,[max]
call iprintLF ;
call quit ; Выход
```

Рис. 2.2.2 Изменение кода.

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:25: error: invalid combination of opcode and operands
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$
```

Рис. 2.2.3 Запуск программы.

```
24
25                                  mov ecx ;
25          ******************       error: invalid combination of opcode and operands
26 00000110 890D[00000000]          mov [max],ecx ;
27
28 00000116 3B0D[39000000]          cmp ecx,[C] ;
29 0000011C 7F0C                    jg check_B ;
30 0000011E 8B0D[39000000]          mov ecx,[C] ;
```

Рис.2.2.4 Полученная ошибка

В этом случае создается только файл .lst, а в сам файл листинга добавляется сообщение об ошибке.

3. Задание для самостоятельной работы

Из предложенных вариантов, я выбираю 5, где a,b,c соотвественно равны 54,62,87.

Создаю новый файл и пишу в нем программу для вывода наименьшего значения. (рис.3.1) и (рис.3.2)

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '54'
C dd '87'
SECTION .bss
min resb 10
B resb 10
SECTION .text
GLOBAL _start
_start:
mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jl check_B    ;
mov ecx, [C]  ;
mov [min], ecx ; 'min = C'


check_B:
mov eax, min
call atoi
```

Рис.3.1 Редактирование файла.

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ touch lab7-3.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 62
Наименьшее число: 54
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab07$
```

Рис.3.2 Запуск программы.

Для следующего задания я выбрала первую функцию. Также пишу программу и запускаю ее. (рис.3.3) и (рис.3.4)

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, esi
jl less_than
mov eax, 8
jmp print_result

less_than:
mov eax, esi
add eax, eax
sub eax, edi

print_result:
mov edi, eax
mov eax, res
```

Рис.3.3 Изменение кода.

Рис.3.4 Вывод результата

4. Выводы

В ходе выполнения данной лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием перходов, познакомилась с назначением и структурой файлов листинга.