

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера
Хан Георгий Игоревич

Оглавление

1 Цель работы	2
2 Задание	2
3 Теоретическое введение	2
4 Выполнение лабораторной работы	3
4.1 Реализация переходов в NASM	3
4.2 Изучение структуры файла листинга.....	8
4.3 Задания для самостоятельной работы	9
5 Выводы.....	13
Список литературы.....	13

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 файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

3 Теоретическое введение

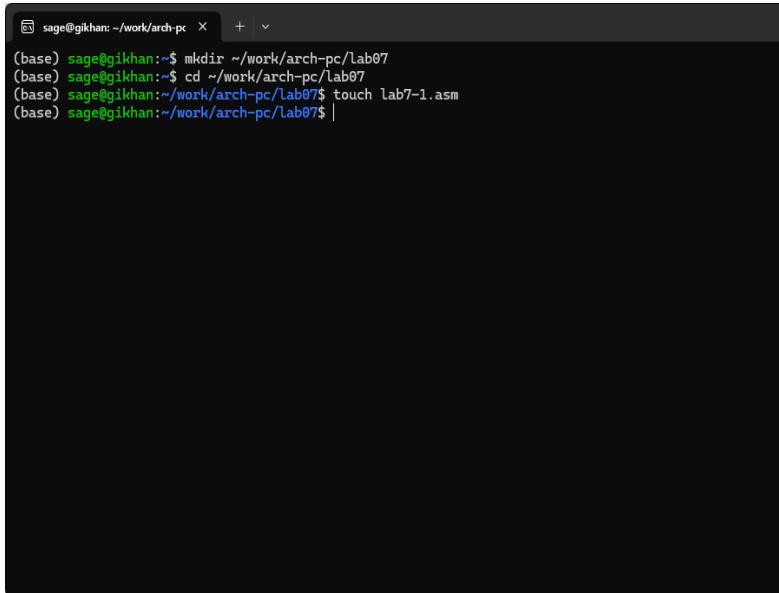
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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

4.1 Реализация переходов в NASM

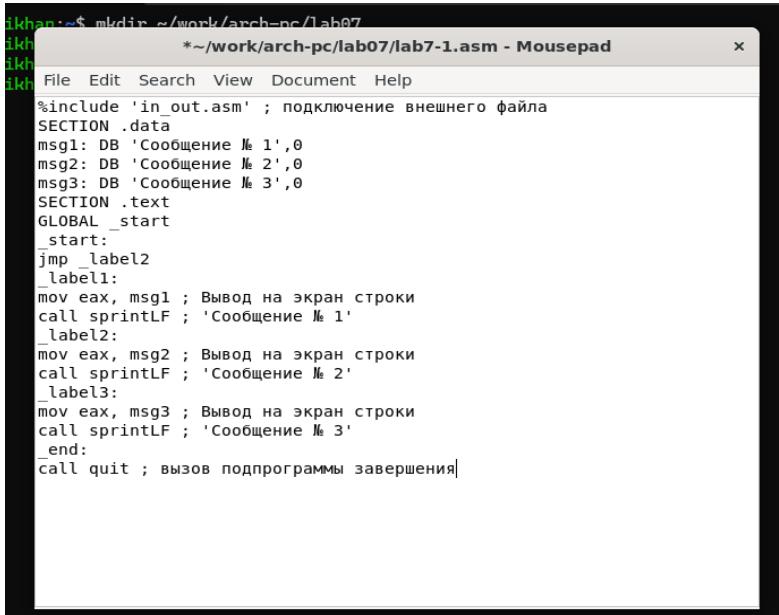
Создаю каталог для программ лабораторной работы №7 (рис. 1).



```
(base) sage@gikhan:~/work/arch-pc ~ + 
(base) sage@gikhan:~$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

Рис. 1: Создание каталога и файла для программы

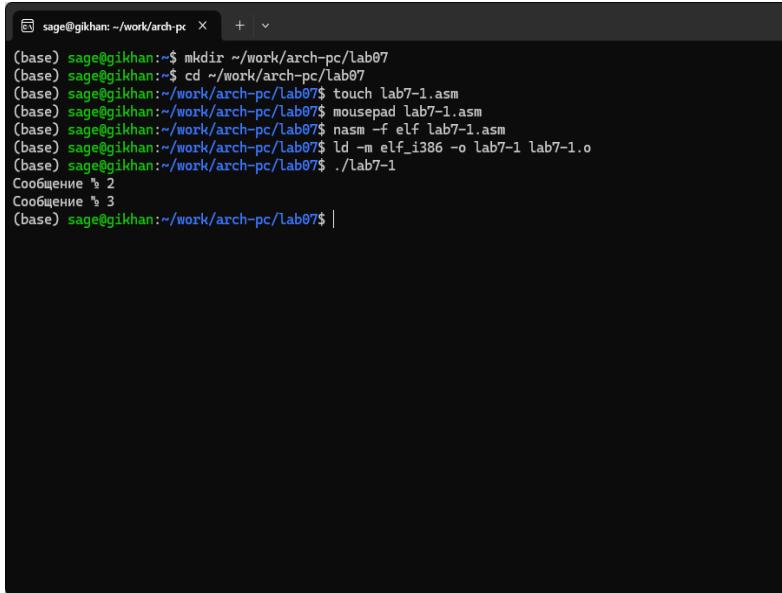
Копирую код из листинга в файл будущей программы. (рис. 2).



```
ikhan:~$ mkdir ~/work/arch-pc/lab07
ikhan:~/work/arch-pc/lab07* *~/work/arch-pc/lab07/lab7-1.asm - Mousepad
ikhan:~/work/arch-pc/lab07 File Edit Search View Document Help
%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|
```

Рис. 2: Сохранение программы

При запуске программы я убедился в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).



The screenshot shows a terminal window with the following command history:

```
(base) sage@gikhan:~/work/arch-pc ~
(base) sage@gikhan:~$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-1
Сообщение % 2
Сообщение % 3
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

Рис. 3: Запуск программы

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

```
lkh:~$ mkdir ~/work/arch-pc/lab07
lkh:~/work/arch-pc/lab07$ mousepad lab7-1.asm - Mousepad
lkh:~/work/arch-pc/lab07$ ~ /work/arch-pc/lab07/lab7-1.asm - Mousepad
lkh:~/work/arch-pc/lab07$ File Edit Search View Document Help
lkh:~/work/arch-pc/lab07$ %include 'in.out.asm' ; подключение внешнего файла
lkh:~/work/arch-pc/lab07$ SECTION .data
lkh:~/work/arch-pc/lab07$ msg1: DB 'Сообщение № 1',0
lkh:~/work/arch-pc/lab07$ msg2: DB 'Сообщение № 2',0
lkh:~/work/arch-pc/lab07$ msg3: DB 'Сообщение № 3',0
lkh:~/work/arch-pc/lab07$ SECTION .text
lkh:~/work/arch-pc/lab07$ GLOBAL _start
lkh:~/work/arch-pc/lab07$ _start:
lkh:~/work/arch-pc/lab07$     jmp _label2
lkh:~/work/arch-pc/lab07$ _label1:
lkh:~/work/arch-pc/lab07$     mov eax, msg1 ; Вывод на экран строки
lkh:~/work/arch-pc/lab07$     call sprintLF ; 'Сообщение № 1'
lkh:~/work/arch-pc/lab07$     jmp _end
lkh:~/work/arch-pc/lab07$ _label2:
lkh:~/work/arch-pc/lab07$     mov eax, msg2 ; Вывод на экран строки
lkh:~/work/arch-pc/lab07$     call sprintLF ; 'Сообщение № 2'
lkh:~/work/arch-pc/lab07$     jmp _label1
lkh:~/work/arch-pc/lab07$ _label3:
lkh:~/work/arch-pc/lab07$     mov eax, msg3 ; Вывод на экран строки
lkh:~/work/arch-pc/lab07$     call sprintLF ; 'Сообщение № 3'
lkh:~/work/arch-pc/lab07$ _end:
lkh:~/work/arch-pc/lab07$ call quit : вызов программы завершения
```

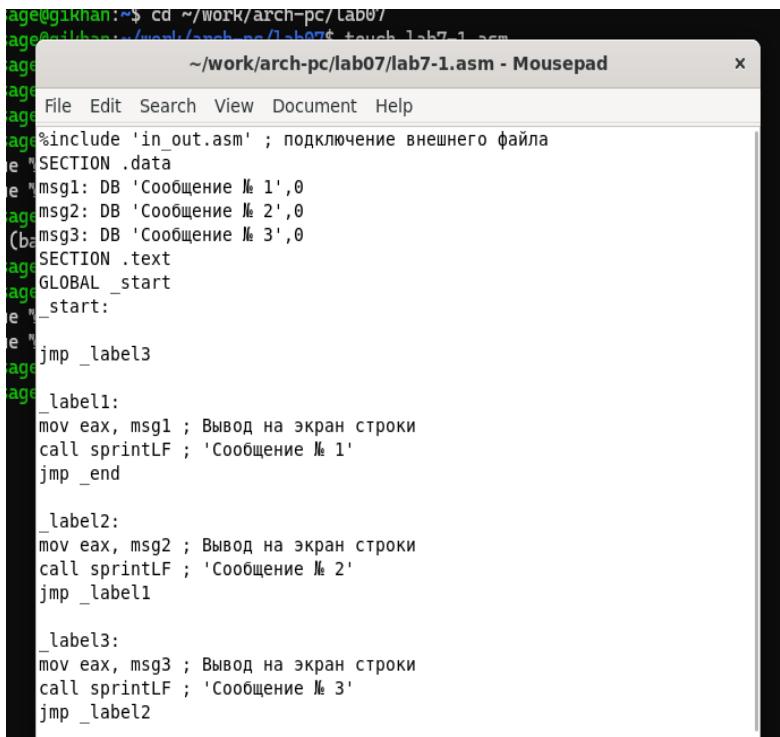
Рис. 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

```
sage@gikhan:~/work/arch-pc$ + ~
(base) sage@gikhan:~$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

Рис. 5: Запуск изменененной программы

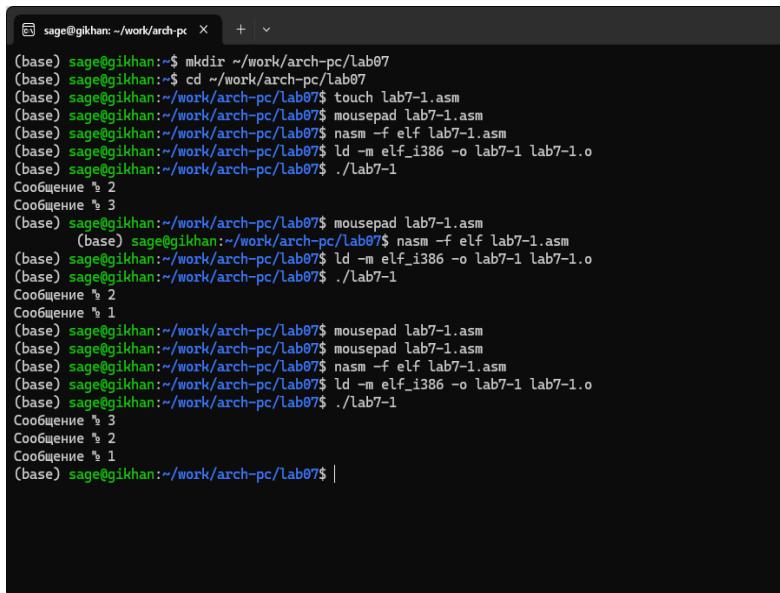
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).



```
sage@gikhan:~$ cd ~/work/arch-pc/lab07
sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
File Edit Search View Document Help
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
sage%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
```

Рис. 6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).



```
sage@gikhan:~/work/arch-pc$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
Сообщение № 1
```

Рис. 7: Проверка изменений

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

```
sage@giikan:~/work/arch-pc/lab07$ mousepad lab7-2.asm - Mousepad
sage
File Edit Search View Document Help
sage
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
ine A dd '20'
ine C dd '50'
section .bss
max resb 10
B resb 10
section .text
ine global _start
ine start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
ine ; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'max'
; ----- Записывается 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравнивают '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'
; ----- Сравнивают '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 iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 8: Сохранение новой программы

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

```
sage@gikan:~/work/arch-pc$ touch lab7-2.asm
(base) sage@gikan:~/work/arch-pc$ mousepad lab7-2.asm
(base) sage@gikan:~/work/arch-pc$ nasm -f elf lab7-2.asm
(base) sage@gikan:~/work/arch-pc$ ld -m elf_i386 -o lab7-2 lab7-2.o
(base) sage@gikan:~/work/arch-pc$ ./lab7-2
Введите B: 25
Наибольшее число: 50
(base) sage@gikan:~/work/arch-pc$ ./lab7-2
Введите B: 60
Наибольшее число: 60
(base) sage@gikan:~/work/arch-pc$ ./lab7-2
Введите B: 10
Наибольшее число: 50
(base) sage@gikan:~/work/arch-pc$ |
```

Рис. 9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

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

```
~/work/arch-pc/lab07/lab7-2.lst - Mousepad
File Edit Search View Document Help
1      %include 'in_out.asm'
1      <1> ;----- slen -----
1      ; Функция вычисления длины сообщения
1      ; в слен
2      push    ebx
3      mov     ebx, eax
4      00000000 53
5      00000001 89C3
6
7      nextchar:
8      00000003 803800
9      00000006 7403
10     00000008 40
11     00000009 EBFB
12
13     finished:
14     0000000B 29D8
15     0000000D 5B
16     0000000E C3
17
18
19     ;----- sprint -----
20     ; Функция печати сообщения
21     ; входные данные: mov eax,<mess>
22     sprint:
23     0000000F 52
24     00000010 51
25     00000011 53
26     00000012 50
27     00000013 E8E8FFFFFF
28
29     00000018 89C2
30     0000001A 58
31
32     0000001B 89C1
33     0000001D BB01000000
34     00000022 B804000000
35     00000027 CD80
36
37     00000029 5B
38     0000002A 59
39     0000002B 5A
40     0000002C C3
41
42
43     ;----- sprintLF -----
44     ; Функция печати сообщения с переносом строки
45     ; входные данные: mov eax,<mess>
46     sprintLF:
47     0000002D E8DDFFFFFF
48
49     00000032 50
50     00000013 B80A000000
```

Рис. 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно 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. 11).

```

*~/work/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
伦n %include 'in.out.asm'
section .data
ия msg1 db 'Введите В: ',0h
ую msg2 db "Наибольшее число: ",0h
цъзA dd '20'
иммC dd '50'
имк section .bss
max resb 10
. ИВ resb 10
section .text
Обы global _start
_start:
фай; ----- Вывод сообщения 'Введите В: '
Соз mov eax,msg1
call sprint
-f ; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
т: ; ----- Преобразование 'В' из символа в число
mov eax,B
т lcall atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
имат; ----- Записываем 'А' в переменную 'max'
рёх mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
трой; ----- Сравниваем 'A' и 'C' (как символы)
тъ cmp ecx,[C] ; Сравниваем 'A' и 'C',
je check_B; если 'A>C' то переход на метку 'check_B',
-f mov ecx,[C]; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
ие check_B;
mov eax,
call atoi ; Вызов подпрограммы перевода символа в число
За mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
Напом mov ecx,[max]
снап cmp ecx,[B]; Сравниваем 'max(A,C)' и 'В'
Зна jg fin ; если 'max(A,C)>B', то переход на 'fin',
примов ecx,[B]; иначе 'ecx = B'
его mov [max],ecx
Нап ; ----- Вывод результата
Нап fin:
зна mov eax, msg2
выб call sprint ; Вывод сообщения 'Наибольшее число: '

```

Рис. 11: 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. 12).

```

*~/work/arch-pc/lab07/lab7-2.lst - Mousepad
File Edit Search View Document Help
4 00000001 00000000000000000000000000000000 msg2 dd "наибольшее число:",0h
4 000000025 D9852001677098801101-
4 00000002E D9852000E3A2000
5 00000002F 00000000000000000000000000000000
6 000000039 35300000
7 0000000000 <res Ah>
9 00000000A <res Ah>
10 00000000B B resb 10
11 00000000C section .text
12 00000000D _start:
13 00000000E 00 mov eax,msg1
14 00000000F E810FFFFF Выход сообщения 'Введите В: '
15 000000010 00 call sprint
16 000000011 00 ; ----- Ввод 'В'
17 000000012 00 mov ecx,B
18 000000013 00 mov edx,10
19 000000014 00 call sread
20 000000015 00 ; ----- Преобразование 'В' из символа в число
21 000000016 00 mov eax,B
22 000000017 00 lcall atoi ; Вызов подпрограммы перевода символа в число
23 000000018 00 mov [B],eax ; запись преобразованного числа в 'В'
24 000000019 00 ; ----- Записываем 'А' в переменную 'max'
25 000000010 00 mov ecx,[A] ; 'ecx = A'
26 000000011 00 ; ----- Сравниваем 'A' и 'C' (как символы)
27 000000012 00 cmp ecx,[C] ; Сравниваем 'A' и 'C',
28 000000013 00 je check_B; если 'A>C' то переход на метку 'check_B',
29 000000014 00 mov ecx,[C]; иначе 'ecx = C'
30 000000015 00 mov [max],ecx ; 'max = C'
31 000000016 00 ; ----- Преобразование 'max(A,C)' из символа в число
32 000000017 00 check_B:
33 000000018 00 mov eax,
34 000000019 00 ; ----- combination of opcode and operand
35 00000001A 00 call atoi ; Вызов подпрограммы перевода символа в число
36 00000001B 00 mov [max],eax ; запись преобразованного числа в 'max'
37 00000001C 00 ; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
38 00000001D 00 cmp ecx,[B]; Сравниваем 'max(A,C)' и 'В'
39 00000001E 00 je fin; если 'max(A,C)>B', то переход на 'fin',
40 00000001F 00 mov ecx,[B]; иначе 'ecx = B'
41 000000020 00 mov [max],ecx ; 'max = B'
42 000000021 00 ; ----- Вывод результата
43 000000022 00 fin:
44 000000023 00 mov eax, msg2
45 000000024 00 call sprint ; Вывод сообщения 'Наибольшее число: '
46 000000025 00 mov eax,[max]
47 000000026 00 call iprintfL ; Вызов 'max(A,B,C)'.
48 000000027 00 call quit ; Выход
49 000000028 00

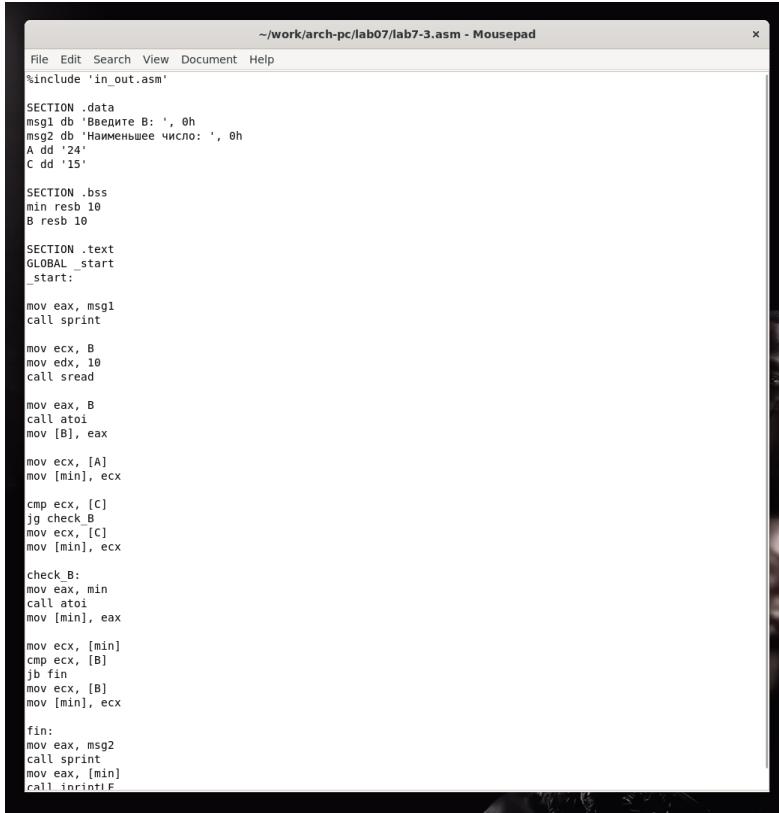
```

Рис. 12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должен был получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе

и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).



```
~/work/arch-pc/lab07/lab7-3.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

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]
    jg check_B
    mov ecx, [C]
    mov [min], ecx

check_B:
    mov eax, min
    call atoi
    mov [min], eax

    mov ecx, [min]
    cmp ecx, [B]
    jb fin
    mov ecx, [B]
    mov [min], ecx

fin:
    mov eax, msg2
    call sprint
    mov eax, [min]
    call inprintf
```

Рис. 13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

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]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
```

Проверяю корректность написания первой программы (рис. 14).

```

(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/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
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-3.asm
x(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ld: cannot find lab7-2.o: No such file or directory
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 98
Наименьшее число: 15
(base) sage@gikhan:~/work/arch-pc/lab07$

```

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных а и х (рис. 15).

```

~/work/arch-pc/lab07/lab7-4.asm - Mousepad
File Edit Search View Document Help
%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:
    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
    jle add_values
    mov eax, esi
    jmp print_result

add_values:
    mov eax, edi
    add eax, esi

print_result:
    mov edi, eax
    mov eax, res
    call sprint
    mov eax, edi
    call iprintfLF
    call quit|

```

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 16).

```
@sage@gikhan:~/work/arch-pc X + v
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/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
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
x(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf_i386 -o lab7-2 lab7-2.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./Lab7-3
Введите В: 98
Наименьшее число: 15
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./Lab7-4
Введите значение переменной х: 5
Введите значение переменной а: 7
Результат: 12
(base) sage@gikhan:~/work/arch-pc/lab07$ ./Lab7-4
Введите значение переменной х: 6
Введите значение переменной а: 4
Результат: 4
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

Рис. 16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.