

Отчет по лабораторной работе №7

Архитектура компьютера

Николенко Анна Николаевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Реализация переходов в NASM	9
4.2	Изучение структуры файлы листинга	14
5	Задания для самостоятельной работы	16
5.1	1 пункт самостоятельной работы	16
5.2	2 пункт самостоятельной работы	18
6	Выводы	20
7	Листинги программ самостоятельной работы	21

Список иллюстраций

4.1	Создание каталога, переход в этот каталог и создание файла . . .	9
4.2	Редактирование файла	10
4.3	Создание исполняемого файла и его запуск	10
4.4	Редактирование файла	11
4.5	Редактирование файла	12
4.6	Создание исполняемого файла и его запуск	12
4.7	Создание файла	12
4.8	Редактирование файла	13
4.9	Создание исполняемого файла и его запуск	13
4.10	Создание файла листинга	14
4.11	Открытие файла	14
4.12	Открытие файла и удаление одного из операндов в инструкции .	15
4.13	Выполнение трансляции с получением файла листинга	15
4.14	ошибка в файле листинга	15
5.1	Написание программы по нахождению минимума среди 3-х чисел	17
5.2	Создание исполняемого файла и его запуск	17
5.3	Написание программы по вычислению функции	18
5.4	Создание исполняемого файла и его запуск	19

Список таблиц

1 Цель работы

Цель работы заключается в изучении команд условного и безусловного переходов, в приобретении навыков написания программ с использованием переходов, а также в знакомстве с назначением и структурой файла листинга

2 Задание

1. Проверить работу разных программ с использованием инструкции jmp.
2. Проверить работу программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.
3. Изучить структуры файлы листинга.
4. Написать программу нахождения наименьшей из 3 целочисленных переменных a, b, c.
5. Написать программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

3 Теоретическое введение

Безусловный переход выполняется инструкцией `jmp`, которая включает в себя адрес перехода, куда следует передать управление: `jmp`

`jmp label` - переход на метку `label` `jmp (label)` (label в квадратных скобках) - переход по адресу в памяти, помеченному меткой `label` `jmp eax` - переход по адресу из регистра `eax`

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов. Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора (такие как CF PF AF ZF SF)

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов: `cmp`, Команда `cmp`, так же как и команда вычитания, выполняет вычитание -, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Структура листинга: * номер строки * адрес — это смещение машинного

кода от начала текущего сегмента * машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности * исходный текст программы

4 Выполнение лабораторной работы

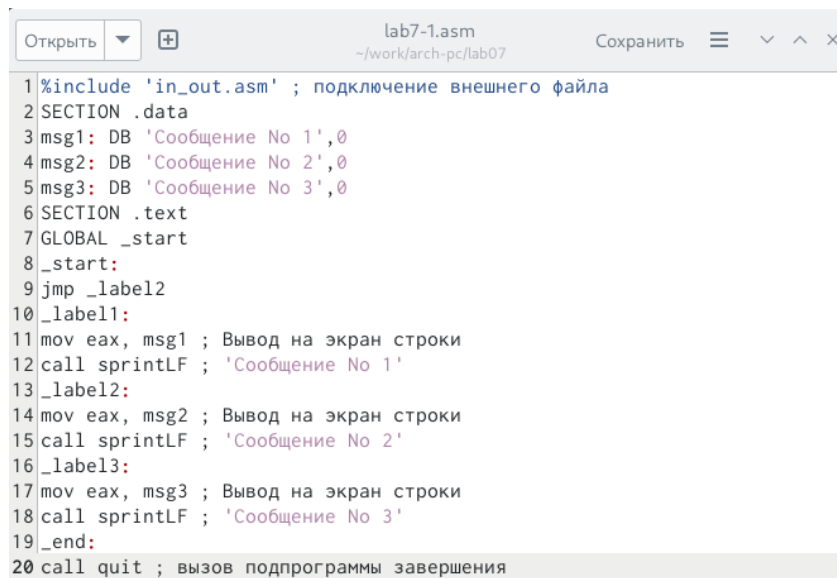
4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. [4.1]).

```
annikolenko@dk8n51 ~ $ mkdir ~/work/arch-pc/lab07
annikolenko@dk8n51 ~ $ cd ~/work/arch-pc/lab07
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 4.1: Создание каталога, переход в этот каталог и создание файла

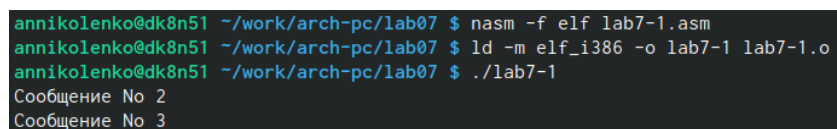
Ввожу в файл lab7-1.asm текст программы с использованием инструкции jmp (рис. [4.2]).



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение No 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение No 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение No 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [4.3]).



```
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
```

Рис. 4.3: Создание исполняемого файла и его запуск

Изменяю текст программы, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу (рис. [4.4]).

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10
11 _label1:
12 mov eax, msg1 ; Вывод на экран строки
13 call sprintLF ; 'Сообщение No 1'
14 jmp _end
15
16 _label2:
17 mov eax, msg2 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение No 2'
19 jmp _label1
20
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [??]) (рис. [??]).

```

annikolenko@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o

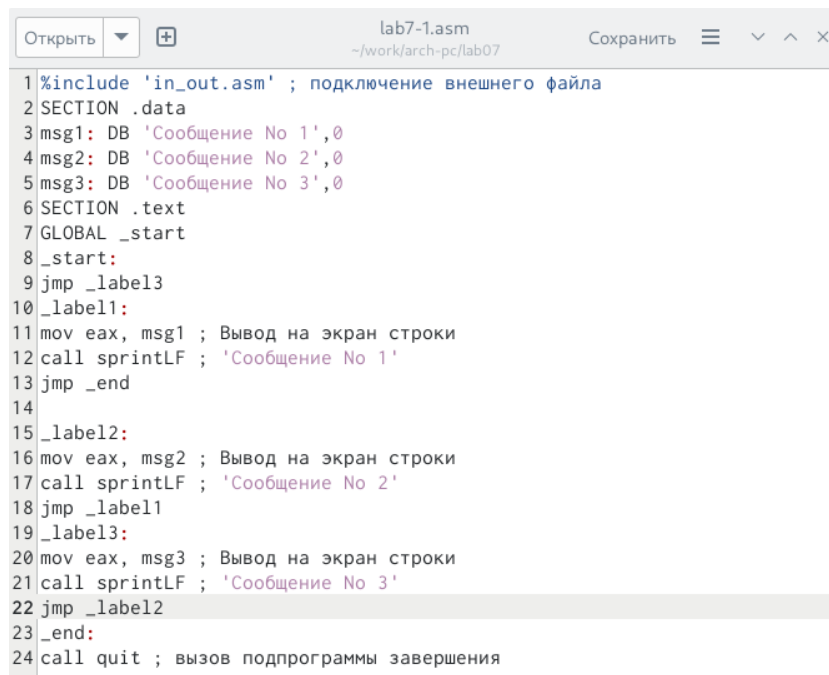
```

```

annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1

```

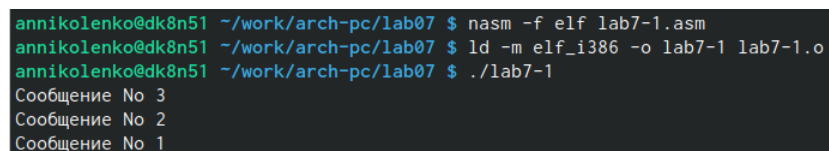
Изменяю текст программы, чтобы она выводила сначала ‘Сообщение № 3’, потом ‘Сообщение № 2’, а затем ‘Сообщение № 1’ и завершала работу (рис. [4.5]).



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение No 1'
13 jmp _end
14
15 _label2:
16 mov eax, msg2 ; Вывод на экран строки
17 call sprintf ; 'Сообщение No 2'
18 jmp _label1
19 _label3:
20 mov eax, msg3 ; Вывод на экран строки
21 call sprintf ; 'Сообщение No 3'
22 jmp _label2
23 _end:
24 call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Редактирование файла

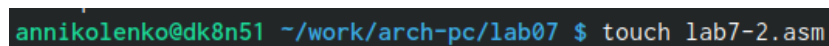
Создаю исполняемый файл и запускаю его (рис. [4.6]).



```
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 4.6: Создание исполняемого файла и его запуск

Создаю файл lab7-2.asm (рис. [4.7]).



```
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ touch lab7-2.asm
```

Рис. 4.7: Создание файла

Ввожу в файл lab7-2.asm текст программы, которая определяет и выводит на экран наибольшую из целочисленных переменных: А,В и С (рис. [4.8]).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2

```

Рис. 4.8: Редактирование файла

Создаю исполняемый файл и запускаю его несколько раз для разных значений B, чтобы проверить его работу (рис. [4.9]).

```

annikolenko@dk8n51 ~/work/arch-pc/lab07 $ touch lab7-2.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 2
Наибольшее число: 50
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 80
Наибольшее число: 80
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 10
Наибольшее число: 50

```

Рис. 4.9: Создание исполняемого файла и его запуск

4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm, указав ключ -l (рис. [4.10]).

```
annikolenko@dk8n51 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 4.10: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора mcedit (рис. [4.11]).

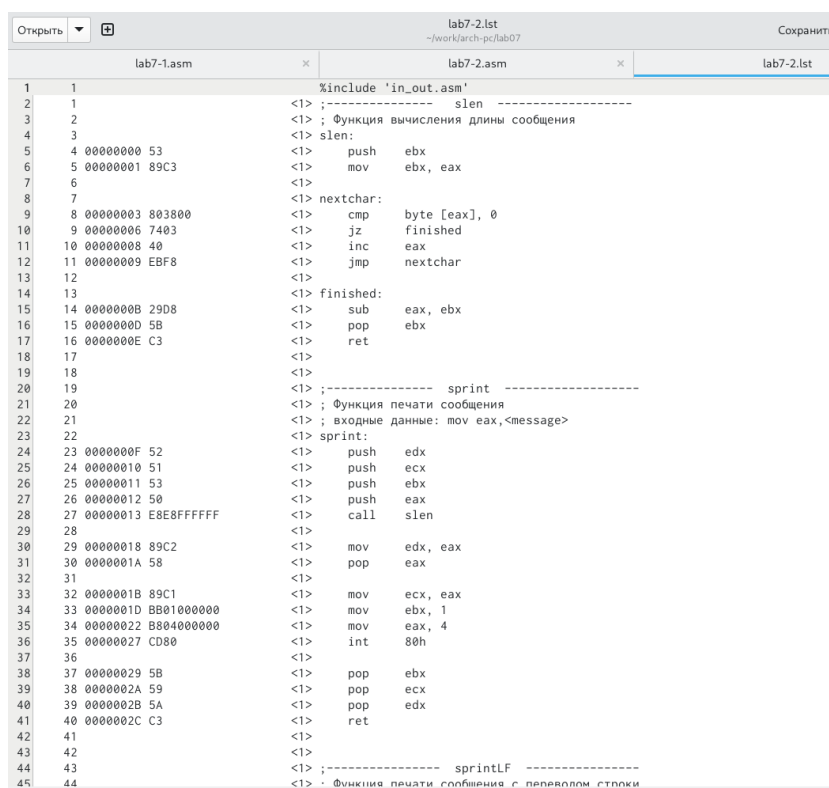


Рис. 4.11: Открытие файла

объяснение содержимого трёх строк файла листинга: * в строке 9 содержится номер строки [8], адрес [00000003], машинный код [803800] и содержимое строки кода [cmp byte [eax], 0] * в строке 12 содержится номер строки [11], адрес

[00000009], машинный код [EBF8] и содержимое строки кода [jmp nextchar] * в строке 31 содержится номер сторки [30], адресс [0000001A], машинный код [58] и содержимое строки кода [ror eax]

Открываю файл с программой lab7-2.asm и в одной из инструкций с двумя операндами удаляю один операнд (рис. [4.12]).

```
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax|
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
```

Рис. 4.12: Открытие файла и удаление одного из операндов в инструкции

Выполните трансляцию с получением файла листинга (рис. [4.13]).

```
annikolenko@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:21: error: invalid combination of opcode and operands
```

Рис. 4.13: Выполнение трансляции с получением файла листинга

В коде появилась ошибка,а также ее описание появилось в файле листинга (рис. [4.14]).

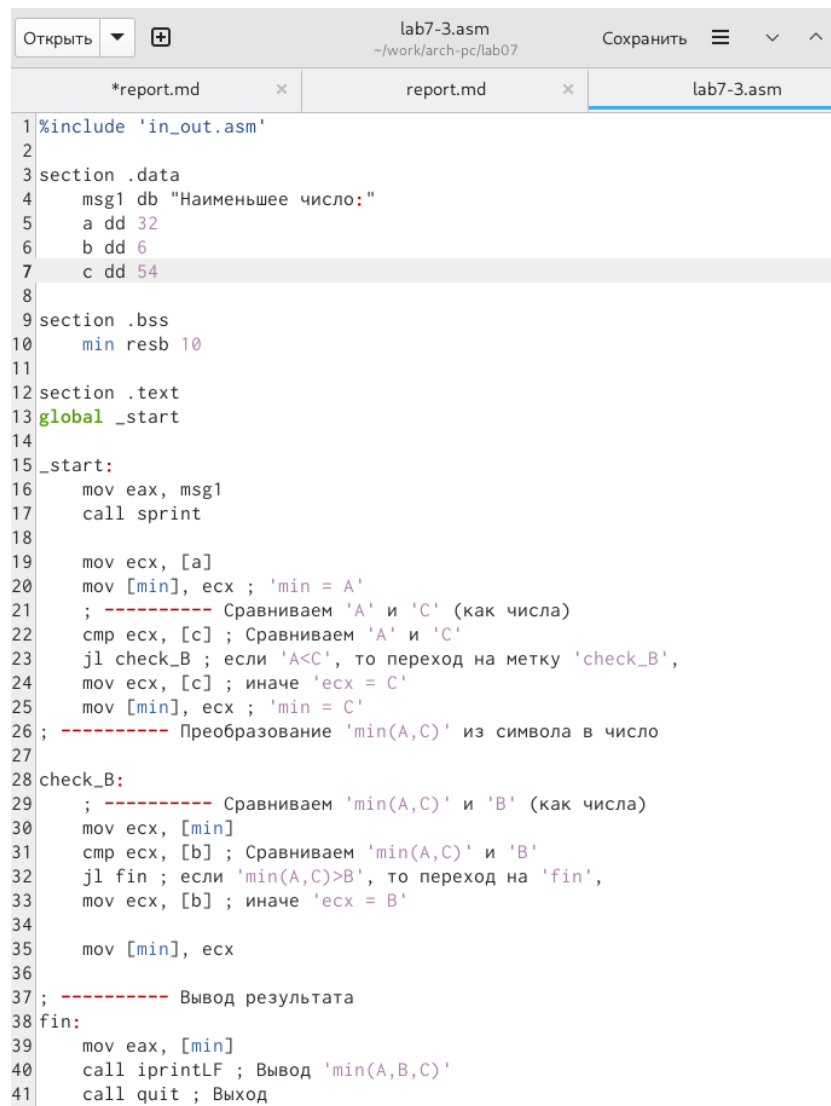
```
lab7-2.lst      [-----]  0 L:[193+14 207/226] *(13004/14547b) 0032 0x020  [*][X]
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread
20                          ; ----- Преобразование 'B' из символа в число
21                          mov eax
21                          *****
21                          error: invalid combination of opcode and operands
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в чи
23 00000106 A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
24                          ; ----- Записываем 'A' в переменную 'max'
```

Рис. 4.14: ошибка в файле листинга

5 Задания для самостоятельной работы

5.1 1 пункт самостоятельной работы

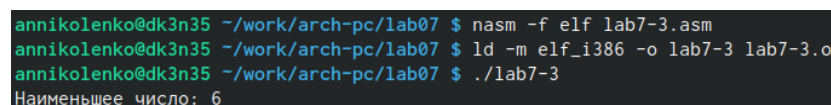
Написала программу нахождения наименьшей из 3 целочисленных переменных a,b,c, предварительно создав файл lab7-3.asm (мой вариант: 15) (рис. [5.1]).



```
1 %include 'in_out.asm'
2
3 section .data
4     msg1 db "Наименьшее число:"
5     a dd 32
6     b dd 6
7     c dd 54
8
9 section .bss
10    min resb 10
11
12 section .text
13 global _start
14
15 _start:
16     mov eax, msg1
17     call sprint
18
19     mov ecx, [a]
20     mov [min], ecx ; 'min = A'
21     ; ----- Сравниваем 'A' и 'C' (как числа)
22     cmp ecx, [c] ; Сравниваем 'A' и 'C'
23     jl check_B ; если 'A<C', то переход на метку 'check_B',
24     mov ecx, [c] ; иначе 'ecx = C'
25     mov [min], ecx ; 'min = C'
26     ; ----- Преобразование 'min(A,C)' из символа в число
27
28 check_B:
29     ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
30     mov ecx, [min]
31     cmp ecx, [b] ; Сравниваем 'min(A,C)' и 'B'
32     jl fin ; если 'min(A,C)>B', то переход на 'fin',
33     mov ecx, [b] ; иначе 'ecx = B'
34
35     mov [min], ecx
36
37 ; ----- Вывод результата
38 fin:
39     mov eax, [min]
40     call iprintf ; Вывод 'min(A,B,C)'
41     call quit ; Выход
```

Рис. 5.1: Написание программы по нахождению минимума среди 3-х чисел

Создаю исполняемый файл и проверяю его работу. Все выполнено верно (рис. [5.2]).

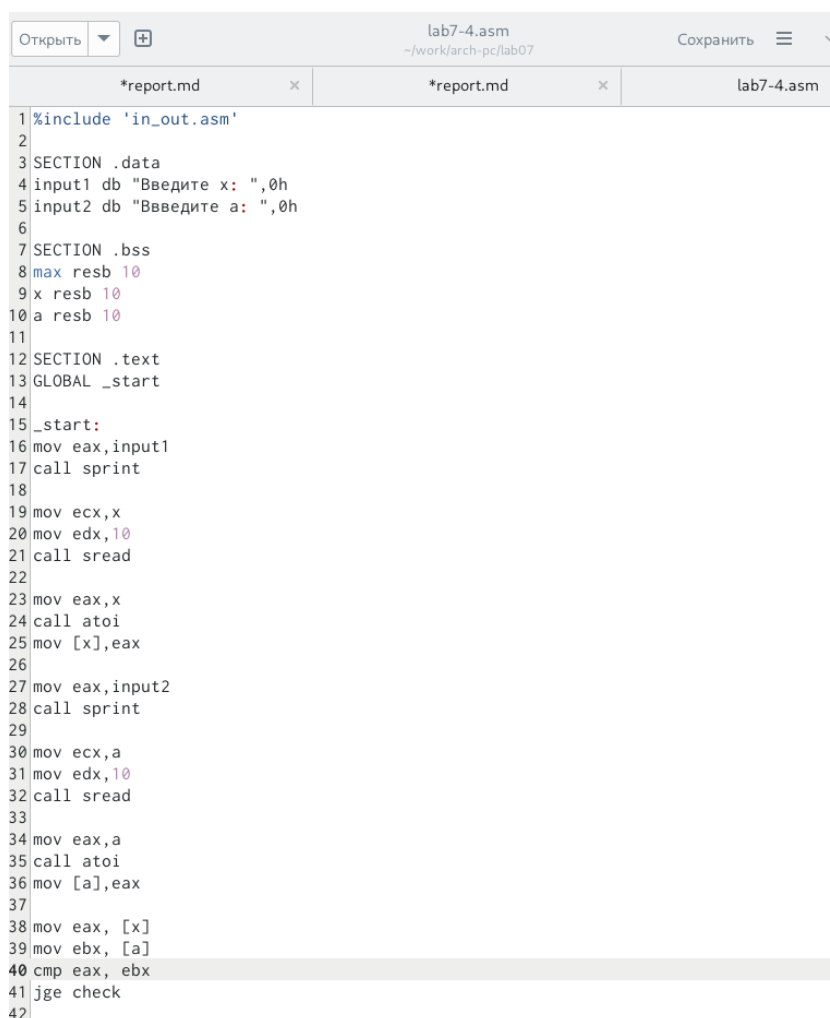


```
annikolenko@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
annikolenko@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
annikolenko@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 6
```

Рис. 5.2: Создание исполняемого файла и его запуск

5.2 2 пункт самостоятельной работы

Написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. предварительно для этого создала файл lab7-4.asm (мой вариант: 15) (рис. [5.3]).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 input1 db "Введите x: ",0h
5 input2 db "Введите a: ",0h
6
7 SECTION .bss
8 max resb 10
9 x resb 10
10 a resb 10
11
12 SECTION .text
13 GLOBAL _start
14
15 _start:
16 mov eax,input1
17 call sprint
18
19 mov ecx,x
20 mov edx,10
21 call sread
22
23 mov eax,x
24 call atoi
25 mov [x],eax
26
27 mov eax,input2
28 call sprint
29
30 mov ecx,a
31 mov edx,10
32 call sread
33
34 mov eax,a
35 call atoi
36 mov [a],eax
37
38 mov eax,[x]
39 mov ebx,[a]
40 cmp eax,ebx
41 jge check
42
```

Рис. 5.3: Написание программы по вычислению функции

Создаю исполняемый файл и проверяю его работу. Все выполнено верно (рис. [5.4]).

```
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 2
Введите a: 3
13
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
annikolenko@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 4
Введите a: 2
14
```

Рис. 5.4: Создание исполняемого файла и его запуск

6 Выводы

Я изучила основные принципы работы с условным и безусловным переходом в assembler и основы чтения файлов листинга.

7 Листинги программ самостоятельной работы

1. Программа для 1-го пункта самостоятельной работы

```
%include 'in_out.asm'
section .data msg1 db "Наименьшее число:" a dd 32 b dd 6 c dd 54
section .bss min resb 10
section .text global _start
_start: mov eax, msg1 call sprint

mov ecx, [a]
mov [min], ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как числа)
cmp ecx, [c] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx, [c] ; иначе 'ecx = C'
mov [min], ecx ; 'min = C'

; ---- Преобразование 'min(A,C)' из символа в число
check_B: ; ---- Сравниваем 'min(A,C)' и 'B' (как числа) mov ecx, [min] cmp ecx,
[b] ; Сравниваем 'min(A,C)' и 'B' jl fin ; если 'min(A,C)>B', то переход на 'fin', mov
ecx, [b] ; иначе 'ecx = B'

mov [min], ecx
```

; ——— Вывод результата fin: mov eax, [min] call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

2. Программа для 2-го пункта самостоятельной работы

```
%include 'in_out.asm'
SECTION .data input1 db "Введите x:",0h input2 db "Введите a:",0h
SECTION .bss max resb 10 x resb 10 a resb 10
SECTION .text GLOBAL _start
_start: mov eax,input1 call sprint
mov ecx,x mov edx,10 call sread
mov eax,x call atoi mov [x],eax
mov eax,input2 call sprint
mov ecx,a mov edx,10 call sread
mov eax,a call atoi mov [a],eax
mov eax, [x] mov ebx, [a] cmp eax, ebx jge check
mov eax, [a] mov ebx, 10 add eax, ebx call iprintLF call quit
check: mov eax, [x] mov ebx, 10 add eax, ebx call iprintLF call quit
```