

Лабораторная работа №8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Медникова Екатерина Михайловна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	12
5	Выводы	17

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Ввод текста из листинга 8.1	7
3.3	Создание и запуск файла	8
3.4	Изменение текста программы	8
3.5	Создание и проверка работы файла	8
3.6	Изменение текста программы	9
3.7	Создание и проверка работы файла	9
3.8	Создание файла lab8-2.asm	9
3.9	Ввод программы из листинга 8.3	10
3.10	Создание исполняемого файла и проверка его работы	10
3.11	Создание файла листинга	10
3.12	push eax	11
3.13	pop eax	11
3.14	dec ecx	11
3.15	Удаление max	11
3.16	Выполнение трансляции и получение ошибки	11
4.1	Файлы	12
4.2	Программа	13
4.3	Программа	14
4.4	Результат	14
4.5	Программа	15
4.6	Программа	16
4.7	Результат	16

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 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.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление:

`jmp`

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

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.

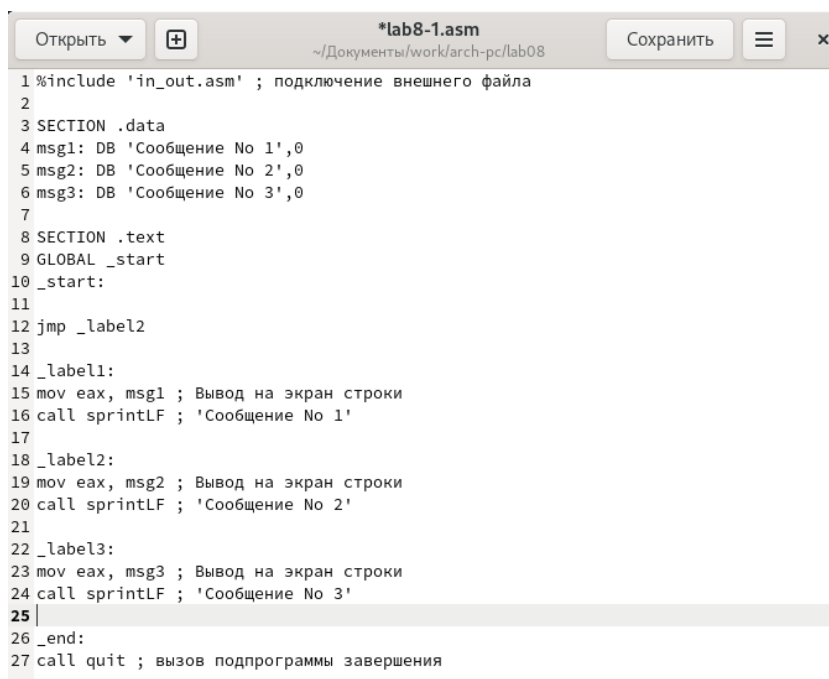
3 Выполнение лабораторной работы

1. Создала каталог для программ лабораторной работы No 8, перешла в него и создала файл lab8-1.asm:

```
[emmednikova@fedora arch-pc]$ mkdir lab08  
[emmednikova@fedora arch-pc]$ cd lab08  
[emmednikova@fedora lab08]$ touch lab8-1.asm  
[emmednikova@fedora lab08]$
```

Рис. 3.1: Создание каталога и файла

2. Ввела в файл lab8-1.asm текст программы из листинга 8.1.



```
*lab8-1.asm  
~/Документы/work/arch-pc/lab08  
Сохранить  
Открыть  
1 %include 'in_out.asm' ; подключение внешнего файла  
2  
3 SECTION .data  
4 msg1: DB 'Сообщение No 1',0  
5 msg2: DB 'Сообщение No 2',0  
6 msg3: DB 'Сообщение No 3',0  
7  
8 SECTION .text  
9 GLOBAL _start  
10 _start:  
11  
12 jmp _label2  
13  
14 _label1:  
15 mov eax, msg1 ; Вывод на экран строки  
16 call sprintf ; 'Сообщение No 1'  
17  
18 _label2:  
19 mov eax, msg2 ; Вывод на экран строки  
20 call sprintf ; 'Сообщение No 2'  
21  
22 _label3:  
23 mov eax, msg3 ; Вывод на экран строки  
24 call sprintf ; 'Сообщение No 3'  
25 |  
26 _end:  
27 call quit ; вызов подпрограммы завершения
```

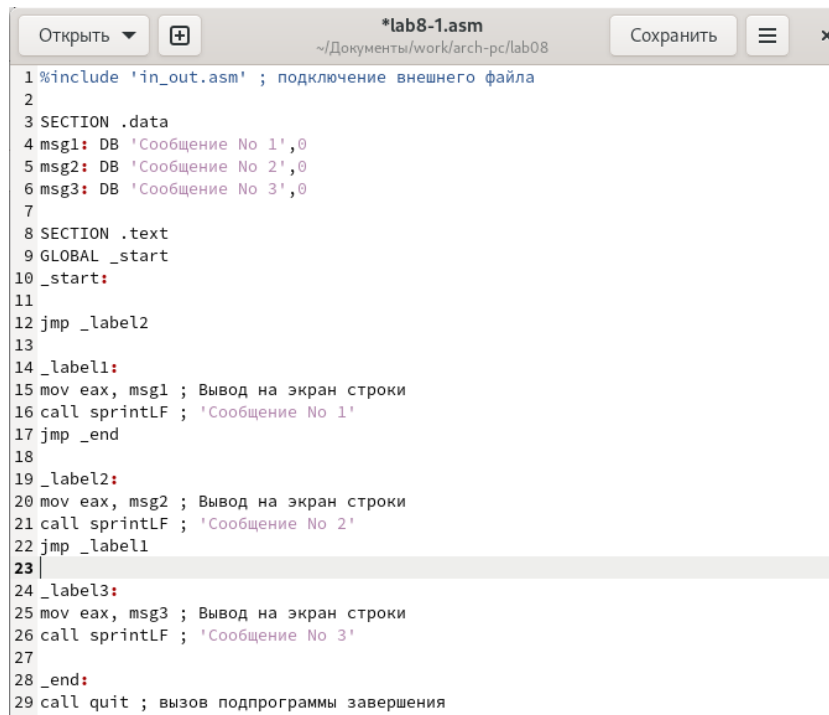
Рис. 3.2: Ввод текста из листинга 8.1

Создала исполняемый файл и запустила его.

```
[emmednikova@fedora lab08]$ nasm -f elf lab8-1.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[emmednikova@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 3
[emmednikova@fedora lab08]$
```

Рис. 3.3: Создание и запуск файла

Изменила текст программы в соответствии с листингом 8.2.



```
*lab8-1.asm
~/Документы/work/arch-pc/lab08
Сохранить

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

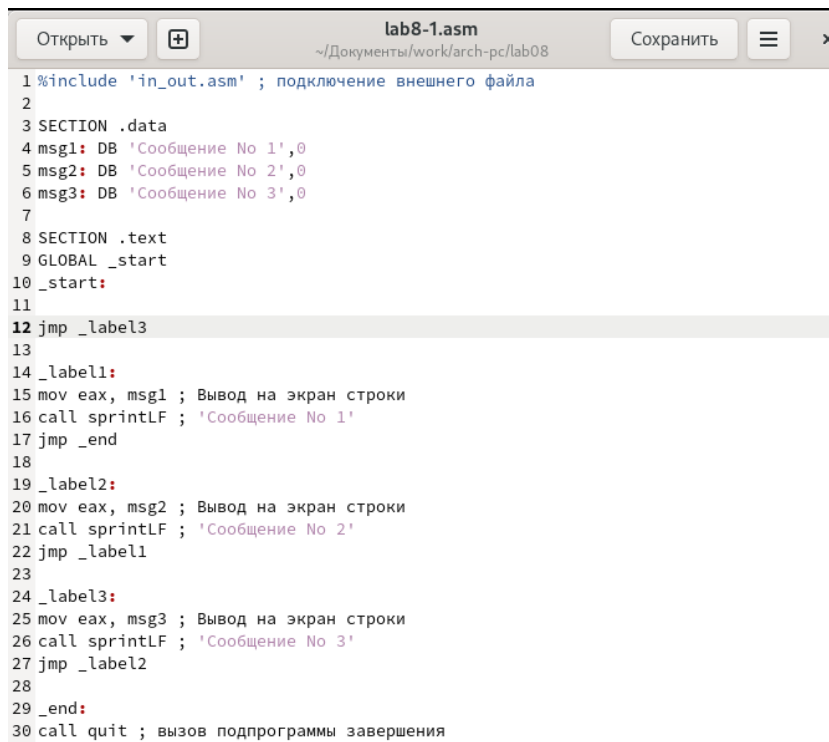
Рис. 3.4: Изменение текста программы

Создала исполняемый файл и проверила его работу.

```
[emmednikova@fedora lab08]$ nasm -f elf lab8-1.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[emmednikova@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 1
[emmednikova@fedora lab08]$
```

Рис. 3.5: Создание и проверка работы файла

Изменила текст программы так, чтобы её вывод соответствовал инструкции, прописанной в лабораторной работе.

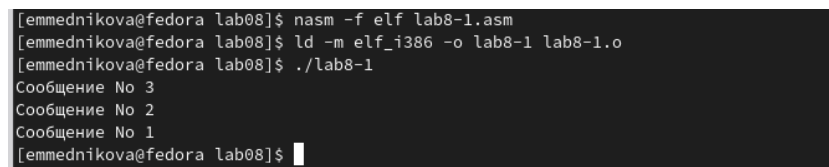


```
lab8-1.asm
~/Документы/work/arch-pc/lab08
Сохранить

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

Рис. 3.6: Изменение текста программы

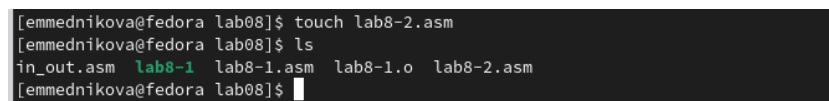
Создала исполняемый файл и проверила его работу.



```
[emmednikova@fedora lab08]$ nasm -f elf lab8-1.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[emmednikova@fedora lab08]$ ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[emmednikova@fedora lab08]$
```

Рис. 3.7: Создание и проверка работы файла

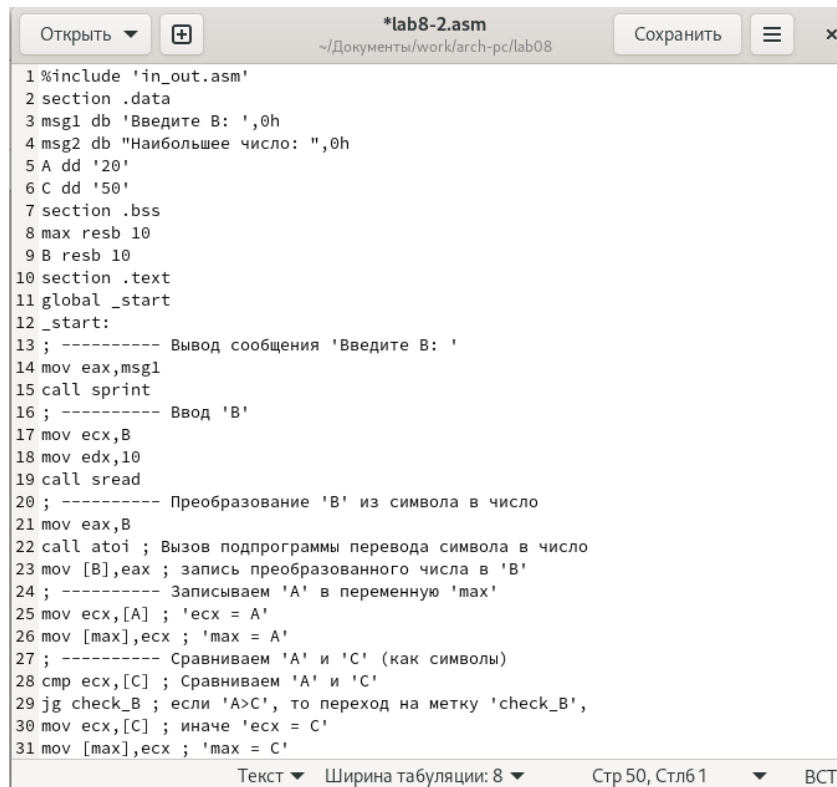
3. Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08.



```
[emmednikova@fedora lab08]$ touch lab8-2.asm
[emmednikova@fedora lab08]$ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
[emmednikova@fedora lab08]$
```

Рис. 3.8: Создание файла lab8-2.asm

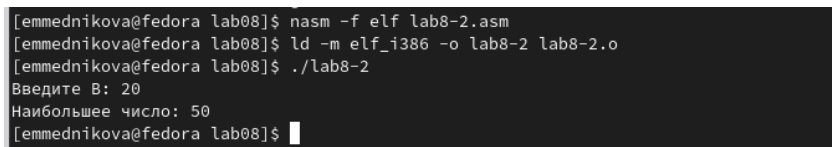
Ввела в созданный файл текст программы из листинга 8.3.



```
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'
```

Рис. 3.9: Ввод программы из листинга 8.3

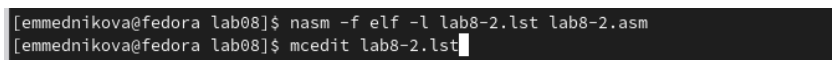
Создала исполняемый файл и проверила его работу.



```
[emmednikova@fedora lab08]$ nasm -f elf lab8-2.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[emmednikova@fedora lab08]$ ./lab8-2
Введите B: 20
Наибольшее число: 50
[emmednikova@fedora lab08]$
```

Рис. 3.10: Создание исполняемого файла и проверка его работы

4. Создала файл листинга для программы из файла lab8-2.asm. Открыла файл листинга lab8-2.lst с помощью текстового редактора mcedit.



```
[emmednikova@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[emmednikova@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.11: Создание файла листинга

27 строка. Адрес 00000012. Машинный код 50. Push eax (исходный текст программы) выделяет место наверху стека и помещает туда значение из регистра eax.

```
27 00000012 50      <1>      push     eax
```

Рис. 3.12: push eax

55 строка. Адрес 00000040. Машинный код 58. Pop eax (исходный текст программы) переносит любые данные из верхней части стека в eax и освобождает эту область памяти.

```
55 00000040 58      <1>      pop      eax
```

Рис. 3.13: pop eax

95 строка. Адрес 00000073. Машинный код 49. Dec ecx (исходный текст программы) уменьшает значение ecx на единицу.

```
95 00000073 49      <1>      dec      ecx.....
```

Рис. 3.14: dec ecx

Открыла файл с программой lab8-2.asm и удалила один операнд - max.

```
34 mov eax,max
```

Рис. 3.15: Удаление max

Выполнила трансляцию с получением файла листинга. Выдалась ошибка.

```
[emmednikova@fedora lab08]$ nasm -f elf lab8-2.asm
lab8-2.asm:34: error: invalid combination of opcode and operands
[emmednikova@fedora lab08]$
```

Рис. 3.16: Выполнение трансляции и получение ошибки

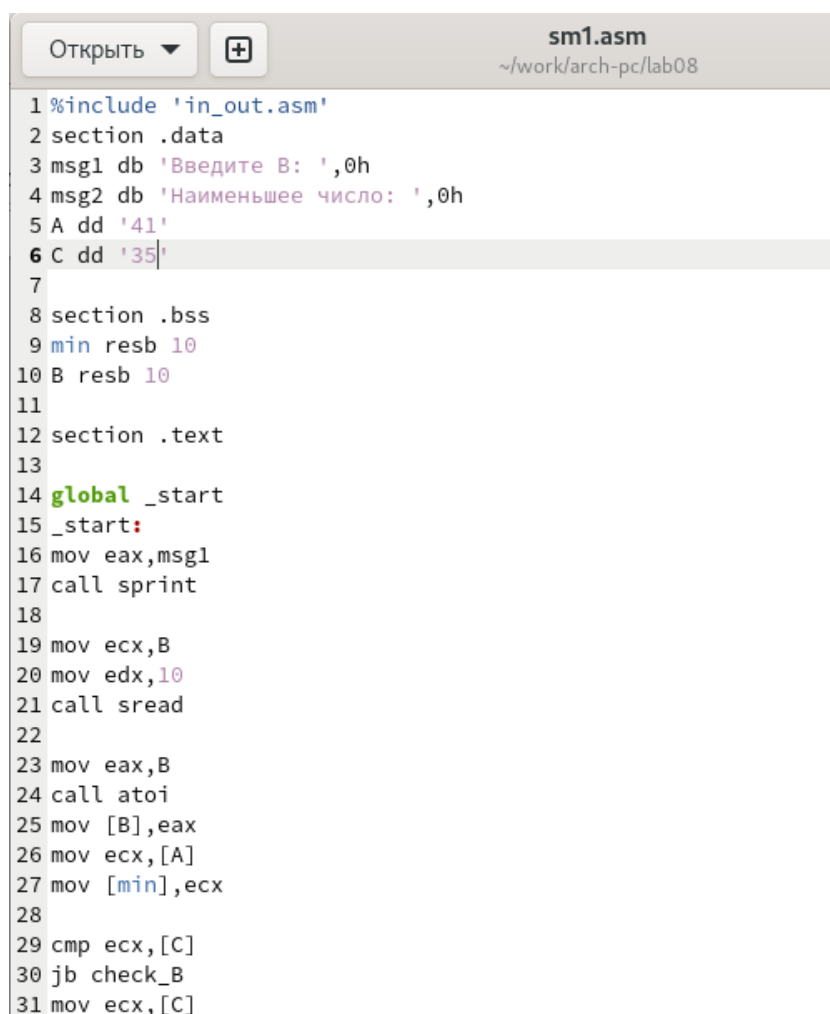
4 Самостоятельная работа

1. Создала файлы для самостоятельной работы.

```
[emmednikova@fedora lab08]$ touch sm1.asm
[emmednikova@fedora lab08]$ touch sm2.asm
[emmednikova@fedora lab08]$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.lst  sm2.asm
lab8-1      lab8-1.o    lab8-2.asm  sm1.asm
[emmednikova@fedora lab08]$
```

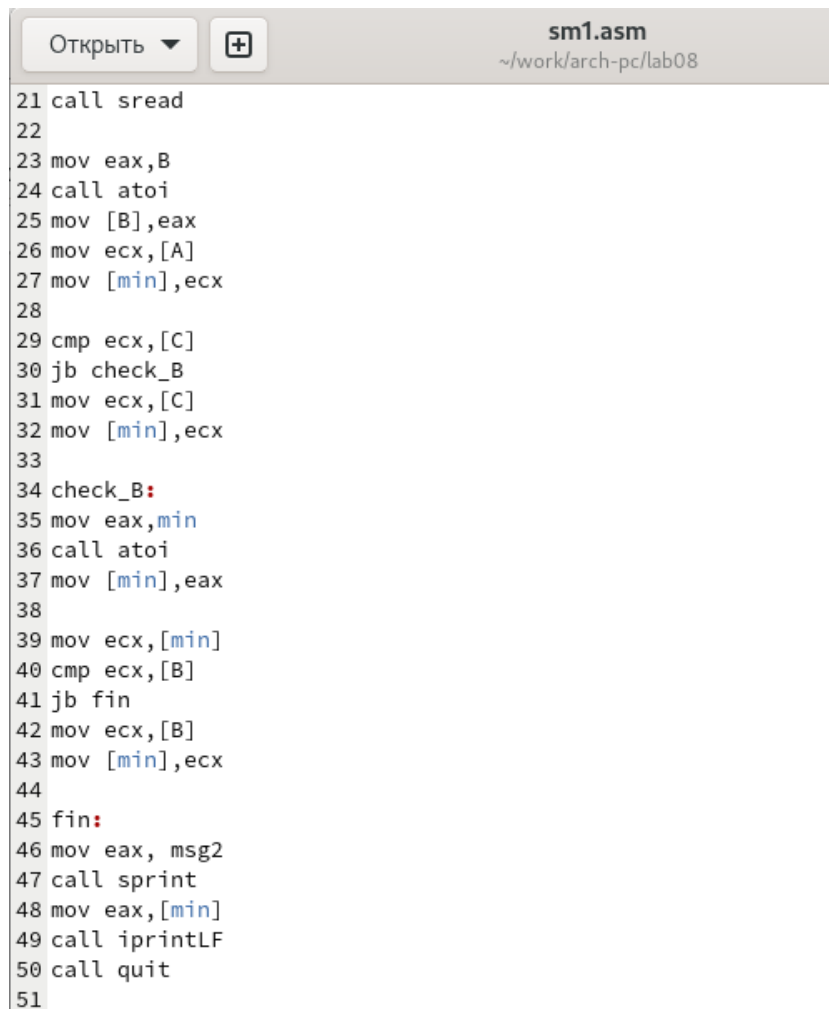
Рис. 4.1: Файлы

Написала программу для нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных взяла из таблицы 8.5 в соответствии с вариантом. У меня вариант 10. Значения: 41, 62, 35.



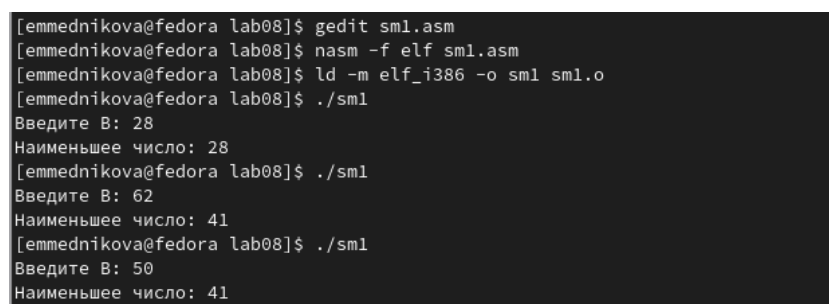
```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db 'Наименьшее число: ',0h
5 A dd '41'
6 C dd '35'
7
8 section .bss
9 min resb 10
10 B resb 10
11
12 section .text
13
14 global _start
15 _start:
16 mov eax,msg1
17 call sprint
18
19 mov ecx,B
20 mov edx,10
21 call sread
22
23 mov eax,B
24 call atoi
25 mov [B],eax
26 mov ecx,[A]
27 mov [min],ecx
28
29 cmp ecx,[C]
30 jb check_B
31 mov ecx,[C]
```

Рис. 4.2: Программа



```
Открыть ▾ + sm1.asm
~/work/arch-pc/lab08
21 call sread
22
23 mov eax,B
24 call atoi
25 mov [B],eax
26 mov ecx,[A]
27 mov [min],ecx
28
29 cmp ecx,[C]
30 jb check_B
31 mov ecx,[C]
32 mov [min],ecx
33
34 check_B:
35 mov eax,min
36 call atoi
37 mov [min],eax
38
39 mov ecx,[min]
40 cmp ecx,[B]
41 jb fin
42 mov ecx,[B]
43 mov [min],ecx
44
45 fin:
46 mov eax, msg2
47 call sprint
48 mov eax,[min]
49 call iprintLF
50 call quit
51
```

Рис. 4.3: Программа



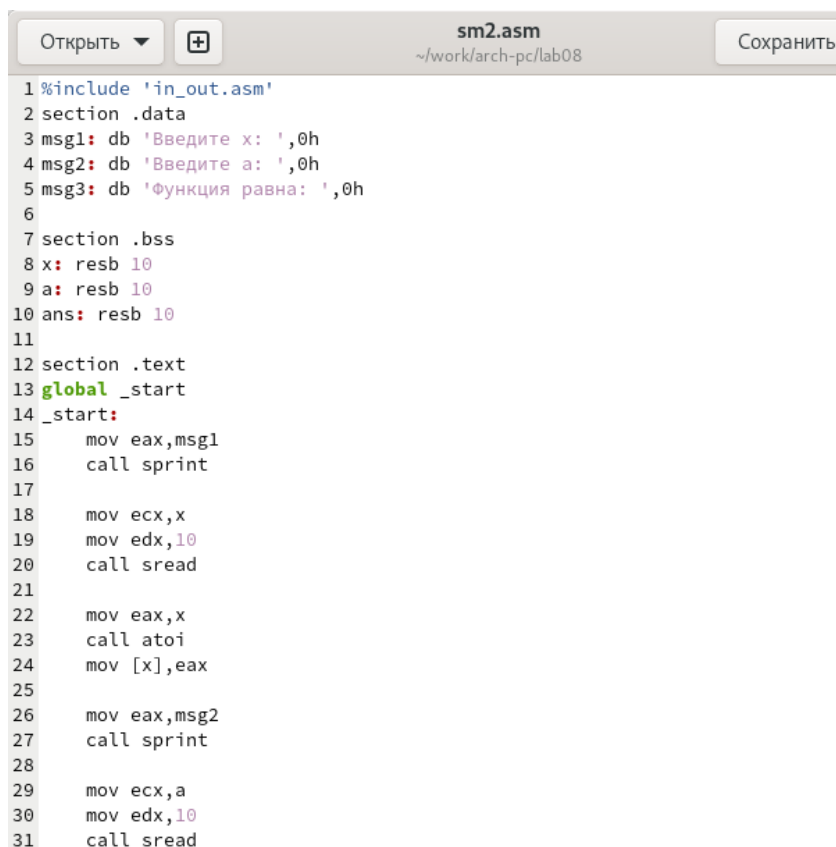
```
[emmednikova@fedora lab08]$ gedit sm1.asm
[emmednikova@fedora lab08]$ nasm -f elf sm1.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o sm1 sm1.o
[emmednikova@fedora lab08]$ ./sm1
Введите B: 28
Наименьшее число: 28
[emmednikova@fedora lab08]$ ./sm1
Введите B: 62
Наименьшее число: 41
[emmednikova@fedora lab08]$ ./sm1
Введите B: 50
Наименьшее число: 41
```

Рис. 4.4: Результат

Программа работает не совсем корректно, выдаёт неправильное число. Однако

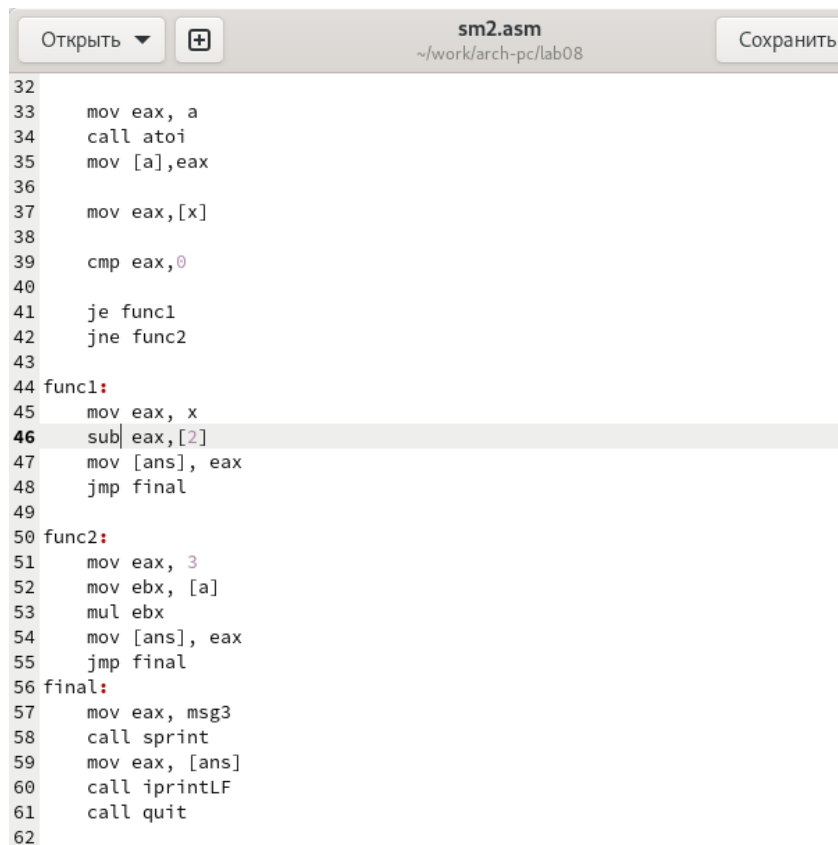
если ввести другое число b , которое меньше заданных, то работает верно.

2. Написала программу для вычисления функции, представленная в таблице 8.6 в соответствии с вариантом. У меня вариант 10. Значения (3;0) и (1;2)



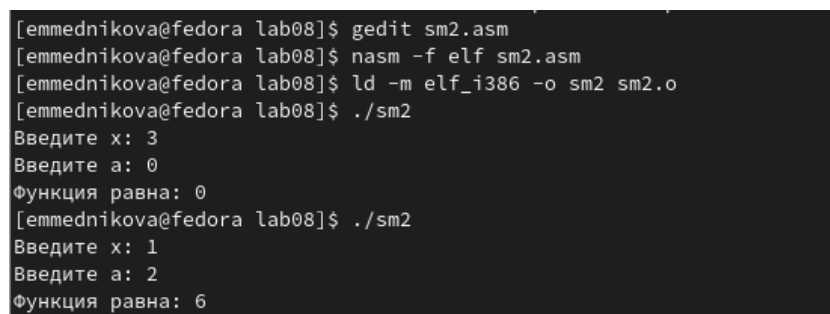
```
1 %include 'in_out.asm'
2 section .data
3 msg1: db 'Введите x: ',0h
4 msg2: db 'Введите a: ',0h
5 msg3: db 'Функция равна: ',0h
6
7 section .bss
8 x: resb 10
9 a: resb 10
10 ans: resb 10
11
12 section .text
13 global _start
14 _start:
15     mov eax,msg1
16     call sprint
17
18     mov ecx,x
19     mov edx,10
20     call sread
21
22     mov eax,x
23     call atoi
24     mov [x],eax
25
26     mov eax,msg2
27     call sprint
28
29     mov ecx,a
30     mov edx,10
31     call sread
```

Рис. 4.5: Программа



```
32
33     mov eax, a
34     call atoi
35     mov [a],eax
36
37     mov eax,[x]
38
39     cmp eax,0
40
41     je func1
42     jne func2
43
44 func1:
45     mov eax, x
46     sub| eax,[2]
47     mov [ans], eax
48     jmp final
49
50 func2:
51     mov eax, 3
52     mov ebx, [a]
53     mul ebx
54     mov [ans], eax
55     jmp final
56 final:
57     mov eax, msg3
58     call sprint
59     mov eax, [ans]
60     call iprintLF
61     call quit
62
```

Рис. 4.6: Программа



```
[emmednikova@fedora lab08]$ gedit sm2.asm
[emmednikova@fedora lab08]$ nasm -f elf sm2.asm
[emmednikova@fedora lab08]$ ld -m elf_i386 -o sm2 sm2.o
[emmednikova@fedora lab08]$ ./sm2
Введите x: 3
Введите a: 0
Функция равна: 0
[emmednikova@fedora lab08]$ ./sm2
Введите x: 1
Введите a: 2
Функция равна: 6
```

Рис. 4.7: Результат

5 Выводы

Изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.