

# **Отчёт по лабораторной работе №7**

**дисциплина: архитектура компьютеров**

Ведьмина Александра Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>15</b>
<b>6</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

4.1	Создание каталога lab07 и файла lab7-1.asm . . . . .	8
4.2	Ввод программы в файл lab7-1.asm . . . . .	9
4.3	Запуск lab7-1 . . . . .	9
4.4	Изменение программы lab7-1.asm . . . . .	10
4.5	Запуск lab7-1 . . . . .	10
4.6	Изменение программы lab7-1.asm . . . . .	11
4.7	Запуск изменённого lab7-1 . . . . .	11
4.8	Ввод программы в lab7-2.asm . . . . .	12
4.9	Запуск lab7-2 при B=5 . . . . .	12
4.10	Запуск lab7-2 при B=6789 . . . . .	12
4.11	Создание lab7-2.lst . . . . .	13
4.12	Просмотр lab7-2.lst . . . . .	13
4.13	Просмотр lab7-2.lst с ошибкой . . . . .	14
5.1	Ввод программы в sumrub-1.asm, 1 часть . . . . .	15
5.2	Ввод программы в sumrub-1.asm, 2 часть . . . . .	16
5.3	Запуск sumrub-1 . . . . .	16
5.4	Создание файла sumrub-2.asm . . . . .	17
5.5	Ввод программы в sumrub-2.asm, 1 часть . . . . .	18
5.6	Ввод программы в sumrub-2.asm, 2 часть . . . . .	19
5.7	Запуск sumrub-2 . . . . .	19

## 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. Ознакомиться с командами условного и безусловного перехода
2. Изучить структуру файла листинга
3. Рассмотреть программы, использующие разные переходы в `asm`
4. Выполнить задания для самостоятельной работы

### 3 Теоретическое введение

Безусловный переход выполняется инструкцией `jmp`. Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Для условных переходов задаются условия: например, мнемокод `JE` обозначает  $a=b$ , значение флага  $ZF=1$  и осуществляет переход, если операнды равны.

Листинг - это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы. В его структуру входят номер строки (это номер строки файла листинга), адрес (смещение машинного кода от начала текущего сегмента), машинный код (ассемблированная исходная строка в виде шестнадцатеричной последовательности), исходный текст программы (строка исходной программы вместе с комментариями).

## 4 Выполнение лабораторной работы

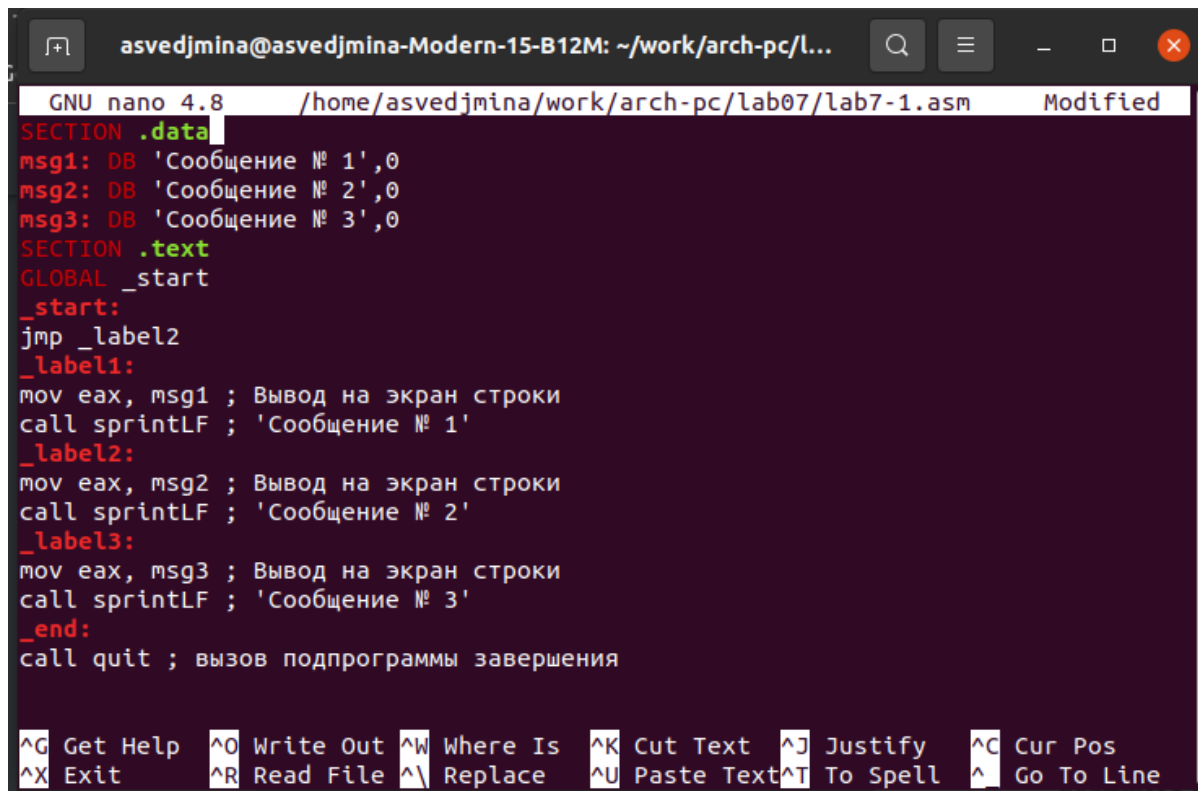
Создаю каталог lab07 и файл lab7-1.asm в нём.

```
asvedjmina@asvedjmina-Modern-15-B12M:~$ mkdir ~/work/arch-pc/lab07
asvedjmina@asvedjmina-Modern-15-B12M:~$ cd ~/work/arch-pc/lab07
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ touch lab7-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание каталога lab07 и файла lab7-1.asm

Ввожу в файл lab7-1.asm программу, использующую инструкцию jmp.



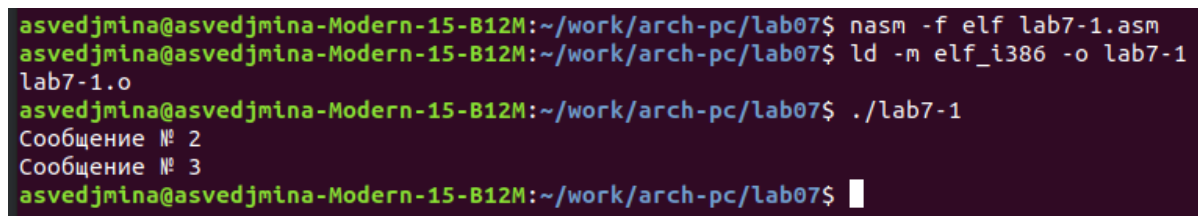


```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/lab7-1.asm Modified
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Рис. 4.2: Ввод программы в файл lab7-1.asm

Создаю исполняемый файл и запускаю его.



```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1
lab7-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 4.3: Запуск lab7-1

Изменяю программу так, чтобы выводилось сообщение 2, а потом сообщение 1.

```
asvedjmina@asvedjmina-Modern-15-B12M: ~/work/arch-pc/lab07
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/lab7-
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'
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'
_end:█
```

Рис. 4.4: Изменение программы lab7-1.asm

Создаю исполняемый файл и запускаю его.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1
lab7-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ █
```

Рис. 4.5: Запуск lab7-1

Снова изменяю текст программы таким образом, что выводилось сначала сообщение 3, затем сообщение 2 и сообщение 1.

```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/lab7-1.asm
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. 4.6: Изменение программы lab7-1.asm

Создаю исполняемый файл и проверяю его работу.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Запуск изменённого lab7-1

Создаю файл lab7-2.asm и ввожу в него текст программы, которая определяет и выводит на экран наибольшую из переменных А, В, С.

```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/lab7-2.asm Modified
%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

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text    ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Рис. 4.8: Ввод программы в lab7-2.asm

Создаю исполняемый файл и запускаю его, проверяю его работу при разных значениях B.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2
lab7-2.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 4.9: Запуск lab7-2 при B=5

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 6789
Наибольшее число: 6789
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 4.10: Запуск lab7-2 при B=6789

Создаю файл листинга для программы из файла lab7-2.asm.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst
lab7-2.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 4.11: Создание lab7-2.lst

Открываю файл lab7-2.lst с помощью mcedit и изучаю содержимое.

```
/home/asve~ab7-2.lst [----] 0 L:[ 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*][X]
1 %include 'in_out.asm'
2 <1> ;----- slen -----
3 <1> ; Функция вычисления длины сообщения
4 <1> slen:.....
5 00000000 53 <1> push ebx.....
6 00000001 89C3 <1> mov ebx, eax.....
7 <1>.....
8 <1> nextchar:.....
9 00000003 803800 <1> cmp byte [eax], 0...
10 00000006 7403 <1> jz finished.....
11 00000008 40 <1> inc eax.....
12 00000009 EBF8 <1> jmp nextchar.....
13 <1>.....
14 <1> finished:
15 0000000B 29D8 <1> sub eax, ebx
16 0000000D 5B <1> pop ebx.....
17 0000000E C3 <1> ret.....
18 <1>
19 <1> ;----- sprint -----
20 <1> ; Функция печати сообщения
21 <1> ; входные данные: mov eax, <message>
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 4.12: Просмотр lab7-2.lst

Строка lab7-2.lst 224: 1. 47 - номер строки в программе 2. 0000015E - адрес 3. A1[00000000] - машинный код 4. mov eax, [max] - перемещение максимального значения среди переменных в eax

Строка lab7-2.lst 225: 1. 48 - номер строки в программе 2. 00000163 - адрес 3. E81EFFFFFF - машинный код 4. call iprintLF - вывод переменной с наибольшим значением

Строка lab7-2.lst 226: 1. 49 - номер строки в программе 2. 00000168 - адрес 3. E86EFFFFFF - машинный код 4. call quit - выход из программы

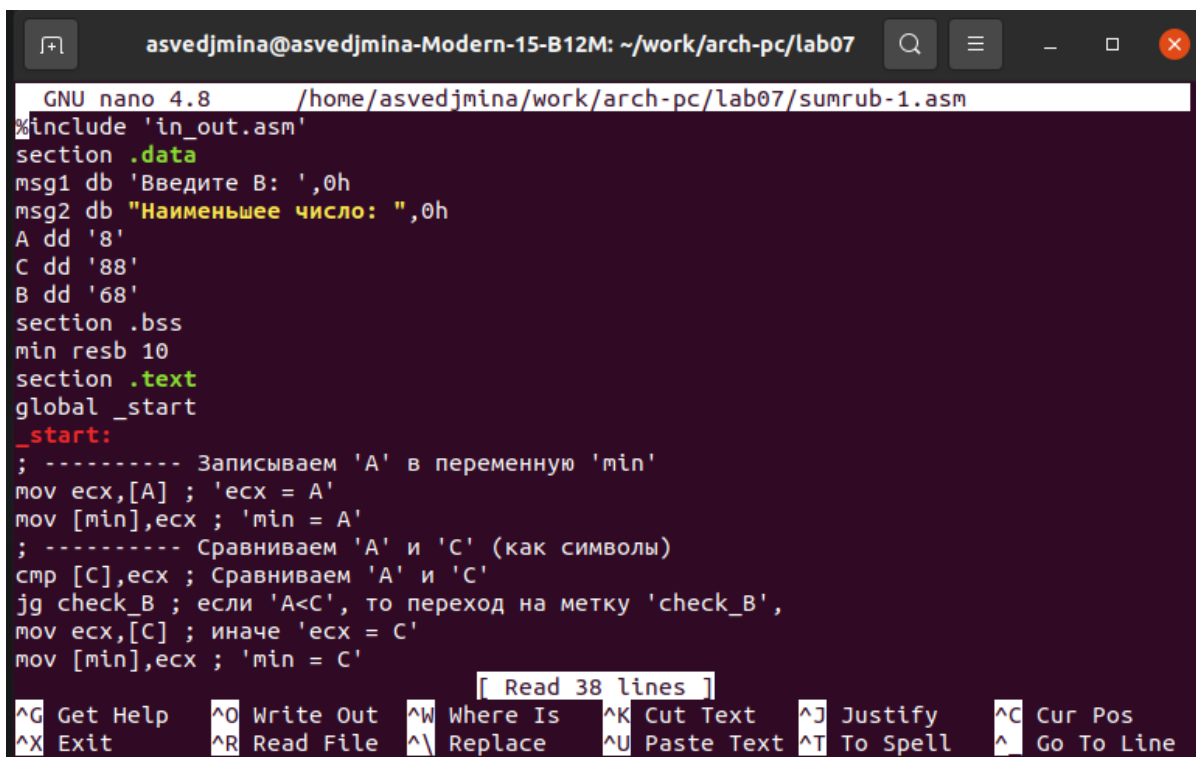
В файле с программой lab7-2.asm в инструкции с двумя операндами удаляю один операнд, после чего выполняю трансляцию с получением файла листинга. Открываю полученный файл и вижу, что звёздочками там отображается место, где в коде программы допущена ошибка и приводится её пояснение.

```
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF      call sprint
16                               ; ----- Ввод 'B'
17                               mov ecx
17      *****
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20                               ; ----- Преобразование 'B' из символа в
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E836FFFFFF      call quit
```

Рис. 4.13: Просмотр lab7-2.lst с ошибкой

## 5 Выполнение заданий для самостоятельной работы

1. Создаю файл sunrub-1.asm и ввожу в него текст программы нахождения наименьшей из 3 целочисленных переменных. Мой вариант, полученный в предыдущей лабораторной - 4, поэтому я буду использовать следующие значения переменных:  $a = 8$ ,  $b = 88$ ,  $c = 68$ .

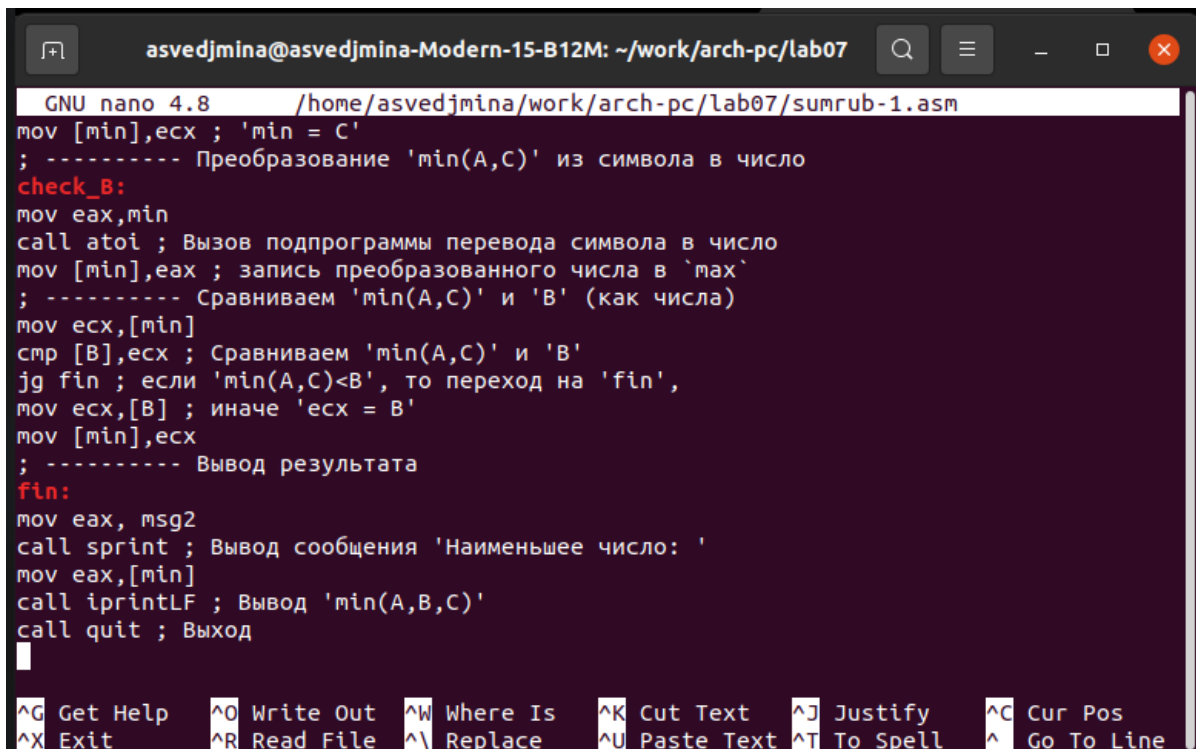


```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/sumrub-1.asm
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '8'
C dd '88'
B dd '68'
section .bss
min resb 10
section .text
global _start
_start:
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp [C],ecx ; Сравниваем 'A' и 'C'
jg check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
```

[ Read 38 lines ]

<b>^G</b> Get Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut Text	<b>^J</b> Justify	<b>^C</b> Cur Pos
<b>^X</b> Exit	<b>^R</b> Read File	<b>^_</b> Replace	<b>^U</b> Paste Text	<b>^T</b> To Spell	<b>^_</b> Go To Line

Рис. 5.1: Ввод программы в sumrub-1.asm, 1 часть

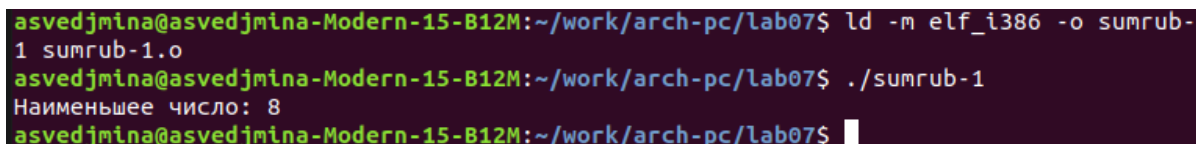


```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab07/sumrub-1.asm
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp [B],ecx ; Сравниваем 'min(A,C)' и 'B'
jg fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Рис. 5.2: Ввод программы в sumrub-1.asm, 2 часть

Создаю исполняемый файл и проверяю его работу.



```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o sumrub-1 sumrub-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./sumrub-1
Наименьшее число: 8
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 5.3: Запуск sumrub-1

2. Создаю файл sumrub-2.asm и ввожу в него программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции. Я буду реализовывать функцию вида  $2x+a$  при  $a$ , не равном нулю, и  $2x+1$  при  $a$ , равном нулю.



```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ touch sumrub-2.asm  
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$
```

Рис. 5.4: Создание файла sumrub-2.asm

```

%include      'in_out.asm'
SECTION .data
msg1: DB 'Введите значение переменной x: ', 0h
msg2: DB 'Введите значение переменной a: ', 0h
rem: DB 'Результат: ', 0h

SECTION .bss
x:      RESB 10
a:      RESB 10

SECTION .text
GLOBAL _start
_start:

;-----ВВОД x
mov eax, msg1
call sprint
mov ecx, x
mov edx, 10
call sread
;-----x в число
mov eax, x
call atoi
mov [x], eax
;-----ВВОД a
mov eax, msg2
call sprint
mov ecx, a
mov edx, 10
call sread
;-----a в число
mov eax, a
call atoi
mov [a], eax

```

Рис. 5.5: Ввод программы в sumrub-2.asm, 1 часть

```

;----- а равно/не равно нулю
mov edx, [a]
mov ebx, 0
cmp edx, ebx
jne _negav
jmp _rav
;-----
_negav:
mov eax, [x]
mov ebx, 2
mul ebx
mov ebx, [a]
add eax, ebx
call iprintLF
call quit
;-----
_rav:
mov eax, [x]
mov ebx, 2
mul ebx
add eax, 1
call iprintLF
call quit

```

Рис. 5.6: Ввод программы в sumrub-2.asm, 2 часть

Создаю исполняемый файл и проверяю его работу для значений x и a, указанных в задании.

```

asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ nasm -f elf sumrub-2.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ld -m elf_i386 -o sumrub-2 sumrub-2.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./sumrub-2
Введите значение переменной x: 3
Введите значение переменной a: 0
7
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ ./sumrub-2
Введите значение переменной x: 3
Введите значение переменной a: 2
8
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab07$ █

```

Рис. 5.7: Запуск sumrub-2

## **6 Выводы**

В ходе лабораторной работы я изучила команды условного и безусловного перехода, освоила написание программ с использованием переходов, а также познакомилась со структурой файла листинга.