

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютеров

Лазарев Даниил Михайлович

Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выполнение самостоятельной работы	14
5	Выводы	17

Список иллюстраций

3.1	Создание файла в каталоге	6
3.2	Код программы в файле	7
3.3	Преобразование в исполняемый файл	7
3.4	Измененный код программы	8
3.5	Преобразование измененного файла	9
3.6	Изменения исходного текста	10
3.7	Преобразование файла	11
3.8	Преобразование файла в исполняемый	12
3.9	Получение листинга и текстовый редактор	12
3.10	Пояснение работы строк	13
4.1	Код программы для определения наименьшей переменной	14
4.2	Код программы	15
4.3	Проверка правильности	16

1 Цель работы

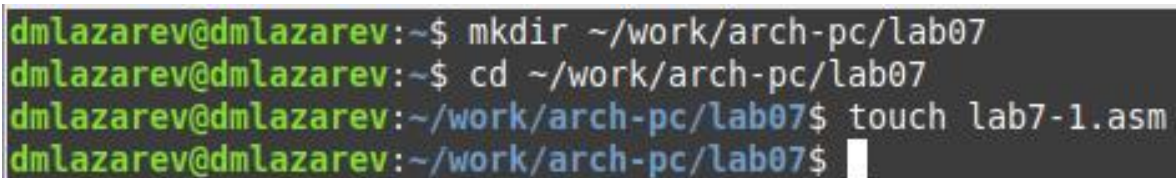
Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

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.

3 Выполнение лабораторной работы

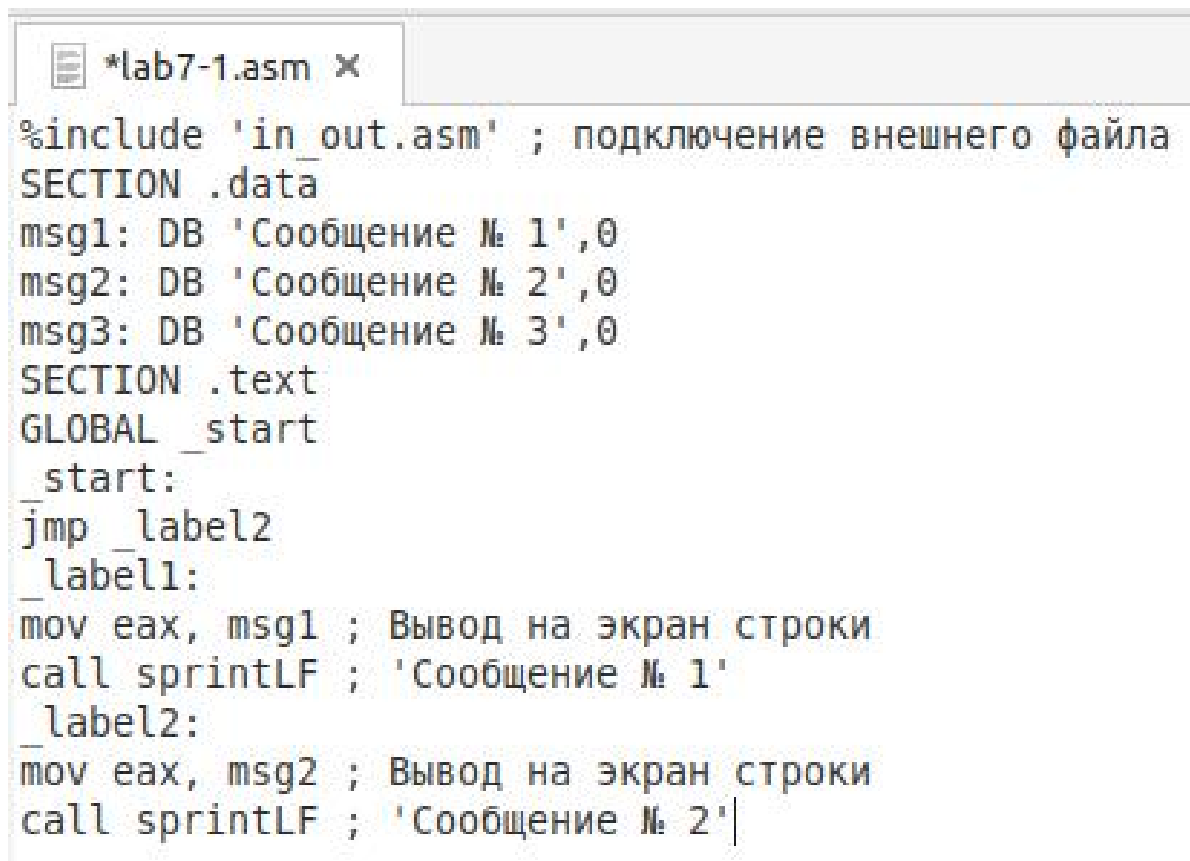
Создадим каталог для программ лаб. работы н.7, перейдем в него и создадим файл “lab7-1.asm” (рис. 3.1)



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

Рис. 3.1: Создание файла в каталоге

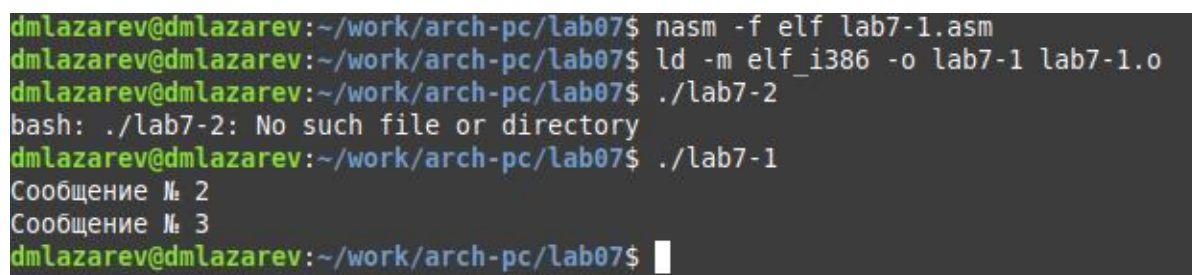
Введем в созданный файл текст программы из предложенного нам листинга 7.1(рис. 3.2)



```
*lab7-1.asm X
%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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
```

Рис. 3.2: Код программы в файле

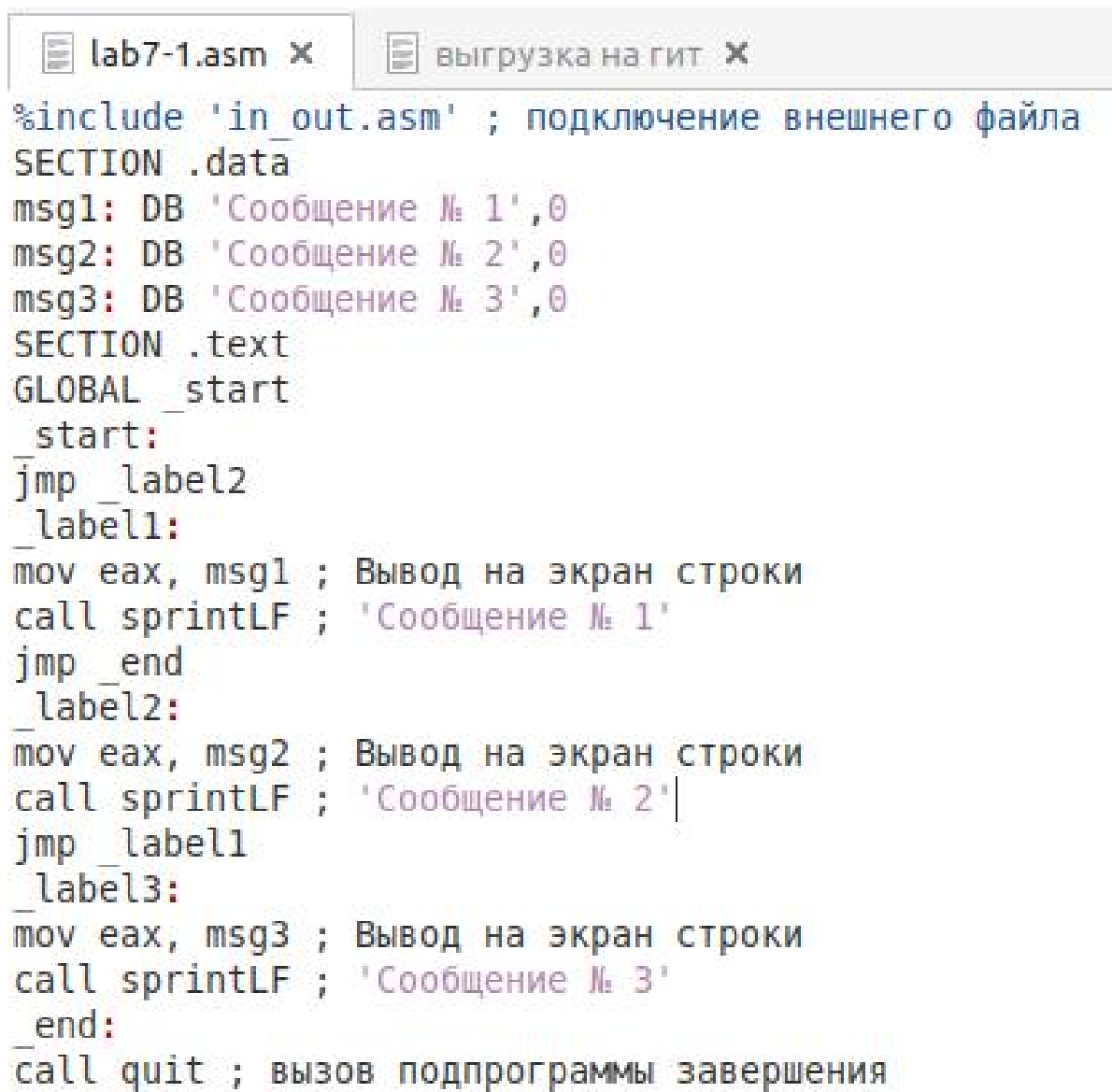
Создадим исполняемый файл и запустим его, предварительно скопировав из предыдущей лаб. работы файл “in_out.asm” для корректной работы (рис. 3.3)



```
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-2
bash: ./lab7-2: No such file or directory
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
dmlazarev@dmlazarev:~/work/arch-pc/lab07$
```

Рис. 3.3: Преобразование в исполняемый файл

Далее изменим текст в соответствии с листингом 7.2. (рис. 3.5)



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

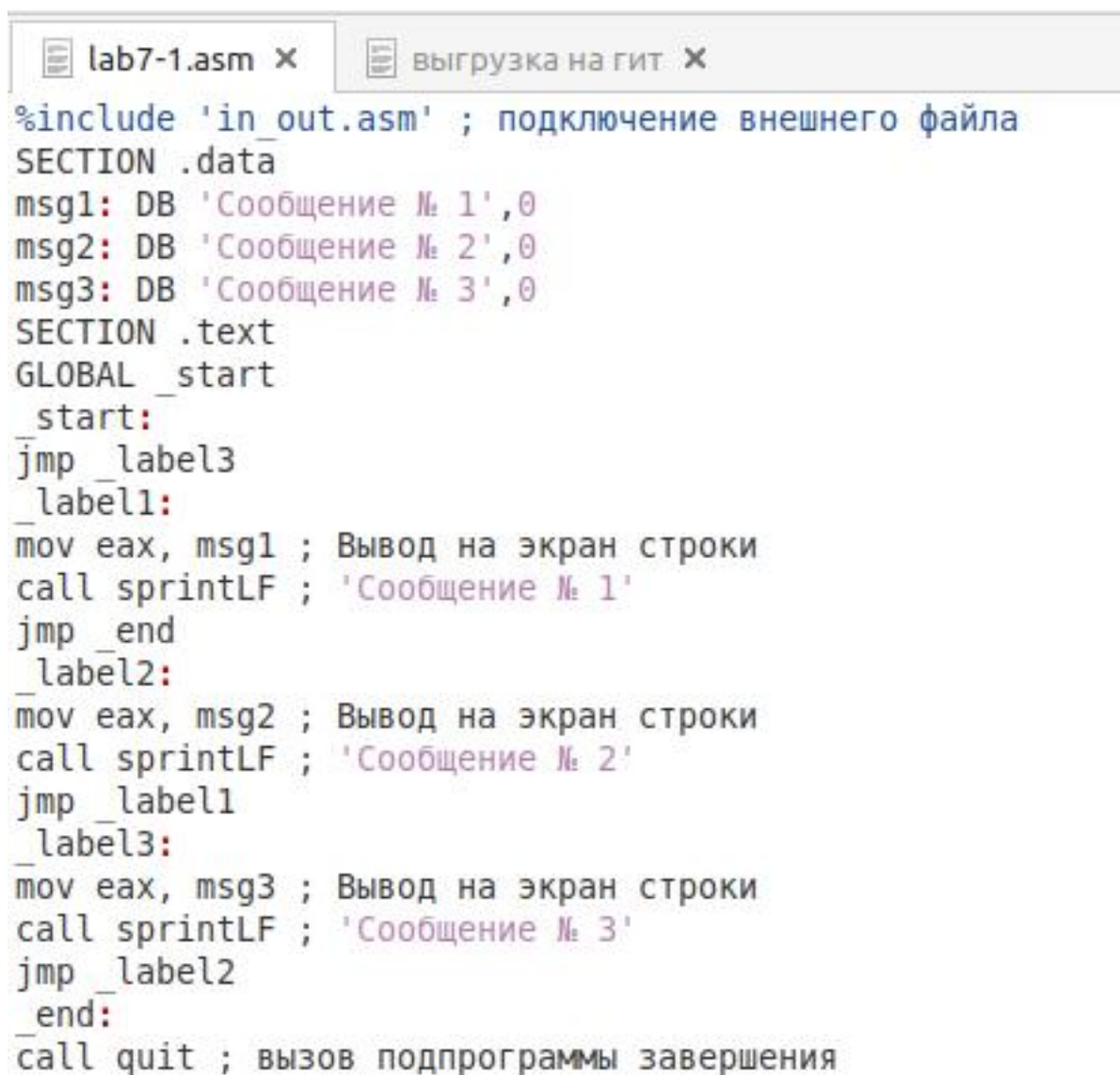
Рис. 3.4: Измененный код программы

Преобразуем в исполняемый файл и проверим правильность выполнения. (рис. ??)


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

Рис. 3.5: Преобразование измененного файла

Изменим текст листинга так, чтобы выводились сообщения в порядке убывания.
(рис. 3.6)



```
%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 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. 3.6: Изменения исходного текста

Преобразуем файл в исполняемый и проверим правильность выполнения. (рис. 3.7)

```

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

```

Рис. 3.7: Преобразование файла

Создадим файл “lab7-2.asm” и вставим в него предложенный нам листинг 7.2 (рис. ??; рис. ??)

```

dmlazarev@dmlazarev:~/work/arch-pc/lab07$ touch lab7-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab07$
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1

```

Преобразуем файл “lab7-2.asm” в исполняемый и проверим правильность выполнения. (рис. 3.8)

```

dmlazarev@dmlazarev:~/work/arch-pc/lab07$ touch lab7-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 14
Наибольшее число: 50
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 3
Наибольшее число: 50
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 23
Наибольшее число: 50
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 69
Наибольшее число: 69

```

Рис. 3.8: Преобразование файла в исполняемый

Получим листинг преобразованного нами файла и откроем через текстовый редактор “gedit”. (рис. 3.9)

```

dmlazarev@dmlazarev:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ gedit lab7-2.lst

```

Line	Address	Disassembly
1	1	%include 'in_out.asm'
2	2	<1> ;----- slen -----
3	3	<1> ; Функция вычисления длины сообщения
4	4	<1> slen:
5	5 00000000 53	<1> push ebx
6	6 00000001 89C3	<1> mov ebx, eax
7	7	<1>
8	8	<1> nextchar:
9	9 00000003 803800	<1> cmp byte [eax], 0

Рис. 3.9: Получение листинга и текстовый редактор

Выберем три случайные строки из файла и поясним что каждая из них значит. (рис. 3.10)

1. Строка №33: `mov ecx, eax` - это инструкция перемещения значения из регистра `eax` в регистр `ecx`. Эта инструкция использует код операции `89C1` и ее адрес в памяти начинается с `0000001B`.
2. Строка №52: `push eax` - это инструкция для помещения значения из регистра `eax` на вершину стека. Эта инструкция использует код операции `50` и ее адрес в памяти начинается с `00000038`.
3. Строка №100: `jnz printLoop` - это инструкция условного перехода, которая переходит к метке `printLoop` если флаг неравенства установлен. Эта инструкция использует код операции `75F2` и ее адрес в памяти начинается с `0000007F`.

Рис. 3.10: Пояснение работы строк

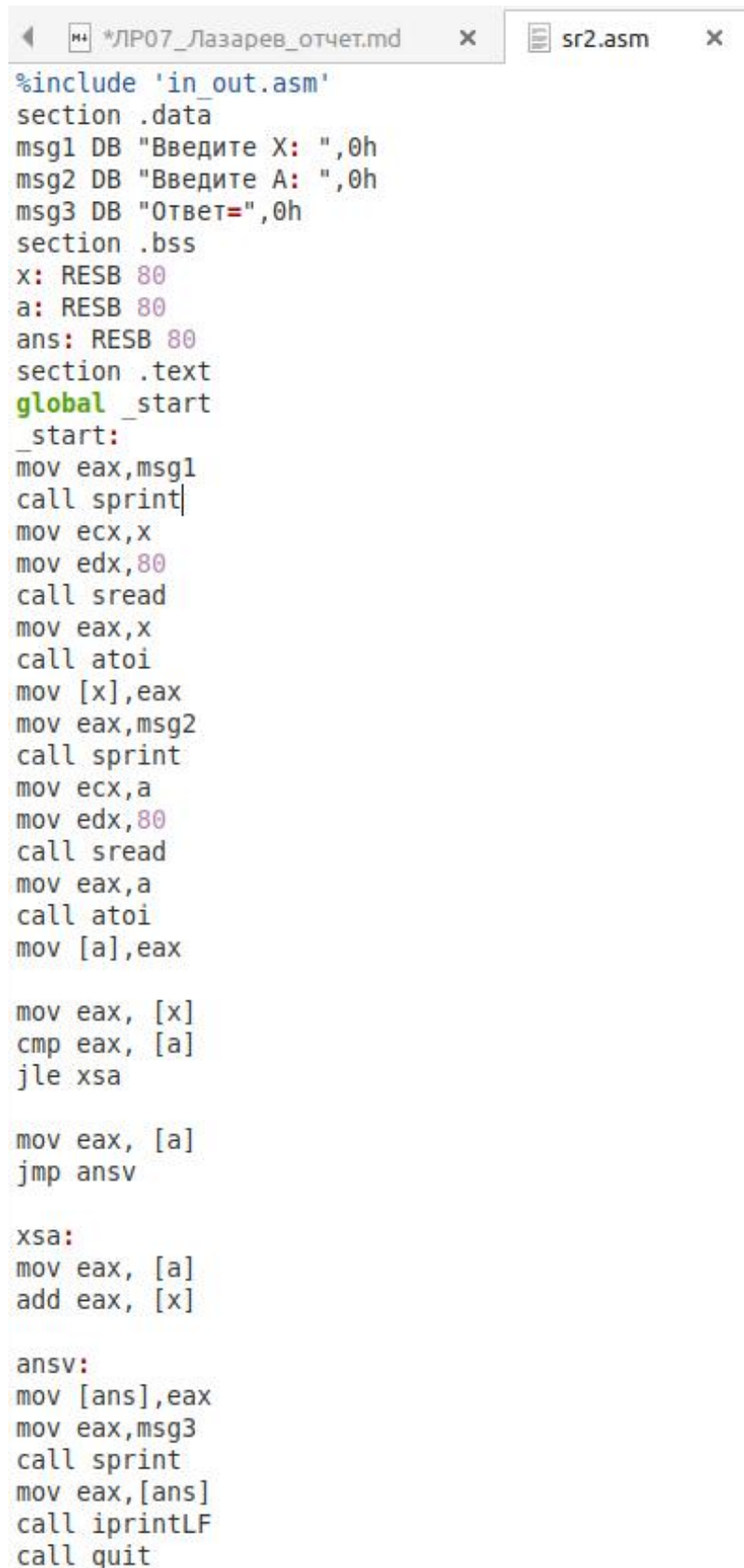
4 Выполнение самостоятельной работы

Основываясь на результате файла “variant.asm” из лаб. работы н.6 выберем из таблицы 7.5 9 номер варианта. Самостоятельно напишем код программы, который будет выбирать наименьшее число из 3 переменных. (рис. 4.1)

Код программы для определения наименьшей переменной

Рис. 4.1: Код программы для определения наименьшей переменной

Так же выберем из таблицы 7.6 9 вариант и напишем код, в котором будут происходить вычисления относительно системы уравнений. (рис. 4.2; рис. 4.3)



```
%include 'in_out.asm'
section .data
msg1 DB "Введите X: ",0h
msg2 DB "Введите A: ",0h
msg3 DB "Ответ=",0h
section .bss
x: RESB 80
a: RESB 80
ans: RESB 80
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,msg2
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
mov [a],eax

mov eax, [x]
cmp eax, [a]
jle xsa

mov eax, [a]
jmp ansv

xsa:
mov eax, [a]
add eax, [x]

ansv:
mov [ans],eax
mov eax,msg3
call sprint
mov eax,[ans]
call iprintLF
call quit
```

Рис. 4.2: Код программы

```
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./sr2
Введите X: 5
Введите A: 7
Ответ=12
dmlazarev@dmlazarev:~/work/arch-pc/lab07$ ./sr2
Введите X: 6
Введите A: 4
Ответ=4
dmlazarev@dmlazarev:~/work/arch-pc/lab07$
```

Рис. 4.3: Проверка правильности

5 Выводы

В ходе лабораторной работы мы освоили арифметические операции на языке NASM.