

Отчет по лабораторной работе No 7

Дисциплины: Архитектура компьютера

Ракутуманандзара Цантамписедрана Сарубиди

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	рис 1	8
4.2	рис 2	8
4.3	рис 3	8
4.4	рис 4	9
4.5	рис 5	9
4.6	рис 6	10
4.7	рис 7	10
4.8	рис 8	11
4.9	рис 9	11
4.10	рис 10	11
4.11	рис 11	12
4.12	рис 12	12
4.13	рис 13	13
4.14	рис 14	13
4.15	рис 15	14
4.16	рис 16	14
4.17	рис 16	14
4.18	рис 18	15
4.19	рис 19	15

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного перехода. Также приобрести навыки написания программ с использованием переходов и понимания назначения и структуры листинга файлов.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной 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

1. Реализация переходов в NASM

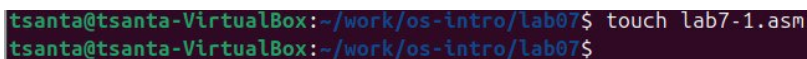
Я создам каталог для программ лабораторных работ 7 под названием lab07 в каталоге ~/work/arch-rc с помощью команды mkdir(рис 1)



```
tsanta@tsanta-VirtualBox:~$ mkdir ~/work/os-intro/lab07
```

Рис. 4.1: рис 1

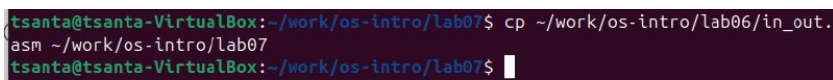
В созданном каталоге я создам файл с именем lab7-1.asm с помощью сенсорной команды(рис 2).



```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ touch lab7-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.2: рис 2

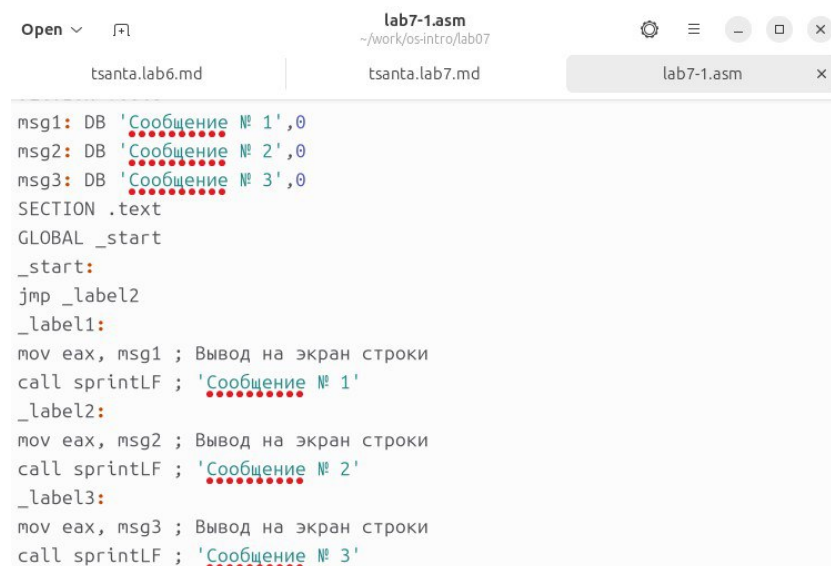
Я скопирую файл in_out.asm в текущий каталог с помощью команды cp, потому что буду использовать его в программах(рис 3)



```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ cp ~/work/os-intro/lab06/in_out.
asm ~/work/os-intro/lab07
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.3: рис 3

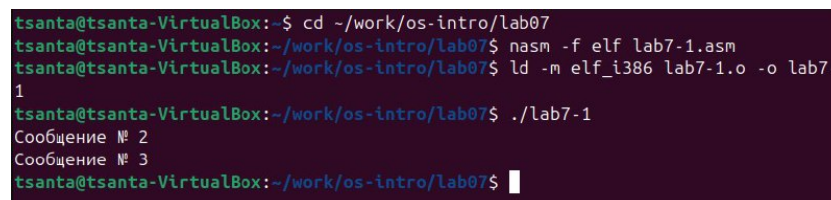
Я открою и введу заданный текст программы в созданный мною файл. Программа использует инструкцию jmp. Инструкция jmp в NASM используется для реализации безусловных переходов(рис 4)



```
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'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
```

Рис. 4.4: рис 4

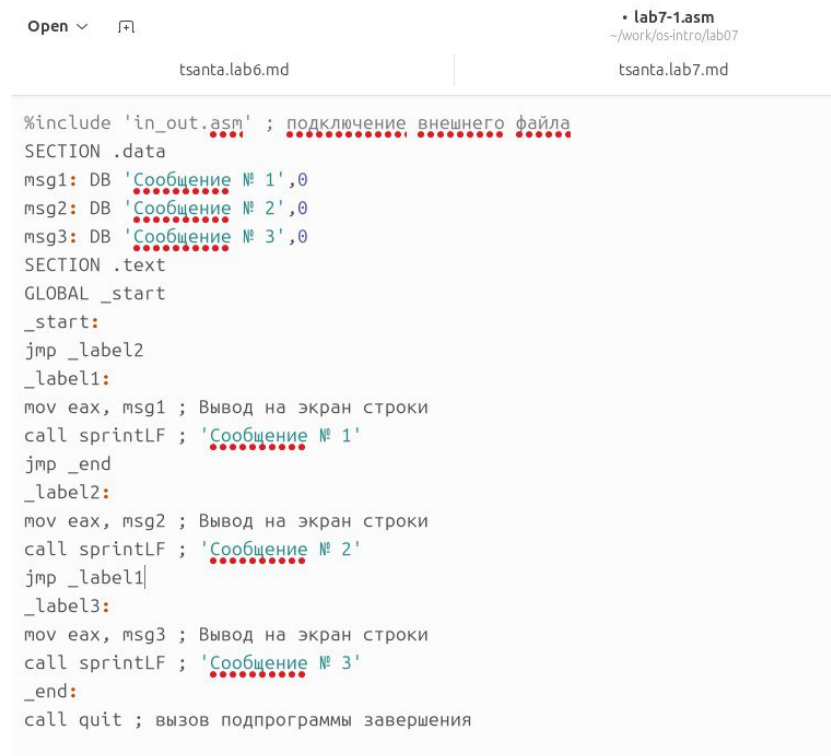
Я создам исполняемый файл и запущу его(рис 5).Как мы видим, использование инструкции `jmp _label2` меняет порядок выполнения инструкций и позволяет выполнять инструкции, начиная с метки `_label2`, пропуская вывод первого сообщения.



```
tsanta@tsanta-VirtualBox:~$ cd ~/work/os-intro/lab07
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf lab7-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.5: рис 5

Инструкция `jmp` позволяет прыгать не только вперед, но и назад.Я изменю программу, добавив «`jmp _label1`» после `label2` и «`jmp _end`» после `label1`, чтобы она сначала отображала «Сообщение 2», затем «Сообщение 1» и завершала работу без отображения «Сообщения 3».(рис 6)

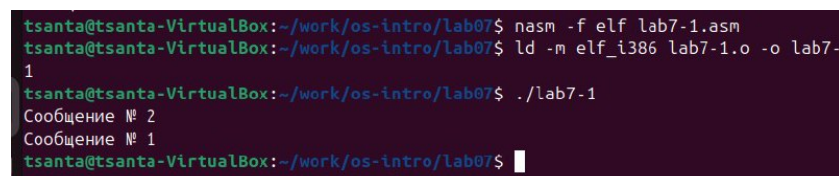


```
Open  tsanta.lab6.md  lab7-1.asm  tsanta.lab7.md

%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. 4.6: рис 6

Я создам исполняемый файл, запущу его и проверю, работает ли программа корректно(рис 7)



```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf lab7-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.7: рис 7

Теперь я изменю программу, добавив «jmp_label3» перед label1, «jmp_label1» после label2, «jmp_label2» после label3 и «jmp_end» после label1, чтобы она сначала отображала «Сообщение 3», затем «Сообщение 2» и, наконец, «Сообщение 1»(рис 8)

```

Open  tsanta.lab6.md  lab7-1.asm  tsanta.lab7.md
~/work/os-intro/lab07

%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. 4.8: рис 8

Я создам исполняемый файл, запущу его и проверю, работает ли программа корректно(рис 9)

```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf lab7-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$

```

Рис. 4.9: рис 9

Я создам новый файл с именем lab7-2.asm в каталоге ~/work/arch-pc/lab07(рис 10)

```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ touch lab7-2.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$

```

Рис. 4.10: рис 10

Я скопирую данную текстовую программу в только что созданный файл. При написании программ необходимо использовать условные переходы, т.е. переход должен происходить при выполнении какого-либо условия. Данная программа определяет и отображает наибольшую из 3-х целочисленных переменных: A, B и C. Значения A и C указаны в программе, значение B вводится с клавиатуры (рис 11)



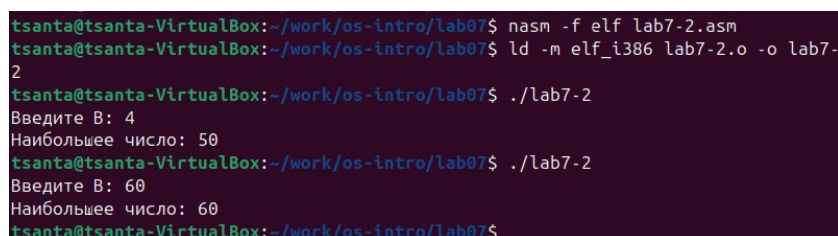
```

%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
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'

```

Рис. 4.11: рис 11

Я создам исполняемый файл и протестирую его на разных значениях B



```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf lab7-2.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-2
Введите B: 4
Наибольшее число: 50
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$

```

Рис. 4.12: рис 12

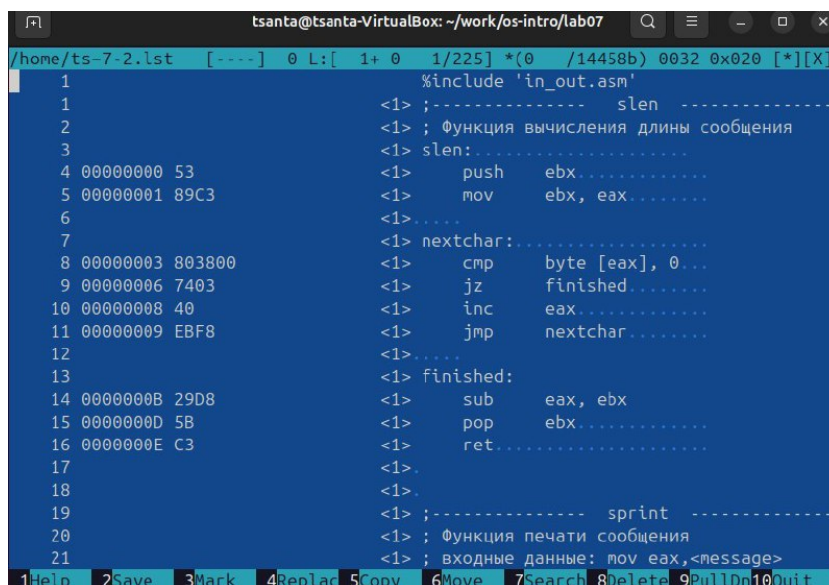
2. Изучение структуры файлы листинга

Указав ключ -l и имя файла листинга в командной строке. Файл листинга программы я создам из файла lab7-2.asm(рис 13)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.13: рис 13

Теперь я открою файл листинга lab7-2.lst с помощью текстового редактора(рис 14)



```
/home/ts-7-2.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 00000000 53                         <1> slen:-----
5 00000001 89C3                       <1> push    ebx
6                                     <1> mov     ebx, eax
7                                     <1> nextchar:-----
8 00000003 803800                     <1> cmp     byte [eax], 0
9 00000006 7403                       <1> jz      finished
10 00000008 40                        <1> inc     eax
11 00000009 EBF8                     <1> jmp     nextchar
12                                     <1> finished:-----
13                                     <1> sub     eax, ebx
14 0000000B 29D8                     <1> pop     ebx
15 0000000D 5B                        <1> ret
16 0000000E C3                       <1>
17                                     <1>
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 9PullDn10Quit
```

Рис. 4.14: рис 14

строка 19 «вызов atoi» меняет В с арифметического символа на число
строка 174 «msg2 db «Наибольшее число:»,0h» отображает текст «Наибольшее число:» на экране

строка 173 «msg1 db «Введите В:»,0h» отображает текст «Введите В:» на экране
Я открою файл lab7-2.asm и удалю один из операндов, затем выполню широко-вещательную рассылку, чтобы получить файл листинга(рис 15 и 16)

```

/home/ts-b7-2.asm  [-M--]  9 L:[ 1+16 17/ 45] *(262 /1496b) 0010 0x00A [*][X]
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
C dd '50'
section .bss
max 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 ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

```

Рис. 4.15: рис 15

```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf -l lab7-2.lst lab7-2
.asm
lab7-2.asm:20: error: symbol 'A' not defined
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$

```

Рис. 4.16: рис 16

При создании файла листинга выдала ошибку, так как в файле lab7-2.asm программа неверна

3. Выполнение заданий для самостоятельной работы

С помощью touch команды я создам новый файл lab7-3.asm(рис 17)

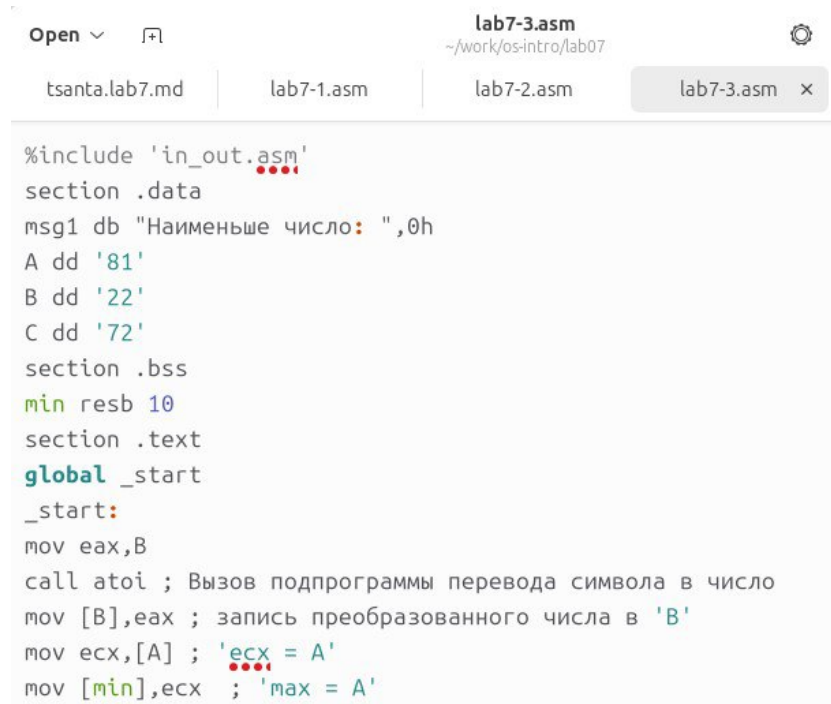
```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ touch lab7-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$

```

Рис. 4.17: рис 16

В созданном мной файле я напишу программу, которая будет находить минимальное значение среди трех чисел a, b и c. Я получу значения a, b и c из варианта, который я получил при выполнении лабораторной работы 6(рис 18)



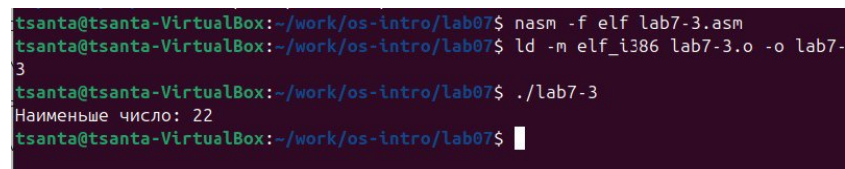
```
lab7-3.asm
~/work/os-intro/lab07

tsanta.lab7.md | lab7-1.asm | lab7-2.asm | lab7-3.asm x

#include 'in_out.asm'
section .data
msg1 db "Наименьше число: ",0h
A dd '81'
B dd '22'
C dd '72'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'max = A'
```

Рис. 4.18: рис 18

Теперь я создам исполняемый файл и запущу его, чтобы посмотреть, даст ли он правильный ответ(рис 19)



```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ nasm -f elf lab7-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$ ./lab7-3
Наименьше число: 22
tsanta@tsanta-VirtualBox:~/work/os-intro/lab07$
```

Рис. 4.19: рис 19

```
%include 'in_out.asm'
section .data
msg1 db "Наименьше число: ",0h
A dd '81'
B dd '22'
C dd '72'
section .bss
min resb 10
```

```

section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'max = A'
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jlt check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'max = C'
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'max'
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jlt fin ; если 'max(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
fin:
mov eax, msg1
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```


5 Выводы

Выполняя эту лабораторную работу, я узнал об условных и безусловных командах перехода. Также приобрел навыки написания программ с использованием переходов и понял назначение и структуру листингов файлов.

Список литературы

Архитектура ЭВМ