

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Юсуфов Джабар Артикович

Содержание

1	Цель работы	1
2	Задание	1
3	Выполнение лабораторной работы	1
3.1	Реализация переходов в NASM.	1
3.2	Изучение структуры файла листинга	6
3.3	Задания для самостоятельной работы.....	8
4	Выводы	11

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. Реализация переходов в NASM.
2. Изучение структуры файла листинга.
3. Задания для самостоятельной работы.

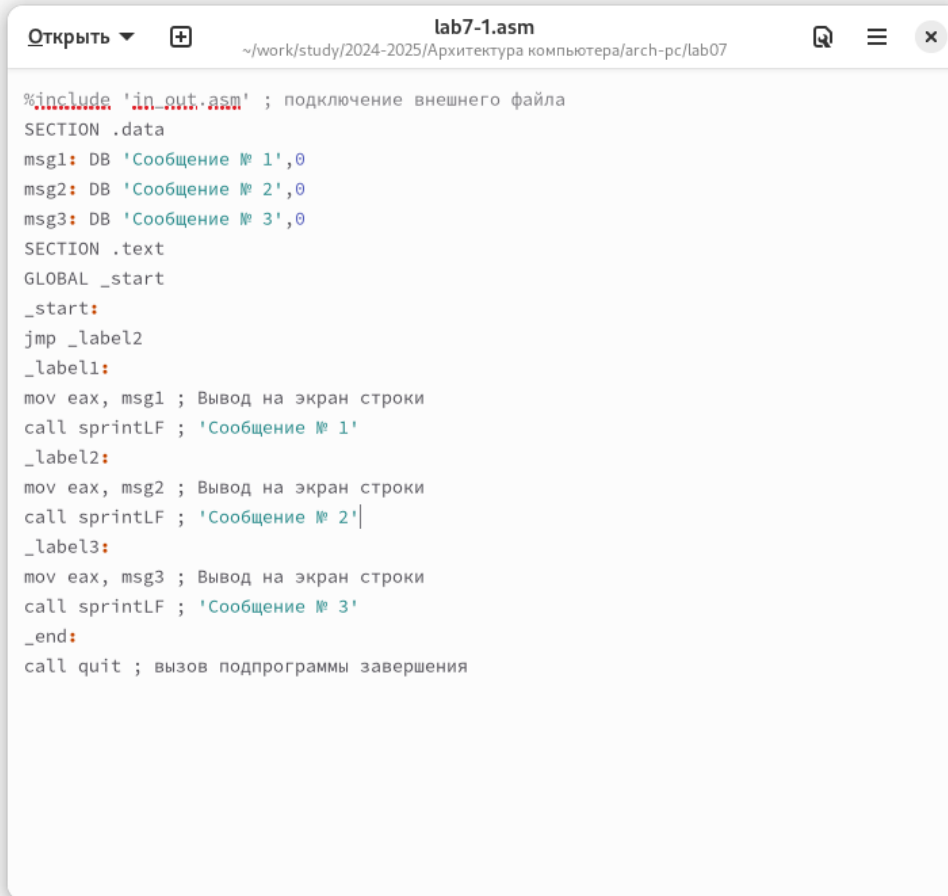
3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM.

Создаю каталог для программ лабораторной работы №7.

```
neroun@fedora:/$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab07
neroun@fedora:/$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab07
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Копирую код из листинга в файл будущей программы.

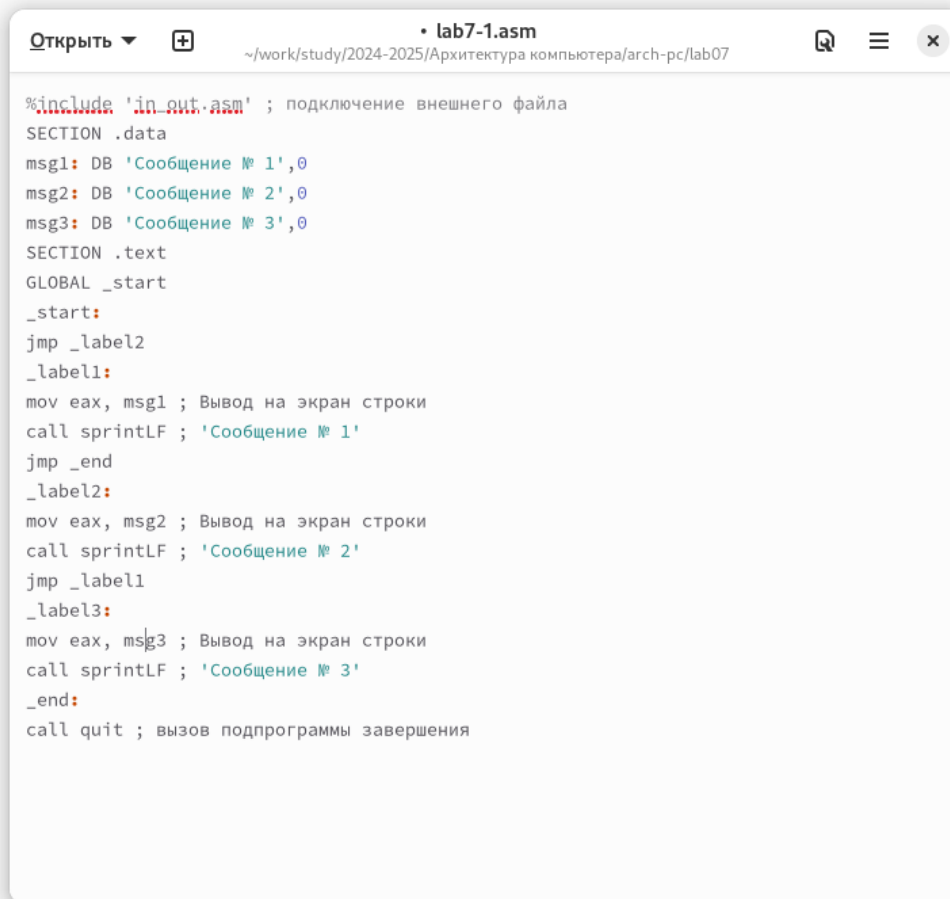


```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций.

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

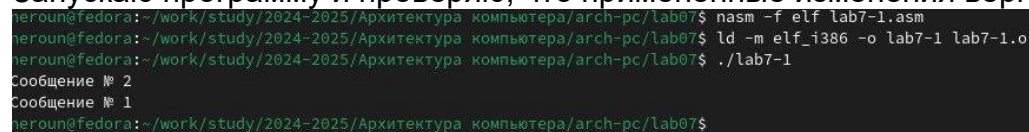
Изменяю программу таким образом, чтобы поменялся порядок выполнения функций.



```
Открыть + lab7-1.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07

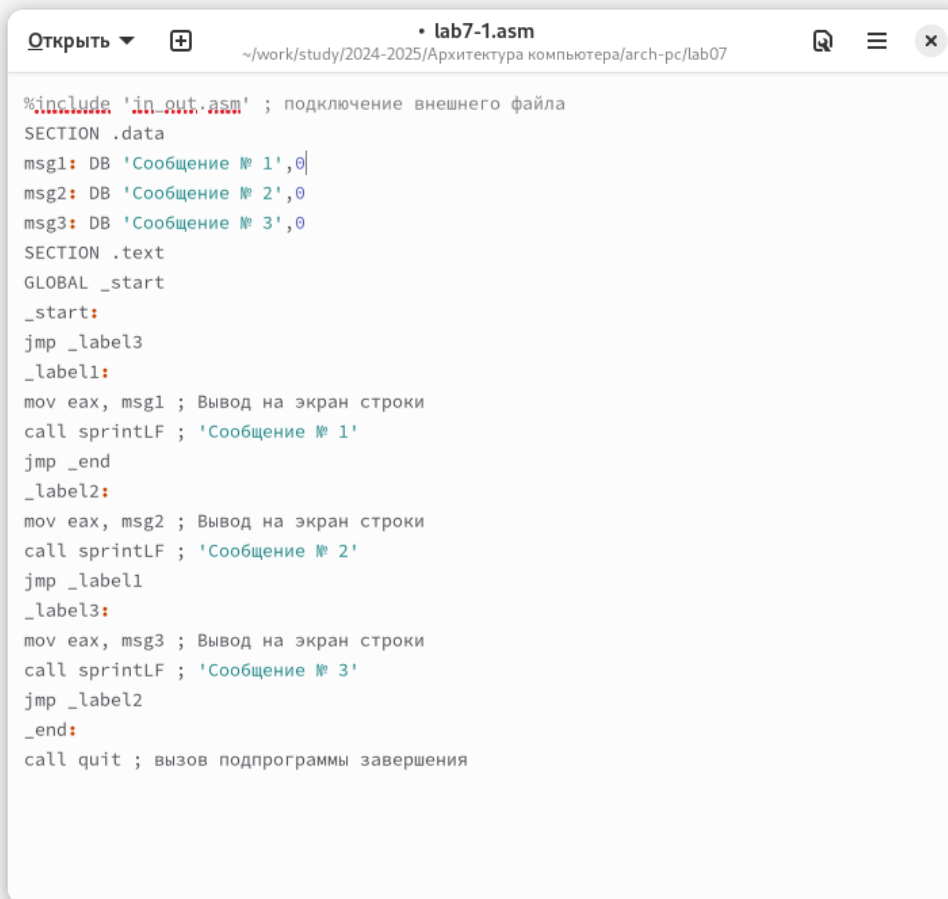
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; 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
```

Запускаю программу и проверяю, что примененные изменения верны.



```
peroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
peroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
peroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
peroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

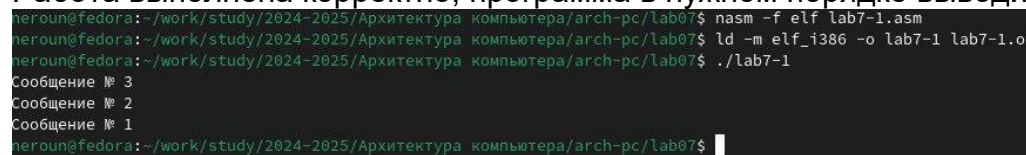
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратной порядке.



```
Открыть + • lab7-1.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/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
```

Работа выполнена корректно, программа в нужном порядке выводит сообщения.



```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Создаю новый рабочий файл и вставляю в него код из следующего листинга.

```
lab7-2.asm
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07

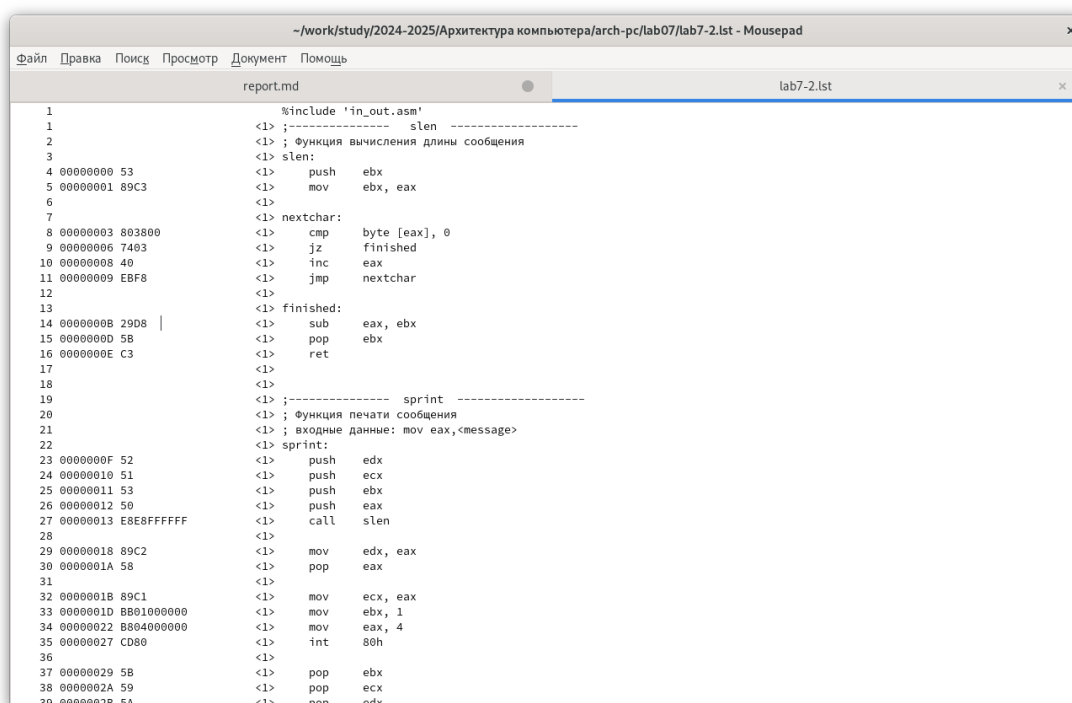
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- 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'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
```

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными.

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите В: 50
Наибольшее число: 50
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

3.2 Изучение структуры файла листинга

Создаю файл листинга с помощью ключа -l команды nasm и открываю его с помощью текстового редактора mousepad.



```
1      %include 'in_out.asm'
2      ;----- slen -----
3      ; функция вычисления длины сообщения
4      slen:
5      <1>      push    ebx
6      <1>      mov     ebx, eax
7      <1>      nextchar:
8      <1>      cmp     byte [eax], 0
9      <1>      jz      finished
10     <1>      inc     eax
11     <1>      jmp     nextchar
12     <1>
13     <1>      finished:
14     <1>      sub     eax, ebx
15     <1>      pop     ebx
16     <1>      ret
17     <1>
18     <1>
19     <1>      ;----- sprint -----
20     <1>      ; функция печати сообщения
21     <1>      ; входные данные: mov eax, <message>
22     <1>      sprint:
23     <1>      push    edx
24     <1>      push    ecx
25     <1>      push    ebx
26     <1>      push    eax
27     <1>      call    slen
28     <1>
29     <1>      mov     edx, eax
30     <1>      pop     eax
31     <1>
32     <1>      mov     ecx, eax
33     <1>      mov     ebx, 1
34     <1>      mov     eax, 4
35     <1>      int     80h
36     <1>
37     <1>      pop     ebx
38     <1>      pop     ecx
39     <1>      pop     edi
```

Первое

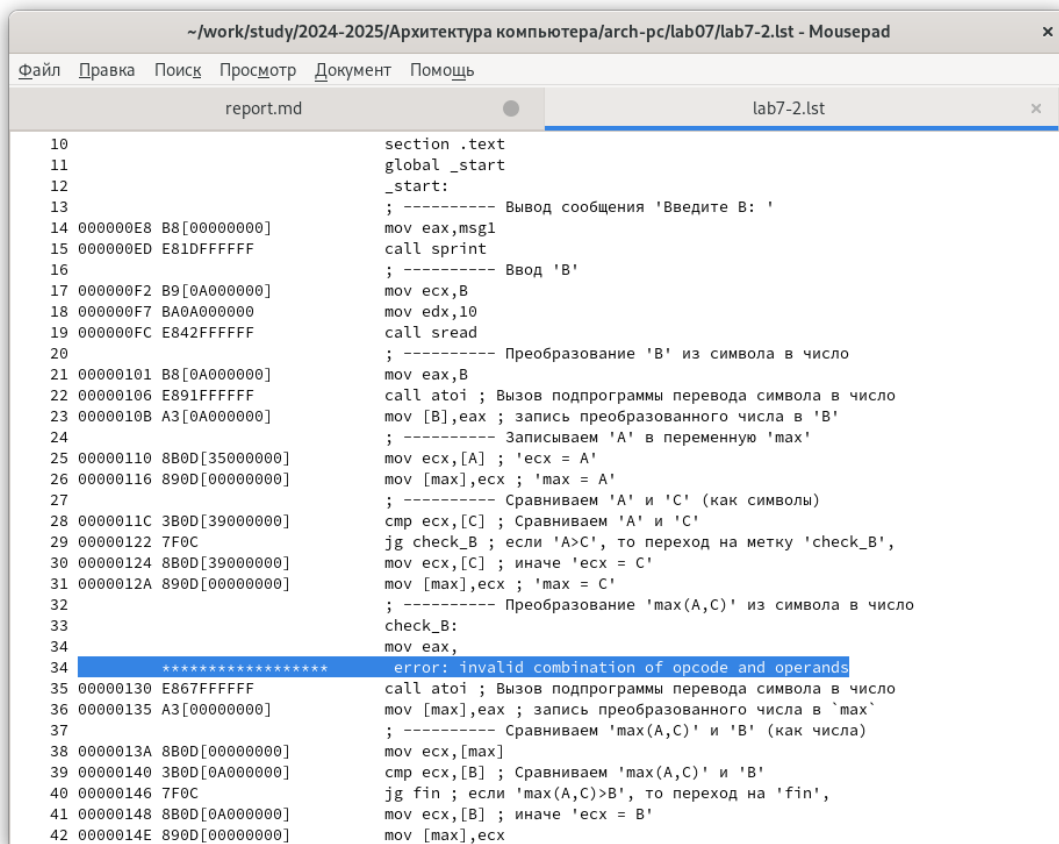
значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем

```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
report.md  lab7-2.lst  lab7-2.asm
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'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,|
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга

не создаются.

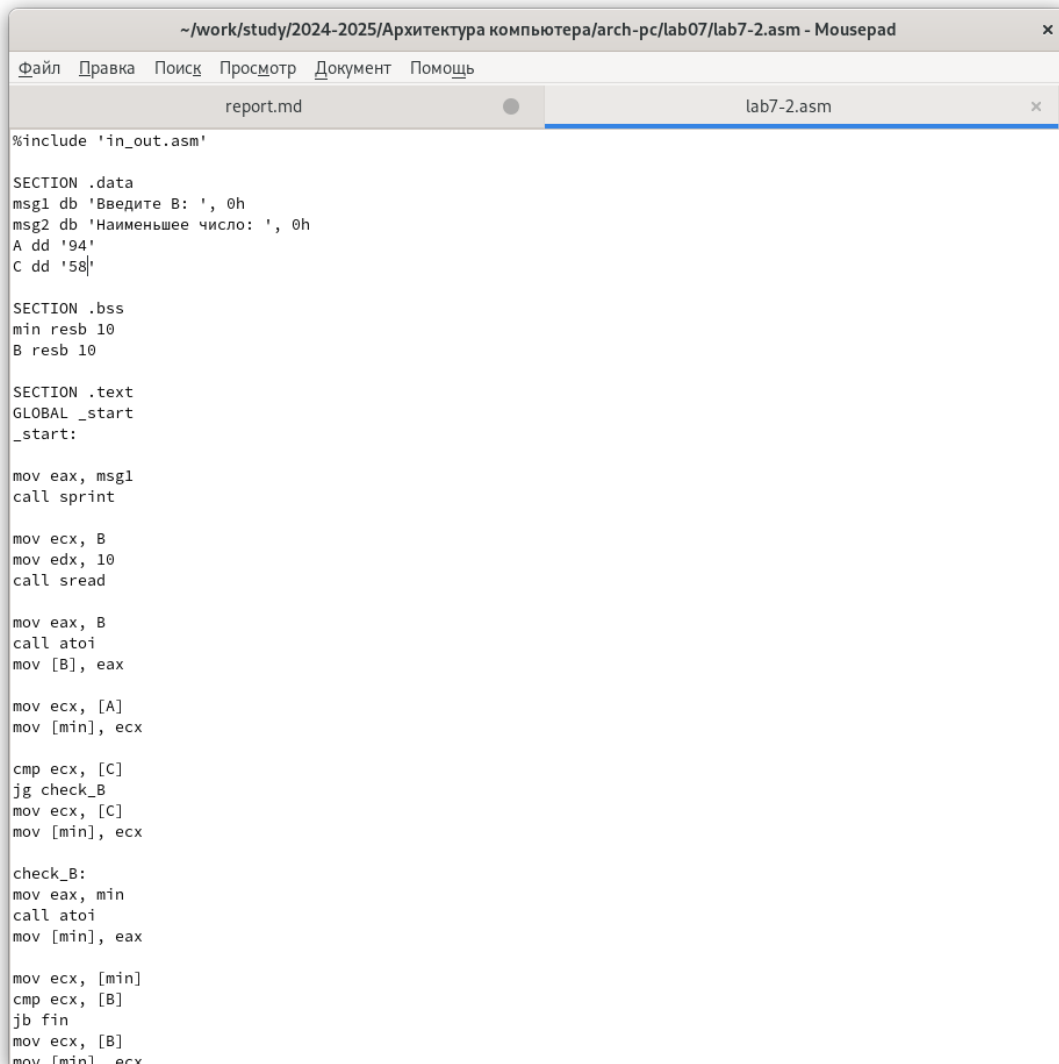


```
~/.work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
report.md  lab7-2.lst
10      section .text
11      global _start
12      _start:
13      ; ----- Вывод сообщения 'Введите B: '
14      000000E8 B8[00000000]    mov eax,msg1
15      000000ED E81DFFFFFF    call sprint
16      ; ----- Ввод 'B'
17      000000F2 B9[0A000000]    mov ecx,B
18      000000F7 BA0A000000    mov edx,10
19      000000FC E842FFFFFF    call sread
20      ; ----- Преобразование 'B' из символа в число
21      00000101 B8[0A000000]    mov eax,B
22      00000106 E891FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
23      0000010B A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
24      ; ----- Записываем 'A' в переменную 'max'
25      00000110 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
26      00000116 890D[00000000]    mov [max],ecx ; 'max = A'
27      ; ----- Сравниваем 'A' и 'C' (как символы)
28      0000011C 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
29      00000122 7F0C    jg check_B ; если 'A>C', то переход на метку 'check_B',
30      00000124 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
31      0000012A 890D[00000000]    mov [max],ecx ; 'max = C'
32      ; ----- Преобразование 'max(A,C)' из символа в число
33      check_B:
34      00000130 E867FFFFFF    mov eax,
35      00000135 A3[00000000]    error: invalid combination of opcode and operands
36      00000135 A3[00000000]    call atoi ; Вызов подпрограммы перевода символа в число
37      00000135 A3[00000000]    mov [max],eax ; запись преобразованного числа в 'max'
38      0000013A 8B0D[00000000]    ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
39      00000140 3B0D[0A000000]    mov ecx,[max]
40      00000146 7F0C    cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
41      00000148 8B0D[0A000000]    jg fin ; если 'max(A,C)>B', то переход на 'fin',
42      0000014E 890D[00000000]    mov ecx,[B] ; иначе 'ecx = B'
43      0000014E 890D[00000000]    mov [max],ecx
```

3.3 Задания для самостоятельной работы.

Скорее всего в описании задания опечатка и я должен использовать свой вариант 3 из лабораторной работы №6. Возвращаю операнд к функции в программе и

изменяю ее так, чтобы она выводила переменную с наименьшим значением.



```
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
report.md  lab7-2.asm

%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '94'
C dd '58'

SECTION .bss
min 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
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jnb fin
mov ecx, [B]
mov [min], ecx
fin:
```

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min 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
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

```

Проверяю корректность написания первой программы.

```

neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наименьшее число: 5
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$

```

4 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.