

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий самостоятельной работы	15
4	Выводы	21

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Работа с файлом lab8-1.asm	7
2.3	Работа с измененным файлом lab8-1.asm	8
2.4	Работа с измененным файлом lab8-1.asm	10
2.5	Работа с файлом lab8-2.asm	12
2.6	Создание файла листинга lab8-2.asm	12
2.7	Удаление операнда	13
2.8	Трансляция файла	13
2.9	Ошибка в файле листинга	14
3.1	Тестирование программы mytask8-1.asm	18
3.2	Тестирование программы mytask8-2.asm	20

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 Выполнение лабораторной работы

Для начала создадим каталог для программ 8-ой лабораторной работы, перейдем в нее и создадим файл lab8-1.asm (рис. 2.1).

```
[aamishina@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[aamishina@fedora ~]$ cd ~/work/arch-pc/lab08  
[aamishina@fedora lab08]$ touch lab8-1.asm  
[aamishina@fedora lab08]$
```

Рис. 2.1: Создание каталога и файла

Вводим текст программы из листинга 8.1 в наш файл. Создадим и запустим исполняемый файл (рис. 2.2). Программа выводит “Сообщение №2” и “Сообщение №3”.

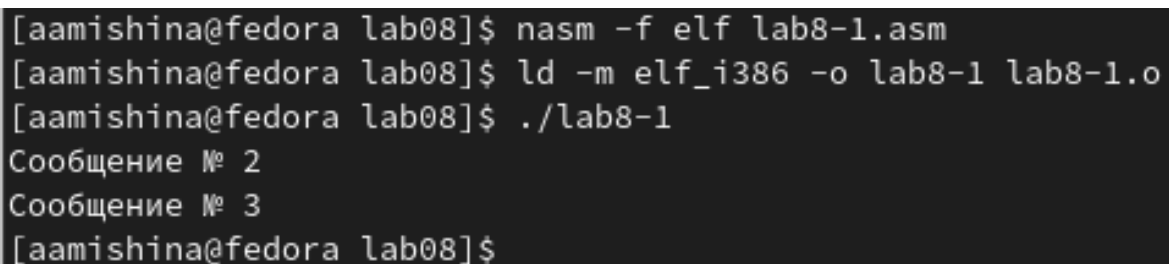
Программа lab8-1.asm:

```
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение No 1',0  
msg2: DB 'Сообщение No 2',0  
msg3: DB 'Сообщение No 3',0  
SECTION .text  
GLOBAL _start  
_start:
```

```

jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```



```

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

```

Рис. 2.2: Работа с файлом lab8-1.asm

Меняем текст программы в соответствии листингом 8.2. Создаем исполняемый файл, видим вывод “Сообщение №2” и “Сообщение №1” (рис. 2.3).

Измененная программа lab8-1.asm:

```

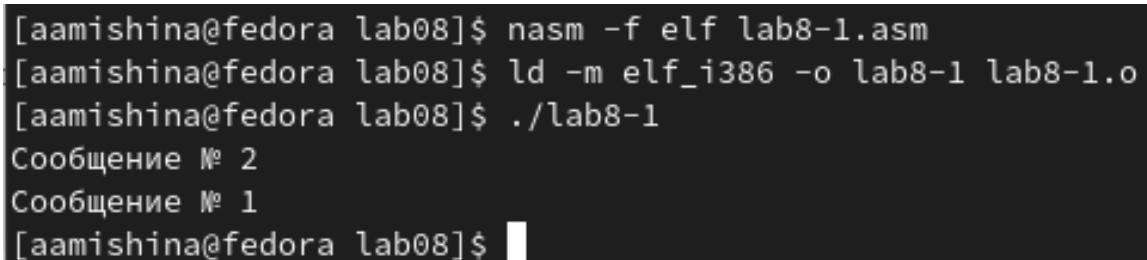
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0

```

```

SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```



```

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

```

Рис. 2.3: Работа с измененным файлом lab8-1.asm

Меняем текст программы, чтобы вывод был в следующем порядке: “Сообщение №3”, “Сообщение №2”, “Сообщение №1” (рис. 2.4).

Измененная программа lab8-1.asm:

```

%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

```

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

```

Рис. 2.4: Работа с измененным файлом lab8-1.asm

Создаем файл lab8-2.asm, вставляем в него текст программы из листинга 8.3 для нахождения наибольшего из 3-ёх чисел. Создаем и запускаем исполняемый файл, проверяем его работу, вводя различные значения переменной В (рис. 2.5).

Программа lab8-2.asm:

```

#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
call sprint
; ----- Ввод 'В'

```

```

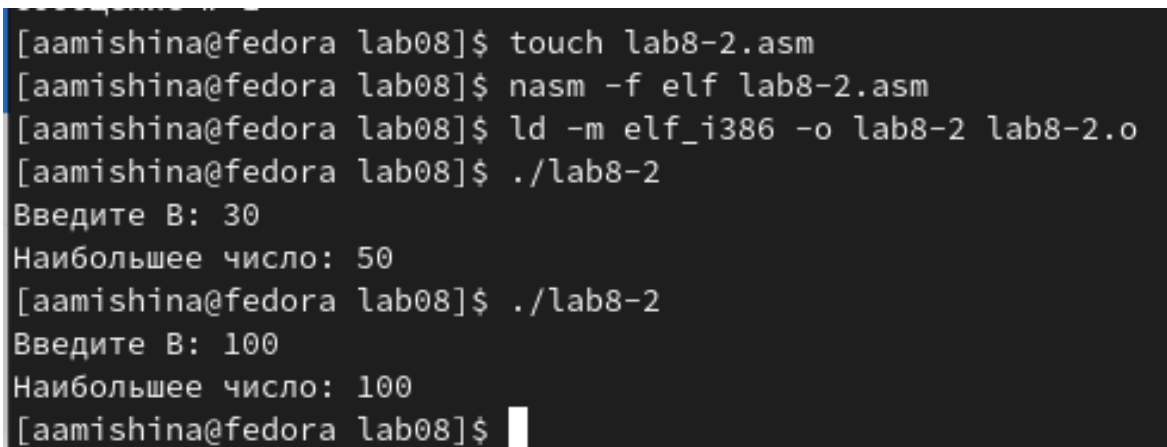
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем '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'
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

```



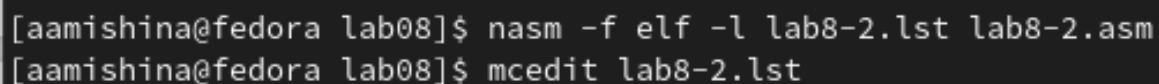
```

[aamishina@fedora lab08]$ touch lab8-2.asm
[aamishina@fedora lab08]$ nasm -f elf lab8-2.asm
[aamishina@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[aamishina@fedora lab08]$ ./lab8-2
Введите B: 30
Наибольшее число: 50
[aamishina@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
[aamishina@fedora lab08]$

```

Рис. 2.5: Работа с файлом lab8-2.asm

Далее создадим файл листинга для программы из файла lab8-2.asm. Открываем файл с помощью редактора mcedit и изучаем его структуру (рис. 2.6).



```

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

```

Рис. 2.6: Создание файла листинга lab8-2.asm

Объясним значение трёх строк из файла.

1. 35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число. 35 - номер строки, 00000135 - адрес, E862FFFFFF - машинный код, call atoi - код программы.
2. 18 000000F7 BA0A000000 mov edx,10. 18 - номер строки, 000000F7 - адрес, BA0A000000 - машинный код, mov edx,10 - код программы.

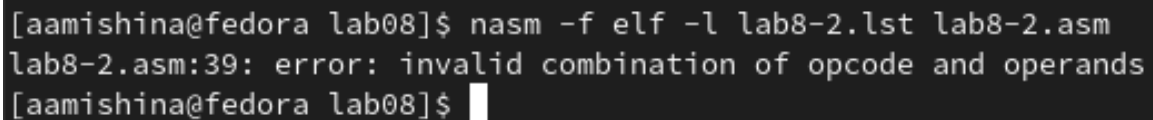
3. 48 00000162 E81FFFFFFF call iprintLF ; Вывод 'max(A,B,C)'. 48 - номер строки, 00000162 - адрес, E81FFFFFFF - машинный код, call iprintLF - код программы.

Теперь открываем файл с программой lab8-2.asm и удаляем один операнд из любой инструкции (рис. 2.7). Выполняем трансляцию файла. Видим, что выводится сообщения об ошибке и создается только файл листинга (рис. 2.8), в котором также присутствует сообщение об ошибке (рис. 2.9).



```
39  cmp ecx, ; Сравниваем 'max(A,C)' и 'B'
```

Рис. 2.7: Удаление операнда



```
[aamishina@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:39: error: invalid combination of opcode and operands
[aamishina@fedora lab08]$
```

Рис. 2.8: Трансляция файла

Открыть

+

lab8-2.lst

~/work/arch-pc/lab08

Стр. 214, Поз. 1

Q

≡

×

lab8-1.asm	lab8-2.asm	lab8-2.lst
202 27		; ----- Сравниваем 'А' и 'С' (как символы)
203 28 0000011C 3B0D[39000000]		cmp ecx,[C] ; Сравниваем 'А' и 'С'
204 29 00000122 7F0C		jg check_B ; если 'А>С', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000]		mov ecx,[C] ; иначе 'ecx = С'
206 31 0000012A 890D[00000000]		mov [max],ecx ; 'max = С'
207 32		; ----- Преобразование 'max(А,С)' из символа в число
208 33		check_B:
209 34 00000130 B8[00000000]		mov eax,max
210 35 00000135 E862FFFFFF		call atoi ; Вызов подпрограммы перевода символа в число
211 36 0000013A A3[00000000]		mov [max],eax ; запись преобразованного числа в 'max'
212 37		; ----- Сравниваем 'max(А,С)' и 'В' (как числа)
213 38 0000013F 8B0D[00000000]		mov ecx,[max]
214 39		cmp ecx, ; Сравниваем 'max(А,С)' и 'В'
215 39 *****		error: invalid combination of opcode and operands
216 40 00000145 7F0C		jg fin ; если 'max(А,С)>В', то переход на 'fin',
217 41 00000147 8B0D[0A000000]		mov ecx,[B] ; иначе 'ecx = В'
218 42 0000014D 890D[00000000]		mov [max],ecx
219 43		; ----- Вывод результата
220 44		fin:
221 45 00000153 B8[13000000]		mov eax, msg2
222 46 00000158 E8B2FFFFFF		call sprint ; Вывод сообщения 'Наибольшее число: '
223 47 0000015D A1[00000000]		mov eax,[max]
224 48 00000162 E81FFFFFFF		call iprintLF ; Вывод 'max(А,В,С)'
225 49 00000167 E86FFFFFFF		call quit ; Выход

Рис. 2.9: Ошибка в файле листинга

3 Выполнение заданий самостоятельной работы

Для начала пишем программу для нахождения минимального из трех чисел. Согласно моему варианту (13) мне следовало проверить правильность выполнения задания на числах: 84, 32, 77. Программа отработала успешно (рис. 3.1).

Программа mytask8-1.asm:

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
SECTION .data
```

```
msg1: DB 'Введите A: ',0
```

```
msg2: DB 'Введите B: ',0
```

```
msg3: DB 'Введите C: ',0
```

```
msg4: DB 'Минимальное число: ',0
```

```
SECTION .bss
```

```
A: RESB 80
```

```
B: RESB 80
```

```
C: RESB 80
```

```
min: RESB 80
```

```
res: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg1
```

```
call sprint
```

```
mov ecx, A
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, A
```

```
call atoi
```

```
mov [A], eax
```

```
mov eax, msg2
```

```
call sprint
```

```
mov ecx, B
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, B
```

```
call atoi
```

```
mov [B], eax
```

```
mov eax, msg3
```

```
call sprint
```

```
mov ecx, C
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, C
```



```

call atoi
mov [C], eax

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

cmp ecx, [B]
jl check_C
mov ecx, [B]
mov [min], ecx

check_C:
cmp ecx, [C]
jl _end
mov ecx, [C]
mov [min], ecx

_end:
mov eax, msg4
call sprint

mov eax, [min]
call iprintLF

call quit ; вызов подпрограммы завершения

```

```

[aamishina@fedora lab08]$ nasm -f elf mytask8-1.asm
[aamishina@fedora lab08]$ ld -m elf_i386 -o mytask8-1 mytask8-1.o
[aamishina@fedora lab08]$ ./mytask8-1
Введите A: 84
Введите B: 32
Введите C: 77
Минимальное число: 32
[aamishina@fedora lab08]$

```

Рис. 3.1: Тестирование программы mytask8-1.asm

Во втором задании требовалось вычислить значение функции 13-ого варианта ($f = a - 7$, $a \geq 7$ и $f = ax$, $a < 7$). Пишем код, создаем и запускаем исполняемый файл, убеждаемся в правильности работы программы (рис. 3.2).

Программа mytask8-2.asm:

```

#include 'in_out.asm'

SECTION .data
msg1: DB 'Введите x: ',0
msg2: DB 'Введите a: ',0

SECTION .bss
A: RESB 80
X: 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 ebx, [A]
cmp ebx, 7
jge first
jmp second
```

first:

```
    mov eax, [A]
    add eax, -7
    call iprintLF
    call quit
```

second:

```
    mov eax, [A]
    mov ebx, [X]
```

```
mul ebx  
call iprintLF  
call quit
```

```
[aamishina@fedora lab08]$ nasm -f elf mytask8-2.asm  
[aamishina@fedora lab08]$ ld -m elf_i386 -o mytask8-2 mytask8-2.o  
[aamishina@fedora lab08]$ ./mytask8-2  
Введите x: 3  
Введите a: 9  
2  
[aamishina@fedora lab08]$ ./mytask8-2  
Введите x: 6  
Введите a: 4  
24  
[aamishina@fedora lab08]$
```

Рис. 3.2: Тестирование программы mytask8-2.asm

4 Выводы

В ходе выполнения данной лабораторной работы я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга. Вся моя работа была записана и прокомментирована мной в данной лабораторной.