

Шаблон отчёта по лабораторной работе №7

Дисциплина: архитектура компьютера

Аннаоразов Сердар Аннаоразович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Изучение структуры файлы листинга	8
4.2	Изучение структуры файлы листинга	11
4.3	Самостоятельная работа.	13
4.3.1	Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c.	13
4.3.2	Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.	14
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Создания каталога и файла	8
4.2	Скопирования файла in_out.asm в нужный каталог	8
4.3	Программа с использованием инструкции jmp	9
4.4	Создания исполняемого файла	9
4.5	Изменения текста файла	10
4.6	Создания(изменённого) исполняемого файла	10
4.7	Создания файла lab7-2.asm	10
4.8	Программа,которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С	11
4.9	Создания исполняемого файла lab7-2.asm	11
4.10	Создания листинга	11
4.11	Открытие листинга	12
4.12	112 строка для объяснения	12
4.13	14 строка для объяснения	12
4.14	42 строка для объяснения	13
4.15	Ошибка в программе	13
4.16	Осмотр листинга	13
4.17	Внесения программы в файл	14
4.18	Создания исполняемого файла lab7-3.asm	14
4.19	Внесения программы в файл lab7-4.asm	15
4.20	Создания исполняемого файла lab7-4.asm	15

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. Реализация переходов в NASM
2. Изучение структуры файлы листинга

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

4.1 Изучение структуры файлы листинга

Для начала я создал каталог для программ Лабораторной работы. потом перешёл в него и создал файл lab07-1.asm (рис. 4.1).

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

Рис. 4.1: Создания каталога и файла

Потом зашел на МС и через него скопировал файл in_out.asm в созданный каталог

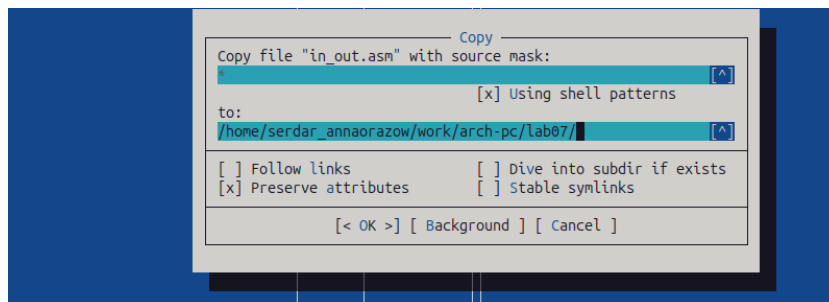


Рис. 4.2: Скопирования файла in_out.asm в нужный каталог

После этого я открыл созданной мною файл с помощью клавиши F4 и ввел туда программу с использованием инструкции jmp


```
mc [serdar_annaorazow@serdar]:~/work/arch-pc/lab07
/home/serdar_annaorazow/work/arch-pc/lab07/lab7-1.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 _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
```

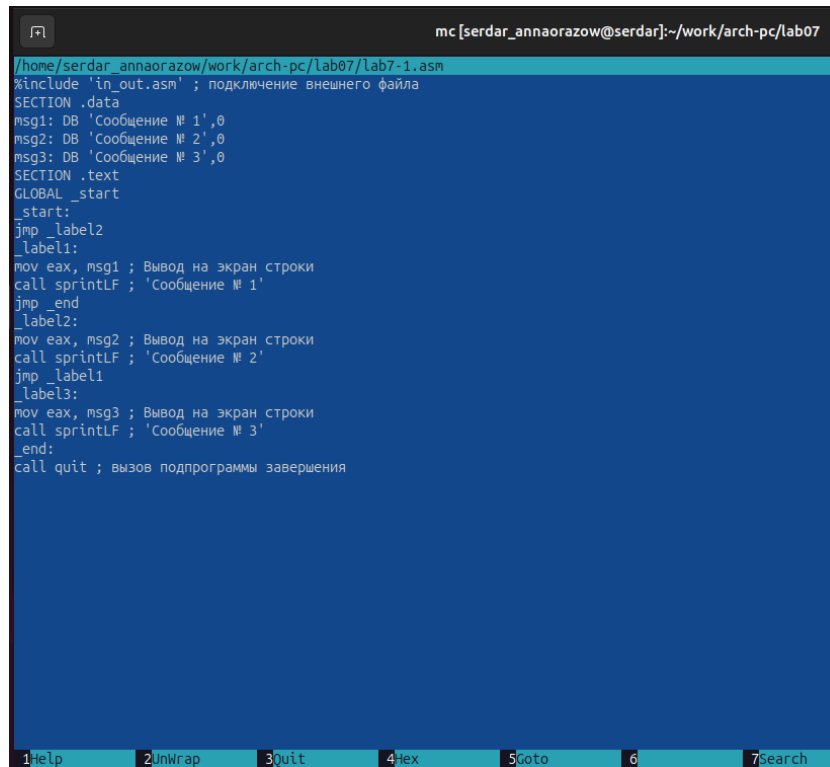
Рис. 4.3: Программа с использованием инструкции jmp

Потом я создал исполняемый файл и запустил его

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

Рис. 4.4: Создания исполняемого файла

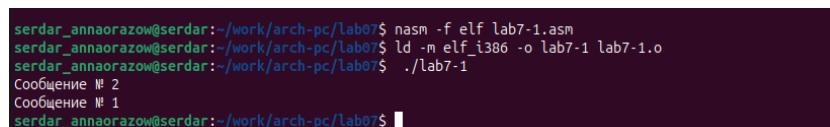
Я изменил текст файла чтобы осуществить переход назад в инструкции jmp. Для этого в текст программы после вывода сообщения № 2 добавил инструкцию jmp с меткой _label1, и после вывода сообщения № 1 добавил инструкцию jmp с меткой _end



```
mc [serdar_annaorazow@serdar]:~/work/arch-pc/lab07
/home/serdar_annaorazow/work/arch-pc/lab07/lab7-1.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 _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
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search
```

Рис. 4.5: Изменения текста файла

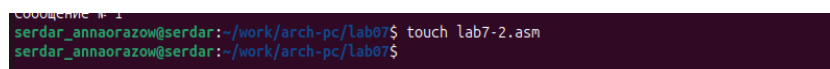
Создал исполняемый файл и запустил его ещё раз но уже изменённого.



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

Рис. 4.6: Создания(изменённого) исполняемого файла

Потом я создал новый файл в том же каталоге lab7-2.asm



```
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ touch lab7-2.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab07$
```

Рис. 4.7: Создания файла lab7-2.asm

После создания я открыл файл и ввёл туда программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С

```
mc [serdar_annaorazow@serdar]:~/work/arch-pc/lab07

/home/serdar_annaorazow/work/arch-pc/lab07/lab7-2.asm
%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)
; ----- Вывод результата
```

Рис. 4.8: Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C

Потом создал исполняемый файл и запустил его. И ещё я проверил его работу

```
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
serdar_annaorazow@serdar:~/work/arch-pc/lab07$
```

Рис. 4.9: Создания исполняемого файла lab7-2.asm

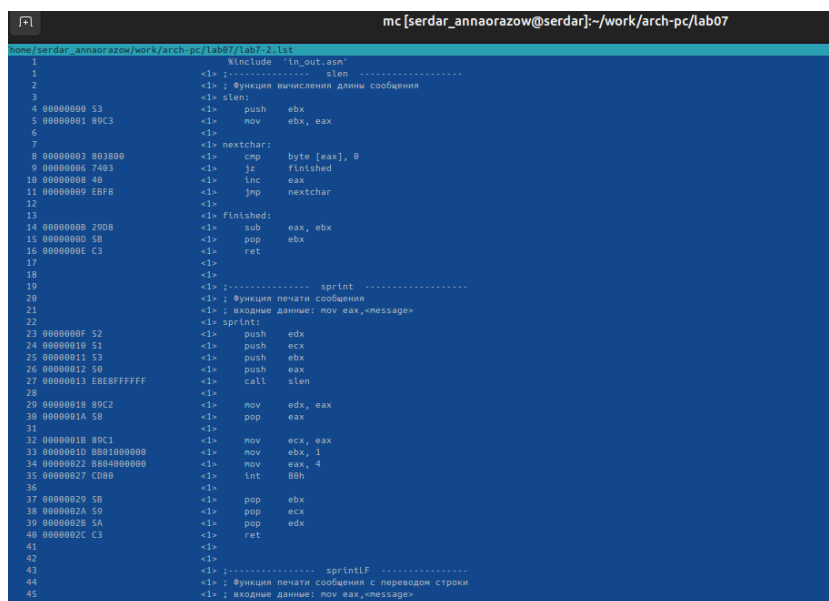
4.2 Изучение структуры файлы листинга

Я создал файл листинга с помощью nasm указав ключ -l и задал имя лисинга в командной строке

```
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab07$
```

Рис. 4.10: Создания листинга

Потом открыл файл листинга с помощью mscedit и изучил содержимое



```
1 00000000 00000000 .include "in_out.asm"
2 00000000 00000000 <1> ; Функция вычисления длины сообщения
3 00000000 00000000 <1> silen
4 00000000 53 <1> push ebx
5 00000000 89C3 <1> mov ebx, eax
6 00000000 00000000 <1>
7 00000000 003800 <1> nextchar:
8 00000000 7403 <1> cmp byte [eax], 0
9 00000000 4B <1> jz finished
10 00000000 EBF8 <1> inc eax
11 00000000 29D8 <1> jmp nextchar
12 00000000 00000000 <1>
13 00000000 29D8 <1> finished:
14 00000000 5B <1> sub eax, ebx
15 00000000 C3 <1> pop ebx
16 00000000 00000000 <1> ret
17 00000000 00000000 <1>
18 00000000 00000000 <1>
19 00000000 52 <1> ; Функция печати сообщения
20 00000000 51 <1> ; Входные данные: mov eax, <message>
21 00000000 50 <1> sprint:
22 00000000 51 <1> push edx
23 00000000 53 <1> push ecx
24 00000000 50 <1> push ebx
25 00000000 50 <1> push eax
26 00000000 E8C9FFFFFF <1> call silen
27 00000000 89C2 <1> mov edx, eax
28 00000000 5B <1> pop eax
29 00000000 00000000 <1>
30 00000000 89C1 <1> mov ecx, eax
31 00000000 80100000 <1> mov ebx, 1
32 00000000 B804000000 <1> mov eax, 4
33 00000000 CD80 <1> int 86h
34 00000000 5B <1>
35 00000000 59 <1> pop ebx
36 00000000 59 <1> pop ecx
37 00000000 5A <1> pop edx
38 00000000 C3 <1> ret
39 00000000 00000000 <1>
40 00000000 00000000 <1>
41 00000000 00000000 <1>
42 00000000 00000000 <1> ; Функция печати сообщения с переводом строки
43 00000000 00000000 <1> ; Входные данные: mov eax, <message>
44 00000000 00000000 <1> sprintf
45 00000000 00000000 <1>
46 00000000 00000000 <1>
```

Рис. 4.11: Открытие листинга

Выбрал первую строку и это 112. В строке которая показана в картинке снизу обозначается “00000086” — адрес в памяти, “E8C9FFFFFF” — машинный код для инструкции call а “call inprint” — обозначает вызов функции inprint.



```
112 00000086 E8C9FFFFFF <1> call inprint
113
```

Рис. 4.12: 112 строка для объяснения

Выбрал вторую строку и это 14. В строке которая показана в картинке снизу обозначается “0000000B” — адрес в памяти, где расположена эта инструкция, 29D8 — машинный код для инструкции sub а “sub eax, ebx” — обозначает операцию, которая вычитает значение регистра ebx из значения регистра eax и сохраняет результат в eax.



```
14 0000000B 29D8 <1> sub eax, ebx
15 00000000 5B <1> pop ebx
```

Рис. 4.13: 14 строка для объяснения

Выбрал третью строку и это 42. В строке которая показана в картинке снизу обозначается “00000153” — адрес в памяти, где расположена эта инструкция,

890D — машинный код для инструкции mov а “mov [max], ecx”— Обозначает операцию, которая копирует значение из регистра ecx в память по адресу, соответствующему метке или переменной max.



Рис. 4.14: 42 строка для объяснения

Потом в строке mov eax,max я убрал max и попробовал создать файл. Выдало ошибку, так как для программы нужно два операнда.

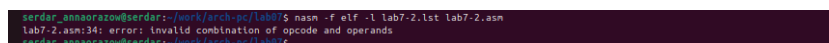


Рис. 4.15: Ошибка в программе

В файле листинга показывает где ошибка и с чем оно связана

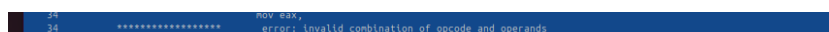


Рис. 4.16: Осмотр листинга

4.3 Самостоятельная работа.

4.3.1 Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c.

Для начала я создал файл и в него я написал программу нахождения наименьшей из 3 целочисленных переменных a,b и c.

```
mc[serdar_annaorazow@serdar]~/work/arch-pc/lab07
/home/serdar_annaorazow/work/arch-pc/lab07/lab7-3.asm
%include "in_out.asm"
SECTION .data
msg1 DB "Введите B: ",0h
msg2 DB "Наименьшее число: ",0h
A dd 8h
C dd 6h
SECTION .bss
min resb 10
B resb 10
SECTION .text
GLOBAL _start
_start:
; ----- Вывод сообщения "Введите B: "
mov eax,msg1
call sprintf
; ----- Ввод "B"
mov ecx,B
mov edx,10
call read
; ----- 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 "min"
mov ecx,[A] ; "ecx = A"
mov [min],ecx ; "min = A"
; ----- Сравниваем "A" и "C" (как символы)
shr ecx,[C] ; Сравниваем "A" и "C"
jl check_B ; если "A<C", то переход на метку "check_B",
mov ecx,[C] ; иначе "ecx = C"
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 "min"
; ----- Сравниваем "min(A,C)" и "B" (как числа)
mov ecx,[min]
shr ecx,[B] ; Сравниваем "min(A,C)" и "B"
jl 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 sprintf ; Вывод сообщения "Наименьшее число: "
mov eax,[min]
call sprintf ; Вывод "min(A,B,C)"
call quit ; Выход
```

Рис. 4.17: Внесения программы в файл

Потом создал исполняемый файл и запустил его и проверил все ли работает

```
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3.o lab7-3.o
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 88
Наименьшее число: 8
serdar_annaorazow@serdar:~/work/arch-pc/lab07$
```

Рис. 4.18: Создания исполняемого файла lab7-3.asm

4.3.2 Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

Для начала я создал файл и в него я написал программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

```
mc [serdar_annaorazow@serdar]:~/work/arch-pc/lab07
/home/serdar_annaorazow/work/arch-pc/lab07/lab7-4.asm
#include "ln_out.asm"

SECTION .data
prln1 DB "2a-x ,x<a" ,0
prln2 DB "8, x=>a" ,0
X1 DB "Введите значение X:",0
A1 DB "Введите значение a:",0
otv DB "Ответ: ",0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax,prln1
call sprintf
mov eax,prln2
call sprintf

mov eax,X1
call sprintf

mov ecx,X
mov edx,10
call read

mov eax,X
call atoi
mov [X],eax

mov eax,A1
call sprintf

mov ecx,A
mov edx,10
call read

mov eax,A
call atoi
mov [A],eax

mov ecx,[X]
mov [F],ecx

cmp ecx,[A]
jl check_or
mov ecx,8
mov [F],ecx
jmp fin
```

Рис. 4.19: Внесения программы в файл lab7-4.asm

После этого я создал исполняемый файл и запустил его. потом я написал цифры которые были таблице на X и на A

```
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-4
2a-x ,x<a
8, x=>a
Введите значение X:1
Введите значение a:2
Ответ: 3
serdar_annaorazow@serdar:~/work/arch-pc/lab07$ ./lab7-4
2a-x ,x<a
8, x=>a
Введите значение X:2
Введите значение a:1
Ответ: 8
serdar_annaorazow@serdar:~/work/arch-pc/lab07$
```

Рис. 4.20: Создания исполняемого файла lab7-4.asm

Все готова!

5 Выводы

Я изучил команды условного и безусловного перехода. Приобрел навыки написания программ с переходами.

Список литературы

(<https://esystem.rudn.ru>) Архитектура компьютеров, Лабораторная работа №7