

Отчёт по лабораторной работе №7

Простейший вариант

Янушкевич Михаил Денисович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	17
4	Выводы	22

Список иллюстраций

2.1	Создание файла	6
2.2	Ввод программы	7
2.3	Создание исполняемого файла	8
2.4	Изменение текста программы	9
2.5	Создание исполняемого файла	10
2.6	Изменение текста программы	11
2.7	Создание исполняемого файла	12
2.8	Создание файла	12
2.9	Ввод текста программы	13
2.10	Создание исполняемого файла	14
2.11	Создание файла листинга	14
2.12	Открытие файла листинга	15
2.13	Удаление операнда	16
2.14	Ошибка при транслировании файла	16
2.15	Ошибка в файле листинга	16
3.1	Создание файла	17
3.2	Написание программы	18
3.3	Создание исполняемого файла, проверка	19
3.4	Написание программы	20
3.5	Создание исполняемого файла, проверка	21

Список таблиц

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Выполнение лабораторной работы

1. Создать каталог для программ ЛБ №7. Перейти в него, создать файл lab7-1.asm. (рис. 2.1)

```
[mdyanushkevich@fedora ~]$ mkdir ~/work/arch-pc/lab07  
[mdyanushkevich@fedora ~]$ cd ~/work/study/lab07  
bash: cd: /home/mdyanushkevich/work/study/lab07: Нет такого фай.  
[mdyanushkevich@fedora ~]$ cd work/arch-pc/lab07  
[mdyanushkevich@fedora lab07]$ touch lab7-1.asm
```

Рис. 2.1: Создание файла

С помощью команды `mkdir` создаём каталог `lab07`, далее переходим в него и с помощью команды `touch` создаём файл `lab7-1.asm`

2. В файл `lab7-1.asm` ввести программу из листинга 7.1.(рис. 2.2)

```
%include 'in_out.asm'

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
call sprintfLF

_label2:
mov eax, msg2
call sprintfLF

_label3:
mov eax, msg3
call sprintfLF

_end:
call quit
```

Рис. 2.2: Ввод программы

Открываем файл lab7-1.asm и водим в него программу с использованием инструкции `jmp` из листинга 7.1

3. Создать исполняемый файл программы и запустить его.(рис. 2.3)

```
[mdyanushkevich@fedora lab07]$ nasm -f elf lab7-1.asm
[mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mdyanushkevich@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 3
[mdyanushkevich@fedora lab07]$ gedit lab7-1.asm
[mdyanushkevich@fedora lab07]$
```

Рис. 2.3: Создание исполняемого файла

В командную строку вводим необходимые команды, чтобы создать исполняемый файл и запустить его. Результатом работы программы будут числа 2 и 3.

4. Изменить текст программы в соответствии с листингом 7.2.(рис. 2.4)


```
%include 'in_out.asm'

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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.4: Изменение текста программы

В файл lab 7-1.asm вносим необходимые изменения из листинга 7.2.

5. Создать исполняемый файл и запустить его.(рис. 2.5)

```
[mdyanushkevich@fedora lab07]$ nasm -f elf lab7-1.asm
[mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mdyanushkevich@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 1
```

Рис. 2.5: Создание исполняемого файла

В командной строке вводим необходимые команды, чтобы создать объектный файл с измененной программой из листинга 7.2. Далее запускаем этот файл. Результатом работы программы будут три последовательных числа: 2 1.

6. Изменить текст программы, изменив инструкции jmp, чтобы результатом программы была последовательность чисел: 3 2 1.(рис. 2.6,007)

```
%include 'in_out.asm'

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 _label3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.6: Изменение текста программы

```
[mdyanushkevich@fedora lab07]$ gedit lab7-1.asm
[mdyanushkevich@fedora lab07]$ gedit lab7-1.asm
[mdyanushkevich@fedora lab07]$ nasm -f elf lab7-1.asm
[mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mdyanushkevich@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 2.7: Создание исполняемого файла

Изменяем порядок выполнения инструкции `jmp` в файле `lab7-1.asm`, далее создаём объектный файл, запускаем его. Результатом работы программы будет последовательность чисел: 3 2 1, что означает, что изменения внесены верно.

6. Создать файл `lab7-2.asm` в каталоге `lab07`. В него ввести текст программы из листинга 7.3.(рис. 2.8,008)

```
[mdyanushkevich@fedora lab07]$ touch lab7-2.asm
```

Рис. 2.8: Создание файла

```

#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:

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 [max],ecx

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

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

mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx

```

Рис. 2.9: Ввод текста программы

С помощью команды `touch` создаём файл `lab7-2.asm`. Далее открываем его и вводим необходимый код из листинга 7.3.

7. Создать исполняемый файл и проверить его работу.(рис. 2.10)

```
[mdyanushkevich@fedora lab07]$ nasm -f elf lab7-2.asm
[mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mdyanushkevich@fedora lab07]$ ./lab7-2
Введите В: 5
Наибольшее число: 50
[mdyanushkevich@fedora lab07]$ ./lab7-2
Введите В: 14
Наибольшее число: 50
[mdyanushkevich@fedora lab07]$ ./lab7-2
Введите В: 51
Наибольшее число: 51
[mdyanushkevich@fedora lab07]$
```

Рис. 2.10: Создание исполняемого файла

В командную строку вводим необходимые команды, чтобы создать исполняемый файл и запустить его. Вводя разные значения `В` мы понимаем, что программа работает исправно.

8. Создать файл листинга для программы из файла `lab7-2.asm`. Открыть файл листинга с помощью любого текстового редактора.(рис. 2.11,011)

```
[mdyanushkevich@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[mdyanushkevich@fedora lab07]$ mcedit lab7-2.lst
```

Рис. 2.11: Создание файла листинга

```

11 00000009 EBF8      <1>      jmp      nextchar.....
12                      <1>.....
13                      <1> finished:
14 0000000B 29D8      <1>      sub     eax, ebx
15 0000000D 5B      <1>      pop     ebx.....
16 0000000E C3      <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 0000000F 52      <1>      push    edx
24 00000010 51      <1>      push    ecx
25 00000011 53      <1>      push    ebx
26 00000012 50      <1>      push    eax
27 00000013 E8E8FFFFFF      <1>      call    slen
28                      <1>.....
29 00000018 89C2      <1>      mov     edx, eax
30 0000001A 58      <1>      pop     eax
31                      <1>.....
32 0000001B 89C1      <1>      mov     ecx, eax
33 0000001D BB01000000      <1>      mov     ebx, 1
34 00000022 B804000000      <1>      mov     eax, 4
35 00000027 CD80      <1>      int     80h
36                      <1>.
37 00000029 5B      <1>      pop     ebx
38 0000002A 59      <1>      pop     ecx

```

Рис. 2.12: Открытие файла листинга

В командной строке прописываем команду для создания файла листинга. Далее с помощью редактора mscedit открываем файл lab7-2.lst, изучаем его содержимое..

mov eax,B call atoi mov[B],eax - эти строчки отвечают за преобразование переменной B в число и запись его в переменную eax.

9. Открыть файл с программой lab7-2.asm. В ней в одной из инструкций удаляем операнд, выполнить трансляцию с получением файла листинга.(рис. 2.15,013,012)

```
0 mov ecx,B
1 mov edx
2 call sread
```

Рис. 2.13: Удаление операнда

```
[mdyanushkevich@fedora lab07]$ gedit lab7-2.asm
[mdyanushkevich@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:21: error: invalid combination of opcode and operands
[mdyanushkevich@fedora lab07]$
```

Рис. 2.14: Ошибка при транслировании файла

```
21                                     mov edx
21      *****                      error: invalid combination of opcode
22 00000007 5847555555                    call sread
```

Рис. 2.15: Ошибка в файле листинга

С помощью редактора gedit открываем файл lab7-2.asm, удаляем один операнд. Далее выполняем трансляцию с получением файла листинга. Можно увидеть, что выдаётся ошибка. Открыв файл листинга, можно увидеть, что в месте удаленного операнда также появилась ошибка.

3 Задание для самостоятельной работы

1. Написать программу нахождения наименьшей из 3 целочисленных переменных a,b,c.(рис. 3.1,016,017)



```
[mdyanushkevich@fedora lab07]$ touch lab7-3.asm
```

Рис. 3.1: Создание файла

```

#include 'in_out.asm'

section .data
msg2 db "Наименьшее число: ",0h
A dd '17'
B dd '23'
C dd '45'

section .bss
min resb 10

section .text
global _start
_start:

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

mov ecx,[B]
mov [min],ecx

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

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

mov ecx,[min]
cmp ecx,[A]
jl fin
mov ecx,[A]
mov [min],ecx

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

```

Рис. 3.2: Написание программы

```
mdyanushkevich@fedora lab07]$ gcc -c lab7-3.asm  
mdyanushkevich@fedora lab07]$ nasm -f elf lab7-3.asm  
mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o  
mdyanushkevich@fedora lab07]$ ./lab7-3  
Наименьшее число: 17
```

Рис. 3.3: Создание исполняемого файла, проверка

С помощью команды `touch` создаём файл `lab7-3.asm`. Открываем его и пишем программу нахождения наименьшей из 3 целочисленных переменных `a, b, c` (мой вариант-1, поэтому числа равны 17, 23, 45). Далее создаём исполняемый файл и проверяем его работу. Наименьшим числом программа посчитала число 17, что является правильным ответом.

2. Написать программу для расчета значения функции для введенных с клавиатуры `a` и `x`. (рис. 3.4, 19)

```

#include "in_out.asm"
SECTION .data
msg1 db "Введите x: ",0h
msg2 db "Введите a: ",0h
msg3 db "f(x)= "

SECTION .bss
x resb 10
a resb 10

SECTION .text
global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,10
    call sread
    mov eax,x
    call atoi
    mov [x],eax
    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,10
    call sread
    mov eax,a
    call atoi
    mov [a],eax
    mov ecx,[a]
    cmp ecx,[x]
    jge check_a
    mov eax,[a]
    mov ecx,2
    mul ecx
    neg eax
    mov ecx,[x]
    add ecx,eax
    jmp _end

check_a:
    mov ecx,8

_end:
    mov eax,msg3
    call sprint
    mov eax,ecx
    call iprintLF
    call quit

```

Рис. 3.4: Написание программы

```
[mdyanushkevich@fedora lab07]$ nasm -f elf lab7-4.asm
[mdyanushkevich@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[mdyanushkevich@fedora lab07]$ ./lab7-4
Введите x: 2
Введите a: 1
f(x)= 0
[mdyanushkevich@fedora lab07]$ ./lab7-4
Введите x: 1
Введите a: 2
f(x)= 8
[mdyanushkevich@fedora lab07]$
```

Рис. 3.5: Создание исполняемого файла, проверка

Создаём файл lab7-4.asm, в нём пишем программу расчёта функции для $a=1$, $x=2$ и $a=2$, $x=1$. Создаём исполняемый файл, проверяем его работу для обоих случаев. Результатом работы программы стали числа 8 и 0, что является правильным ответом.

4 Выводы

Благодаря этой лабораторной работе я изучил команды условного и безусловного перехода, благодаря чему смог написать несколько программ с использованием приобретенных ранее знаний по использованию языка ассемблера NASM.