

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Чипурной Михаил Евгеньевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	10
2.3	Задание для самостоятельной работы	12
3	Выводы	16

Список иллюстраций

2.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
2.2	Заполняем файл	6
2.3	Запускаем файл и смотрим на его работу	7
2.4	Изменяем файл	7
2.5	Запускаем файл и смотрим на его работу	7
2.6	Редактируем файл	8
2.7	Проверяем, сошелся ли наш вывод с данным в условии выводом . .	8
2.8	Создаем файл командой <code>touch</code>	8
2.9	Заполняем файл	9
2.10	Смотрим на работу программ	9
2.11	Создаем файл листинга	10
2.12	Изучаем файл	10
2.13	Удаляем операндум из файла	11
2.14	Транслируем файл	11
2.15	Изучаем файл с ошибкой	12
2.16	Создаем файл командой <code>touch</code>	12
2.17	Пишем программу	13
2.18	Смотрим на работу программы(всё верно)	14
2.19	Создаем файл командой <code>touch</code>	14
2.20	Пишем программу	15
2.21	Проверяем работу программы	15
2.22	Проверяем работу программы	15

Список таблиц

1 Цель работы

Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

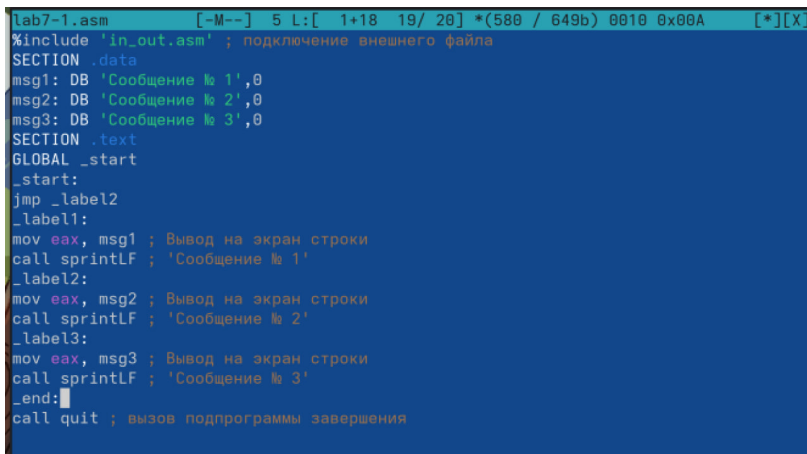
2.1 Реализация переходов в NASM

Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. Рисунок 2.1).

```
mechipurnoyj@fedora:~$ mkdir ~/work/arch-pc/lab07
mechipurnoyj@fedora:~$ cd ~/work/arch-pc/lab07
mechipurnoyj@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$
```

Рисунок 2.1: Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. Рисунок 2.2).



```
lab7-1.asm [-M--] 5 L: [ 1+18 19/ 20] *(580 / 649b) 0010 0x00A [*][X]
%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
```

Рисунок 2.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.3).

```

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

```

Рисунок 2.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. Рисунок 2.4).

```

lab7-1.asm  [-M--]  0 L:[ 1+12 13/ 23] *(359 / 671b) 0106 0x06A
%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

```

Рисунок 2.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.5).

```

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

```

Рисунок 2.5: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. Рисунок 2.6).

```

lab7-1.asm [----] 11 L: [ 2+19 21/ 23] *(341 / 357b) 0010 0x00A [*][X
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
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
jmp _label2
_end:
call quit
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС 10Выход

```

Рисунок 2.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.7).

```

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

```

Рисунок 2.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. Рисунок 2.8).

```

mechipurnoyj@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$

```

Рисунок 2.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. Рисунок 2.9).


```

lab7-2.asm          [-M--]  9 L:  1+ 2  3/ 49] *(45  /1743b) 1042 0x412
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '28'
C dd '58'
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'
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

```

Рисунок 2.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B (рис. Рисунок 2.10).

```

mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
mechipurnoyj@fedora:~/work/arch-pc/lab07$

```

Рисунок 2.10: Смотрим на работу программ

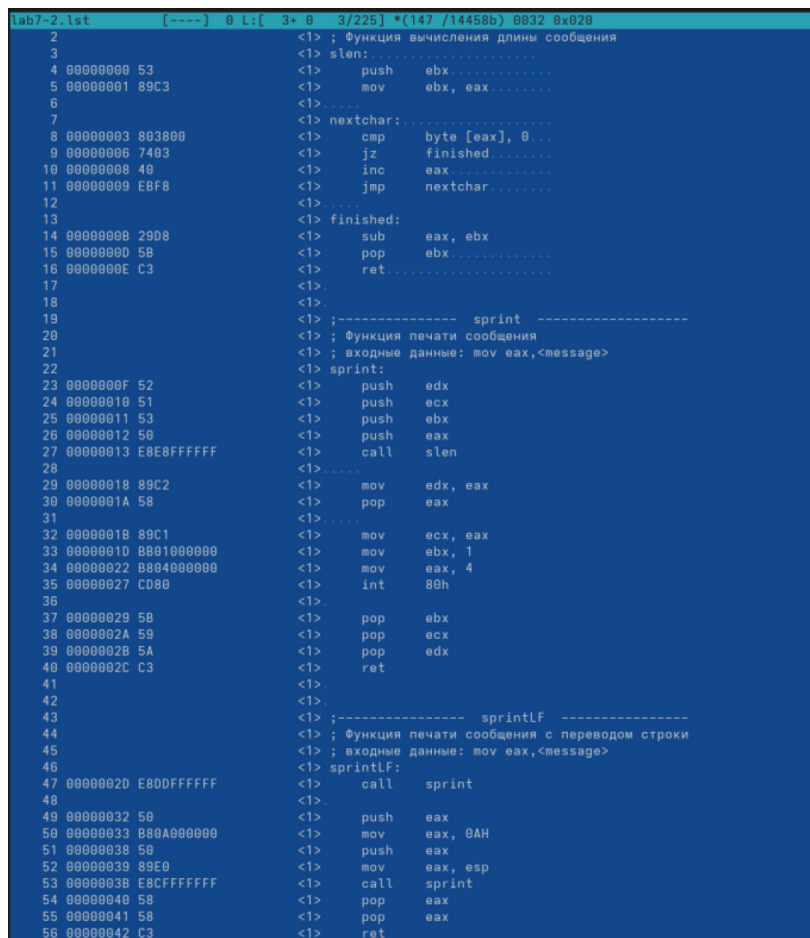
2.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. Рисунок 2.11).

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

Рисунок 2.11: Создаем файл листинга

Открываем файл листинга с помощью команды mcedit и изучаем его (рис. Рисунок 2.12).



```
lab7-2.lst  [-----]  0 L:[ 3+ 0 3/225] *(147/14458b) 0032 0x020
2          <1> ; Функция вычисления длины сообщения
3          <1> slen:.....
4 00000000 53          <1>      push    ebx.....
5 00000001 89C3        <1>      mov     ebx, eax.....
6          <1>.....
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>.....
13          <1> finished:
14 0000000B 2908        <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 5B          <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 CD00        <1>      int     80h
36          <1>.....
37 00000029 5B          <1>      pop     ebx
38 0000002A 59          <1>      pop     ecx
39 0000002B 5A          <1>      pop     edx
40 0000002C C3          <1>      ret.....
41          <1>.....
42          <1>.....
43          <1> ;----- sprintf -----
44          <1> ; Функция печати сообщения с переводом строки
45          <1> ; входные данные: mov eax,<message>
46          <1> sprintf:
47 0000002D E800FFFFFF    <1>      call    sprint
48          <1>.....
49 00000032 50          <1>      push    eax
50 00000033 B80A000000    <1>      mov     eax, 0AH
51 00000038 50          <1>      push    eax
52 00000039 89E0        <1>      mov     eax, esp
53 0000003B E8CFFFFFFF    <1>      call    sprint
54 00000040 5B          <1>      pop     eax
55 00000041 5B          <1>      pop     eax
56 00000042 C3          <1>      ret.....
```

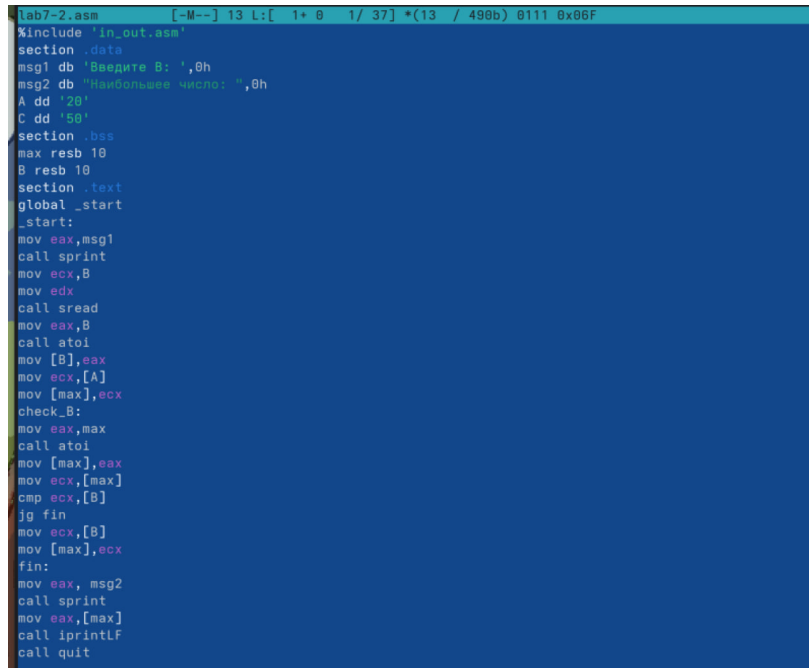
Рисунок 2.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, mov ebx,1-присвоение переменной ebx значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

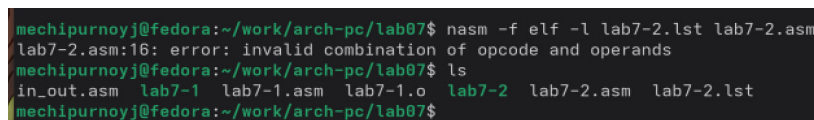
Открываем файл и удаляем один операндум (рис. Рисунок 2.13).



```
lab7-2.asm [-M--] 13 L: [ 1+ B 1/ 37] *(13 / 498b) 8111 8x86F
#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
call sread
mov eax,B
call atoi
mov [B],eax
mov ecx,[A]
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
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintf
call quit
```

Рисунок 2.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. Рисунок 2.14).



```
mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
mechipurnoyj@fedora:~/work/arch-pc/lab07$
```

Рисунок 2.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. Рисунок 2.15).

```

lab7-2.lst      [----] 68 L:[ 1+ 8      9/214] *(634 /12816b) 0120 0x078
1               %include 'in_out.asm'
2               <1> ;----- slen -----
3               <1> ; Функция вычисления длины сообщения
4 00000000 53    <1> push    ebx
5 00000001 89C3  <1> mov     ebx, eax
6               <1>.....
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>.....
13               <1> finished:
14 0000000B 2908  <1> sub     eax, ebx
15 0000000D 58    <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 B801000000 <1> mov     ebx, 1
34 00000022 B804000000 <1> mov     eax, 4
35 00000027 CD80  <1> int     80h
36               <1>.....
37 00000029 58    <1> pop     ebx
38 0000002A 59    <1> pop     ecx
39 0000002B 5A    <1> pop     edx
40 0000002C C3    <1> ret
41               <1>.....
42               <1>.....
43               <1> ;----- sprintf -----
44               <1> ; Функция печати сообщения с переводом строки
45               <1> ; входные данные: mov eax,<message>

```

Рисунок 2.15: Изучаем файл с ошибкой

2.3 Задание для самостоятельной работы

ВАРИАНТ-17

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. Рисунок 2.16).

```

mechipurnoyj@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$

```

Рисунок 2.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3-е вводится из консоли) (рис. Рисунок 2.17).

```
lab7-3.asm      [-M--] 1 L:[ 1+ 0 1/ 41] *(1 / 781b) 0105 0x
%include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd '26'
    C dd '68'
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]
    jl     check_B
    mov     ecx,[C]
    mov     [min],ecx
check_B:
    mov     eax,min
    call    atoi
    mov     [min],eax
    mov     ecx,[min]
    cmp     ecx,[B]
    jl     fin
    mov     ecx,[B]
    mov     [min],ecx
fin:
    mov     eax,msg2
    call    sprint
    mov     eax,[min]
    call    iprintLF
    call    quit
```

Рисунок 2.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. Рисунок 2.18).

```

lab7-3.asm          [-M--] 15 L:[ 1+13 14/ 41] *(276 / 703b) 8010 0x00A
#include 'in_out.asm'
section             .data
    msg1 db 'Введите B: ',0h
    msg2 db 'Наименьшее число: ',0h
    A dd '26'
    C dd '68'
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]
    jl     check_B
    mov     ecx,[C]
    mov     [min],ecx
check_B:
    mov     eax,min
    call    atoi
    mov     [min],eax
    mov     ecx,[min]
    cmp     ecx,[B]
    jl     fin
    mov     ecx,[B]
    mov     [min],ecx
fin:
    mov     eax, msg2
    call    sprint
    mov     eax,[min]
    call    iprintf
    call    quit

```

Рисунок 2.18: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создаем новый файл (рис. Рисунок 2.19).

```

mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 12
Наименьшее число: 12

```

Рисунок 2.19: Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. Рисунок 2.20).

```
mechipurnoyj@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$
```

Рисунок 2.20: Пишем программу

Транслируем файл и проверяем его работу при $x=3$ и $a=4$ (рис. Рисунок 2.21).

```
lab7-4.asm [----] 6 L:[ 1+29 38/ 45] *(501 / 735b) 0032 0x020
#include 'in_out.asm'
SECTION .data
    msg1: DB 'Введите x: ',0h
    msg2: DB 'Введите a: ',0h
    otv: DB 'F(x) = ',0h
SECTION .bss
    x: RESB 80
    a: RESB 80
    res: 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
    cmp     eax,8
    jl      check_A
    mov     eax,[a]
    mov     ebx,[x]
    imul    eax,ebx
    mov     [res],eax
    jmp     fin
check_A:
    mov     eax,[a]
    add     eax,8
    mov     [res],eax
fin:
    mov     eax,otv
    call    sprint
    mov     eax,[res]
    call    iprintLF
    call    quit
```

Рисунок 2.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x=2$ и $a=9$ (рис. Рисунок 2.22).

```
mechipurnoyj@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mechipurnoyj@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 4
F(x) = 12
```

Рисунок 2.22: Проверяем работу программы

3 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.