

Отчёт по лабораторной работе №6

дисциплина: Архитектура вычислительных систем

Мосолов Александр Денисович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Символьные и численные данные в NASM	6
2.2	Выполнение арифметических операций в NASM	10
2.3	Задание для самостоятельной работы	13
3	Выводы	15

Список иллюстраций

2.1	Создаем каталог для работы	6
2.2	Текст из листинга 6.1	6
2.3	Каталог lab06	7
2.4	Запускаем lab6-1.asm	7
2.5	Вводим текст	8
2.6	Вывод программы после изменения символов на числа	8
2.7	Создаём файл lab6-2.asm	8
2.8	Текст из листинга 6.2	9
2.9	Вывод программы lab6-2	9
2.10	Текст программы после изменений	9
2.11	Используем iprintLF	9
2.12	Используем iprint	10
2.13	Создание файла lab6-3.asm	10
2.14	Текст программы lab6-3.asm	10
2.15	Результат компиляции файла lab6-3	11
2.16	Текст файла lab6-3 после изменения	11
2.17	Результат запуска изменённого файла lab6-3	11
2.18	Текст программы variant.asm	12
2.19	Номер варианта	12
2.20	Текст программы - вычисление значения функции	13
2.21	Находим значение функции при $x = 3$	13
2.22	Находим значение функции при $x = 1$	14

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера *NASM*.

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

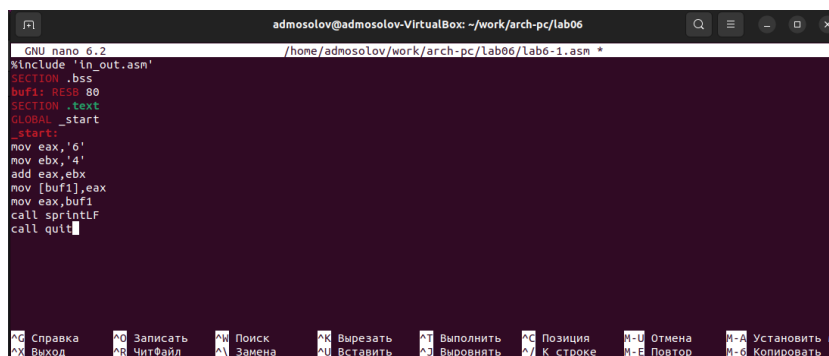
Создаём каталог для программ лабораторной работы № 6, переходим в него и создаём файл *lab6-1.asm*.

```
admosolov@admosolov-VirtualBox:~$ mkdir ~/work/arch-pc/lab06
admosolov@admosolov-VirtualBox:~$ cd ~/work/arch-pc/lab06
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ touch lab6-1.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ ls
lab6-1.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.1: Создаем каталог для работы

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр *eax*.

Введём в файл *lab6-1.asm* текст программы из листинга 6.1.



```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/lab6-1.asm *
%include 'in_out.asm'
SECTION .bss
buf1: resb 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit
```

Рис. 2.2: Текст из листинга 6.1

Для того, чтобы программа транслировалась без ошибок перенесем файл *in_out.asm* в *~/work/arch-pc/lab06*, проверим содержимое каталога.

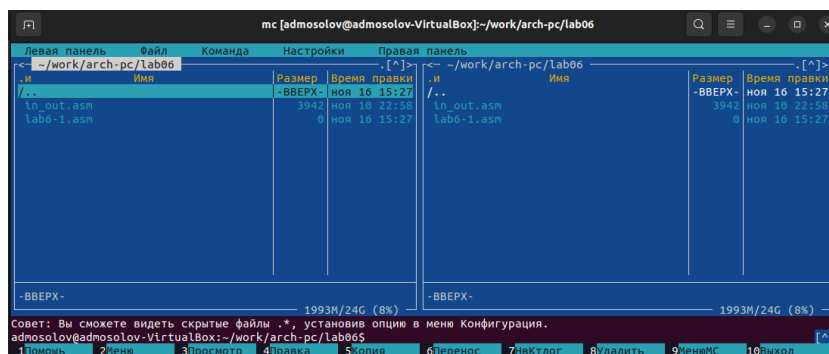


Рис. 2.3: Каталог lab06

Транслируем полученный текст программы *lab6-1.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
j
```

Рис. 2.4: Запускаем lab6-1.asm

В данном случае при выводе значения регистра *eax* мы ожидаем увидеть число 10. Однако результатом будет символ *j*. Это происходит потому, что код символа *б* равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа *4* – 00110100 (52). Команда *add eax,ebx* запишет в регистр *eax* сумму кодов – 01101010 (106), что в свою очередь является кодом символа *j*.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ gedit hello.asm

hello.asm
~/work/arch-pc/lab04

1; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 2.5: Вводим текст

Далее изменим текст программы и вместо символов, запишем в регистры числа.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1

admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.6: Вывод программы после изменения символов на числа

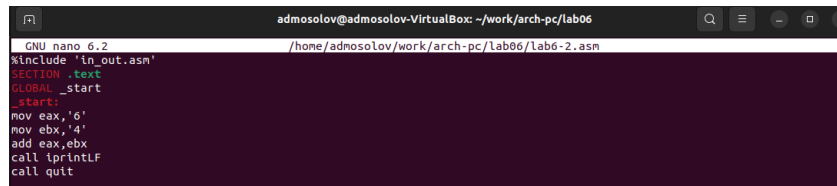
В данном случае выводится символ с кодом 10. Но на экране он не отображается. Создадим файл *lab6-2.asm* в каталоге *~/work/arch-pc/lab06*.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ touch lab6-2.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.7: Создаём файл lab6-2.asm

Введём в него текст программы из листинга 6.2, запустим программу с помощью команд:

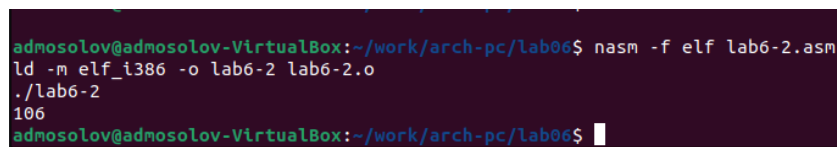
```
nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
```

```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/lab6-2.asm
#include "in_out.asm"
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 2.8: Текст из листинга 6.2

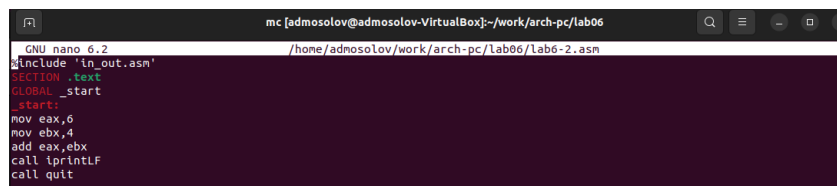
В результате работы программы мы получим число *106*. В данном случае, как и в первом, команда *add* складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличие от программы из листинга 6.1, функция *iprintLF* позволяет вывести число, а не символ, кодом которого является это число.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
106
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.9: Вывод программы lab6-2

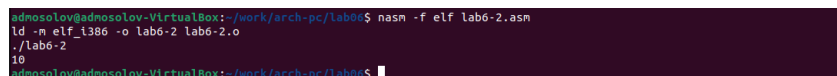
Аналогично предыдущему примеру изменим символы на числа.



```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/lab6-2.asm
#include "in_out.asm"
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 2.10: Текст программы после изменений

Посмотрим на вывод программы с использованием *iprintLF*.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
10
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.11: Используем iprintLF

А теперь на вывод программы с использованием *iprint*.

```
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$
```

Рис. 2.12: Используем `iprint`

Сравним полученные результаты. В случае с использованием `iprintLF` курсор переводится на следующую строчку, а в случае с использованием `iprint` курсор не переносится на новую строку.

2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$.

```
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$ ls
ln_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3.asm
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$
```

Рис. 2.13: Создание файла `lab6-3.asm`

Изучим и введём текст из листинга 6.3 в `lab6-3.asm`.

```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/lab6-3.asm
#include "ln_out.asm"
section .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
section .text
GLOBAL _start
_start:
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; --- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
[Прочитано 25 строк]
^G Справка ^O Записать ^M Поиск ^K Вырезать ^T Выполнить ^G Позиция ^U Отмена ^I-А Установить
^X Выход ^O ЧитФайл ^A Замена ^U Вставить ^Z Выровнять ^V К строке ^M-В Повтор ^I-Б Копировать
```

Рис. 2.14: Текст программы `lab6-3.asm`

Запустим исполняемый файл, убедимся в правильности результата.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
ld -m elf_i386 -o lab6-3 lab6-3.o
./lab6-3
Результат: 4
Остаток от деления: 1
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$
```

Рис. 2.15: Результат компиляции файла lab6-3

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/variant.asm
#include "in_out.asm"
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx

```

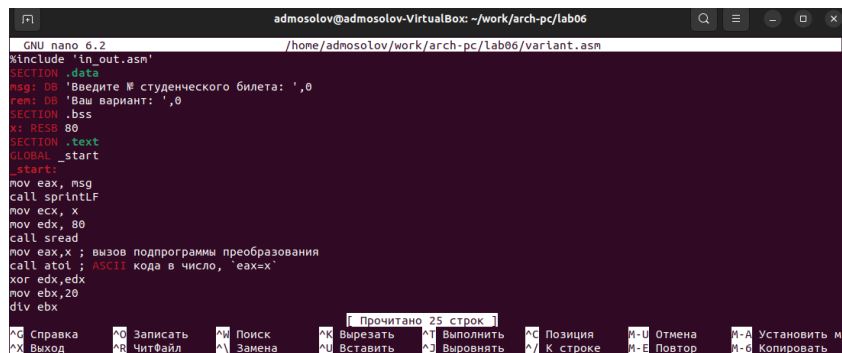
Рис. 2.16: Текст файла lab6-3 после изменения

Транслируем полученный текст программы *lab6-3.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
ld -m elf_i386 -o lab6-3 lab6-3.o
./lab6-3
Результат: 5
Остаток от деления: 1
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$
```

Рис. 2.17: Результат запуска изменённого файла lab6-3


Создаём файл *variant.asm* в каталоге *~/work/arch-pc/lab06* с помощью команды: *touch ~/work/arch-pc/lab06/variant.asm*. Читаем текст из листинга 6.4 и вводим его в файл *variant.asm*.



```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/variant.asm
#include "in_out.asm"
SECTION .data
msg: DB "Введите № студенческого билета: ",0
rem: DB "Ваш вариант: ",0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
```

Рис. 2.18: Текст программы variant.asm

Транслируем текст программы *variant.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.



```
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$ nasm -f elf variant.asm
ld -m elf_i386 -o variant variant.o
./variant
Введите № студенческого билета:
1132236128
Ваш вариант: 9
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm variant variant.o
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o variant.asm
admosolov@admosolov-VirtualBox: /work/arch-pc/lab06$
```

Рис. 2.19: Номер варианта

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

mov eax, rem call sprint

2. Для чего используются следующие инструкции?

mov ecx, x - запись адреса переменной в EAX

mov edx, 80 - запись длины вводимого сообщения в EBX

call sread - вызов подпрограммы ввода сообщения

3. Для чего используется инструкция “*call atoi*”?

Вызывается функция преобразования *ascii-код символа* в целое число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

mov ebx, 20

div ebx

5. В какой регистр записывается остаток от деления при выполнении инструкции “*div ebx*”?

eax

6. Для чего используется инструкция “*inc edx*”?

Увеличиваем *edx* на единицу

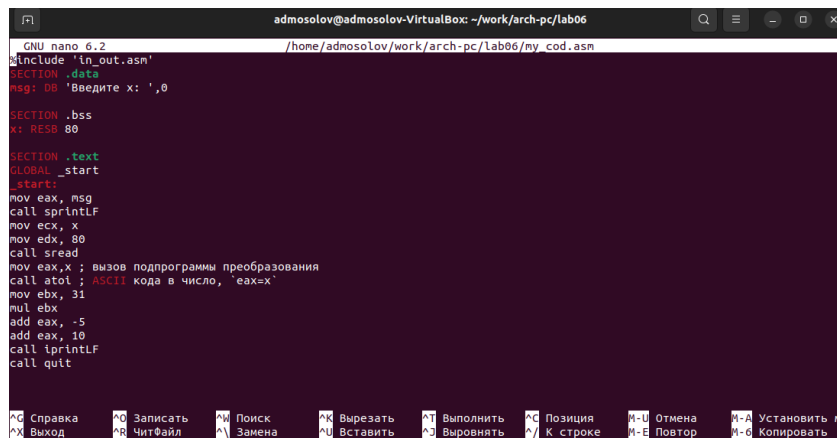
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

mov eax,edx

call iprintLF

2.3 Задание для самостоятельной работы

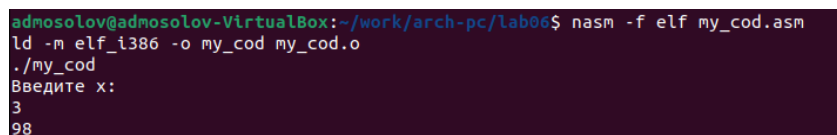
Напишем программу для вычисления выражения $f(x) = 10 + (31x - 5)$ (9 вариант).



```
GNU nano 6.2 /home/admosolov/work/arch-pc/lab06/my_cod.asm
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
mov ebx, 31
mul ebx
add eax, -5
add eax, 10
call iprintLF
call quit
```

Рис. 2.20: Текст программы - вычисление значения функции

Выведем значение функции при $x = 3$ и $x = 1$.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf my_cod.asm
ld -m elf_i386 -o my_cod my_cod.o
./my_cod
Введите x:
3
98
```

Рис. 2.21: Находим значение функции при $x = 3$

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf my_cod.asm
ld -m elf_i386 -o my_cod my_cod.o
./my_cod
Введите x:
1
36
```

Рис. 2.22: Находим значение функции при $x = 1$

Перенесём файлы с программой в `~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06`, загрузим изменения на *github*.

3 Выводы

В ходе лабораторной работы были освоены арифметических инструкции языка ассемблера *NASM*.