

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**дисциплина: Архитектура компьютеров и операционные системы**

**Студент: Ян Роман Алексеевич**

**Группа: НПИбд-02-23**

# Содержание

## 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

### 2.1 Символьные и численные данные в NASM

Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл *lab6-1.asm*:

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

Рис 2.1.1: Создание каталога и файла .asm

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

Введем в файле *lab6-1.asm* текст программы из листинга 6.1. В данной программе в регистр *eax* записывается символ 6 (*mov eax,'6'*), в регистр *ebx* символ 4 (*mov ebx,'4'*). Далее к значению в регистре *eax* прибавляем значение регистра *ebx* (*add eax,ebx*, результат сложения запишется в регистр *eax*). Далее выводим результат. Так как для работы функции *sprintLF* в регистр *eax* должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра *eax* в переменную *buf1* (*mov [buf1],eax*), а затем запишем адрес переменной *buf1* в регистр *eax* (*mov eax,buf1*) и вызовем функцию *sprintLF*.

```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,'6'
8 mov ebx,'4'
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintLF
13 call quit

```

Рис 2.1.2: Демонстрация текста программы в файле

Скопируем *in\_out.asm* из каталога *lab05* в *lab06*

```

rayan@rayan-VirtualBox:~$ cd ~/work/arch-pc/lab05
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ cp in_out.asm ~/work/arch-pc/lab06
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$

```

Рис 2.1.3: Копирование файла для программы

Создадим исполняемый файл и запустим его

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ cd ~/work/arch-pc/lab06
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./lab6-1
j
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$

```

Рис 2.1.4: Создание исполняемого файла и проверка работы

В данном случае при выводе значения регистра *eax* мы ожидаем увидеть число 10. Однако результатом будет символ *j*. Это происходит потому, что код символа 6 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* (см. таблицу ASCII в приложении).

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы следующим образом:

- `mov eax,'6'`
- `mov ebx,'4'`

на строки

- `mov eax,6`
- `mov ebx,4`

```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintf
13 call quit

```

Рис 2.1.5: Демонстрация изменения программы

Создадим исполняемый файл и запустим его.

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./lab6-1

```

Рис 2.1.6: Создание исполняемого файла и проверка работы

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10, это *LF*, */n*. Который не отображается на выводе. Как отмечалось выше, для работы с числами в файле *in\_out.asm* реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

Создадим файл *lab6-2.asm* в каталоге *~/work/arch-pc/lab06* и введем в него текст программы из листинга 6.2.

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch lab6-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$

```

Рис 2.1.7: Создание файла

```

1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,'6'
8 mov ebx,'4'
9 add eax,ebx
10 call sprintLF
11
12 call quit

```

Рис 2.1.8: Демонстрация программы

Создадим исполняемый файл и запустим его.

```

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

```

Рис 2.1.9: Создание исполняемого файла и проверка работы

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 отличии от программы из листинга 6.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. Заменяем строки

- `mov eax,'6'`
- `mov ebx,'4'`

на строки

- `mov eax,6`
- `mov ebx,4`

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit

```

Рис 2.1.10: Демонстрация измененной программы

Создадим исполняемый файл и запустим его.

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

Рис 2.1.11: Создание исполняемого файла и проверка работы

В результате мы получили число 10

Заменяем функцию *iprintLF* на *iprint*. Создадим исполняемый файл и запустим его.

```
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./lab6-2
10rayan@rayan-VirtualBox:~/work/arch-pc/lab06$
```

Рис 2.1.12: Создание исполняемого файла и проверка работы

В результате также мы получили число 10, но в случае *iprint* число выведено без красной строки

## 2.2 Выполнение арифметических операций в NASM

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

Создадим файл *lab6-3.asm* в каталоге `~/work/arch-pc/lab06`:

```
10rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch lab6-3.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$
```

Рис 2.2.1: Создание файла

Внимательно изучим текст программы из листинга 6.3 и введем в *lab6-3.asm*.

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
; вызов подпрограммы завершения

```

Рис 2.2.2: Демонстрация программы

Создадим исполняемый файл и запустим его.

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch lab6-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch lab6-3.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$

```

Рис 2.2.3: Создание исполняемого файла и проверка работы

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

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,4 ; EAX=4
13 mov ebx,6 ; EBX=6
14 mul ebx ; EAX=EAX*EBX
15 add eax,2 ; EAX=EAX+2
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,5 ; EBX=5
18 div ebx ; EAX=EAX/5, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Результат: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения

```

Рис 2.2.4: Демонстрация измененной программы

Создадим исполняемый файл и проверим его работу.

```

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

```

Рис 2.2.5: Создание исполняемого файла и проверка работы

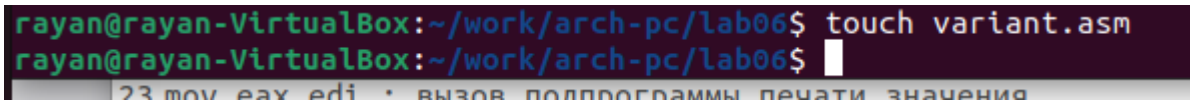
В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле:  $(S_n \bmod 20) + 1$ , где  $S_n$  – номер студенческого билета (В данном случае  $a \bmod b$  - это остаток от деления  $a$  на  $b$ ).
- вывести на экран номер варианта.



В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

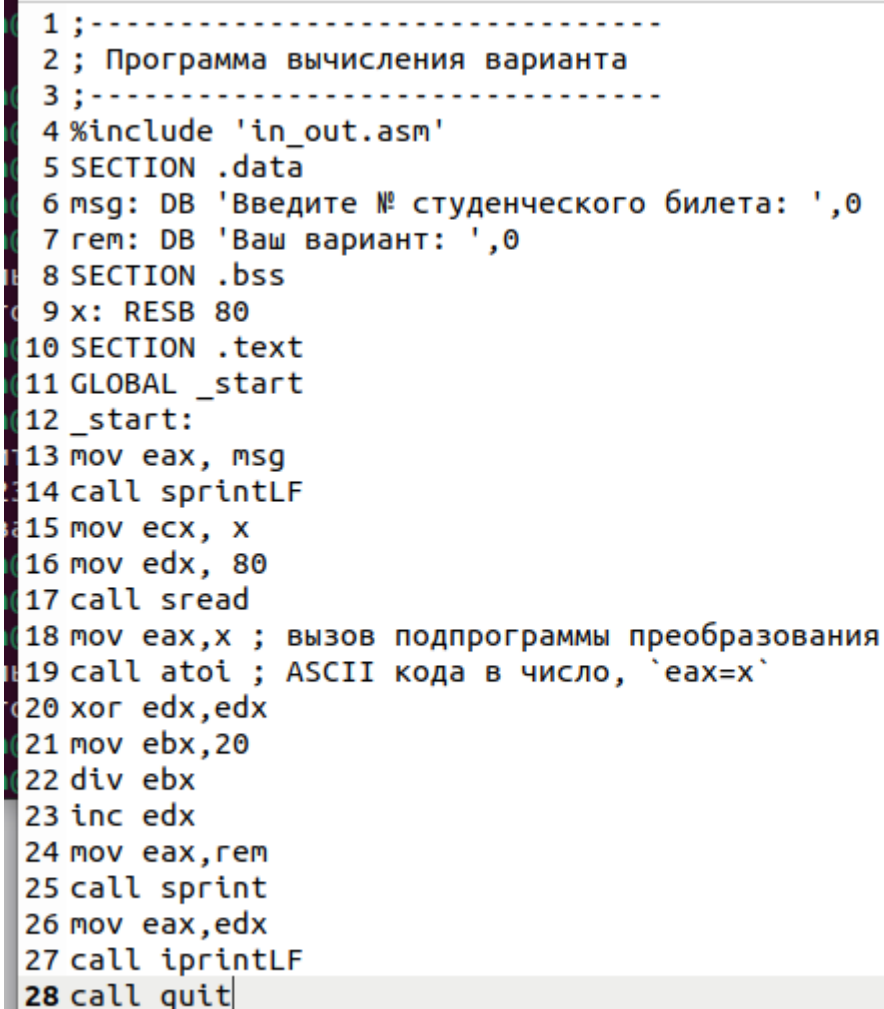
Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab06`:



```
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch variant.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$
```

Рис 2.2.6: Создание файла

Внимательно изучим текст программы из листинга 6.4 и введем в файл `variant.asm`.



```
1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите № студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintLF
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax,x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, `eax=x`
20 xor edx,edx
21 mov ebx,20
22 div ebx
23 inc edx
24 mov eax,rem
25 call sprint
26 mov eax,edx
27 call iprintLF
28 call quit
```

Рис 2.2.7: Демонстрация программы

Создадим исполняемый файл и запустим его.

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch variant.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
113221440
Ваш вариант: 1
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$

```

Рис 2.2.8: Создание исполняемого файла и проверка работы

## 2.3 Вопросы

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

- 1) Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

```

call sprint
mov eax,ret

```

- 2) Для чего используются следующие инструкции?

```

mov esx, x - перемещает адрес вводимой строки в esx
mov edx, 80 - записывает длину строки в регистр edx
call sread- вызывает подпрограммы, которые обеспечивают ввод сообщения с помощью клавиатуры

```

- 3) Для чего используется инструкция "call atoi"?

Она используется для вызова подпрограммы, которая преобразует ASCII код символа в целое число, записывая его в результат регистра "eax"

- 4) Какие строки листинга 6.4 отвечают за вычисления варианта?

```

xor edx, edx ; обнуление ebx для div
mov ebx, 10 ; ebx=10
div ebx ; eax = eax/10, edx - остаток от деления
inc edx ; edx=edx+1

```

- 5) В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"?

При div ebx остаток от деления записывается в edx

- 6) Для чего используется инструкция "inc edx"?

inc edx увеличивает значение регистра edx на +1

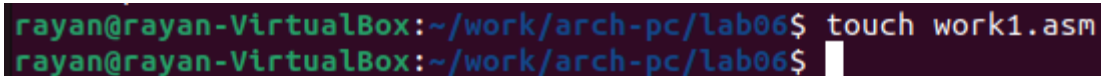
7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

```
mov eax, edx  
call iprintLF
```

### 3 Самостоятельная работа

*Задание №1 Написать программу вычисления выражения  $y=f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3. При выполнении задания преобразовывать (упрощать) выражения для  $f(x)$  нельзя. При выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е.  $5 : 2 = 2$ )*

Создадим новый файл для задания и напомним программу для  $f(x)=(10+2x)/3$



```
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ touch work1.asm  
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$
```

Рис 3.1.1: Создание файла

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: db 'Введите значение x: ', 0
5 rem: db 'Результат выражения: ', 0
6
7 SECTION .bss
8 x: resb 80
9
10 SECTION .text
11 GLOBAL _start
12
13 _start:
14 mov eax, msg
15 call sprintf
16
17 mov ecx, x
18 mov edx, 80
19 call sread
20
21 mov eax, x
22 call atoi
23
24 mov ebx, eax
25
26 mov eax, 2
27 mul ebx
28
29 add eax, 10
30 mov ecx, 3
31 div ecx
32
33 mov edi, eax
34
35 mov eax, rem
36 call sprintf

```

Рис 3.1.2: Демонстрация программы

Проверим программу

```
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf work1.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o work1 work1.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./work1
Введите значение x:
1
Результат выражения: 4
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$ ./work1
Введите значение x:
10
Результат выражения: 10
rayan@rayan-VirtualBox:~/work/arch-pc/lab06$
```

*Рис 3.1.3: Проверка программы*

Загрузим все файлы на github по окончании лаб. работы

## **4 Выводы**

Я освоил арифметические инструкции языка ассемблера NASM и приобрел практические навыки по работе с арифметикой в NASM.