

РУДН. Архитектура компьютеров

Отчёт по лабораторной работе №6

Косинов Никита Андреевич, НПИМбв-02-20

Содержание

1	Цель работы	5
2	Ход работы	6
3	Вывод результата вычислений в стандартный вывод	7
4	Вычисление простейшего выражения на ассемблере	13
5	Применение вычислений для практических задач	17
6	Самостоятельная работа	21
7	Выводы	25

Список иллюстраций

3.1	Создание файла	7
3.2	Программа сложения символов и вывода по коду	7
3.3	Презультат исполнения	8
3.4	Программа сложения чисел и вывода по коду	9
3.5	Результат исполнения	9
3.6	Программа сложения символов и вывода результата	9
3.7	Результат исполнения	10
3.8	Программа сложения чисел и вывода результата	10
3.9	Результат исполнения	10
3.10	Программа с выводом без переноса на новую строку	11
3.11	Результат исполнения	11
4.1	Программа вычисления А	13
4.2	Результат исполнения	14
4.3	Программа вычисления В	15
4.4	Запуск менеджера	16
5.1	Программа вычисления варианта	18
5.2	Результат исполнения	20
6.1	Программа вычисления выражения $f(x)$	22
6.2	Результат исполнения	24

Список таблиц

1 Цель работы

ЭВМ - электронно вычислительная машина, а это значит, что в первую очередь компьютер необходим для вычислений. Поэтому основная задача любого языка программирования, идущая после ввода/вывода сообщений на экран - это предоставить возможность арифметических действий. Исключением не является и язык ассемблер.

Поэтому, цель данной работы - изучить команды, позволяющие проделывать элементарные арифметические действия на ассемблере **NASM** такие, как сложение, вычитание, умножение и деление с остатком.

2 Ход работы

Лабораторная работа выполнена с использованием консоли **ОС Linux** и языка программирования ассемблера **NASM**.

1. Вывод результата вычислений в стандартный вывод;
2. Вычисление простейшего выражения на ассемблере;
3. Применение вычислений для практических задач.

В конце выполнена самостоятельная работа.

3 Вывод результата вычислений в стандартный вывод

Чтобы разобраться, как работает вывод на ассемблере, используем в качестве примера операцию сложения **add**, применив её к различным вариантам данных.

1. Создадим рабочий каталог и файл программы, в которой будем проводить эксперименты.

```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ gedit labs/lab05/report/report.md
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ cd labs/lab06
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-1.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 3.1: Создание файла

2. Напишем программу, вычисляющую сумму символов '4' и '6'. Программа выведет результат командой **sprintf** из подключаемого файла **in_out.asm**.



```
lab6-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
Сохранить

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

Рис. 3.2: Программа сложения символов и вывода по коду

```
%include 'in_out.asm'
```

```
SECTION .bss
```

```
    buf1: RESB 80
```

```
SECTION .text
```

```
    GLOBAL _start
```

```
_start:
```

```
    mov eax,6
```

```
    mov ebx,4
```

```
    add eax,ebx
```

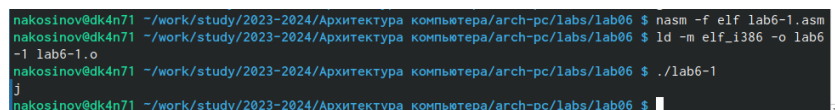
```
    mov [buf1],eax
```

```
    mov eax,buf1
```

```
    call sprintfLF
```

```
    call quit
```

3. Скомпилируем и запустим написанную программу.



```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-1.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6
-1 lab6-1.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-1
j
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 3.3: Результат исполнения

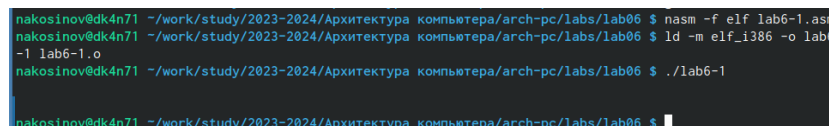
4. Мы видим, что результат исполнения - это символ 'j'. Программа так сработала, потому что при сложении символов она складывает их двоичные коды. Код символа '4' в системе *ASCII* равен 00110100, а код символа '6' - 00110110. Суммой этих кодов будет 01101010 (проверить это можно, например, сложив столбиком), что является кодом символа 'j'.
5. Изменим программу, убрав апострофы, тем самым заменив символы '4' и '6' в регистрах на соответствующие символы.



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

Рис. 3.4: Программа сложения чисел и вывода по коду

6. Скомпилируем полученную программу, запустим и посмотрим на результат.



```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-1.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6
-l lab6-1.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-1
10
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 3.5: Результат исполнения

7. На первый взгляд, в консоли ничего не напечатано. Но это не так - обратим внимание на лишний образовавшийся отступ. Действительно, сумма чисел 4 и 6 равна 10, но команду вывода мы не поменяли, т.е. она показывает символ с кодом 10, а это как раз перенос каретки.
8. Снова изменим нашу программу. На этот раз добавим апострофы, чтобы складывать именно символы, но изменим вывод на **iprintLF**.



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

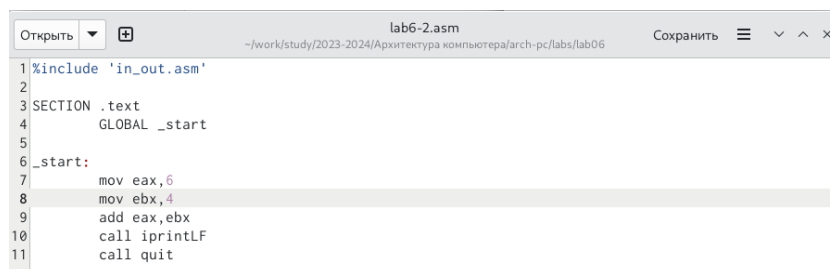
Рис. 3.6: Программа сложения символов и вывода результата

9. Скомпилируем и запустим.

```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
106
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 3.7: Результат исполнения

10. На терминале получаем число 106. Действительно, **iprint** выводит уже численный результат операции, но складывали мы символы, а операция **add** при этом складывает их коды. Тем самым мы получаем в результате 01101010, как в первом случае, но теперь мы выводим этот код как число, причём в десятичной записи: $01101010 = 2^1 + 2^3 + 2^5 + 2^6 = 2 + 8 + 32 + 64 = 106$.
11. Снова изменим программу, оставив численный вывод и заменив символы на числа.



```
lab6-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
Сохранить

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

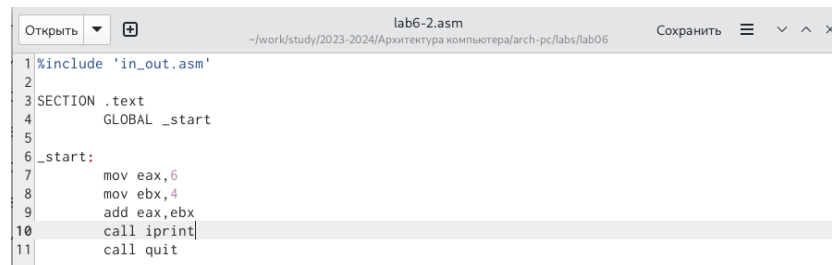
Рис. 3.8: Программа сложения чисел и вывода результата

12. Скомпилируем и запустим.

```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
106
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 3.9: Результат исполнения

13. Мы, наконец, получаем результат сложения чисел 4 и 6.
14. В подключаемом файле **in_out.asm** наряду с командами **sprintLF** и **iprintLF** есть команды **sprint** и **iprint**. Чтобы понять разницу в их работе, изменим команду вывода в последней программе.



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

Рис. 3.10: Программа с выводом без переноса на новую строку

```
%include 'in_out.asm'
```

```
SECTION .text
```

```
    GLOBAL _start
```

```
_start:
```

```
    mov eax, 6
```

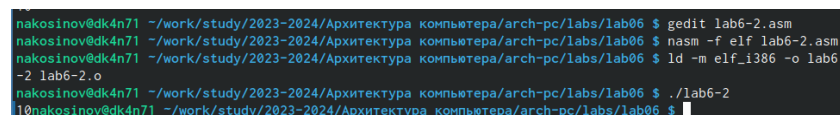
```
    mov ebx, 4
```

```
    add eax, ebx
```

```
    call iprint
```

```
    call quit
```

15. Скомпилируем и запустим.



```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
10nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

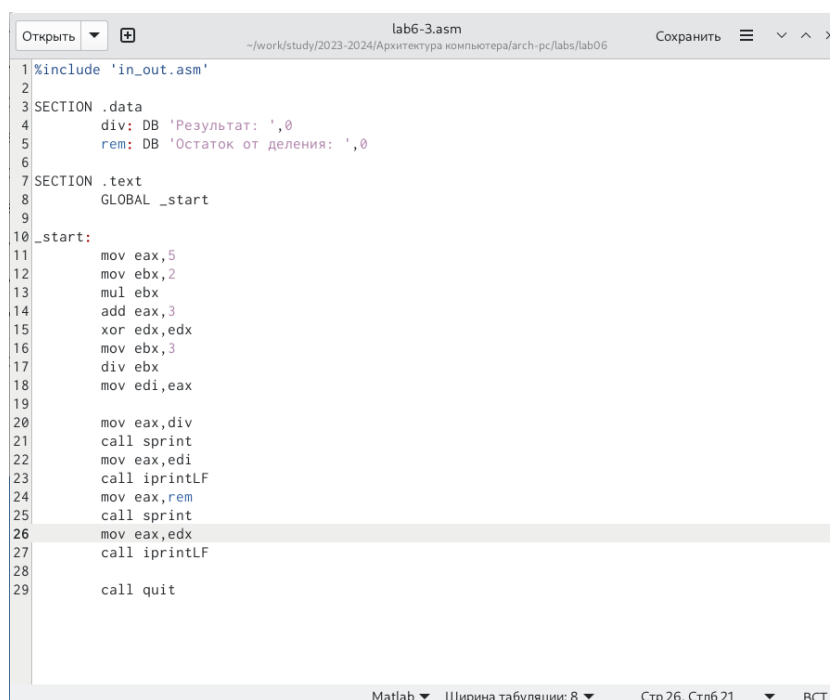
Рис. 3.11: Результат исполнения

16. Результат: без добавленного **LF** в конце команды вывода, мы не получаем перенос строки на новую.

4 Вычисление простейшего выражения на ассемблере

Выводить результат в численном виде мы научились, теперь разберём, как этот результат получить. Предлагается с помощью языка посчитать значения выражения $A : \frac{5 \times 2 + 3}{3}$.

1. Напишем программу, вычисляющую данное выражение.



```
1 %include 'in_out.asm'
2
3 SECTION .data
4     div: DB 'Результат: ',0
5     rem: DB 'Остаток от деления: ',0
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,5
12     mov ebx,2
13     mul ebx
14     add eax,3
15     xor edx,edx
16     mov ebx,3
17     div ebx
18     mov edi,eax
19
20     mov eax,div
21     call sprint
22     mov eax,edi
23     call iprintLF
24     mov eax,rem
25     call sprint
26     mov eax,edx
27     call iprintLF
28
29     call quit
```

Рис. 4.1: Программа вычисления A

2. Разберём, что происходит в коде:

2.1. Записываем 5 и 2 в регистры **eax** и **ebx**.

2.2. Увеличиваем число из **eax**, т.е. 5, в число раз, записанное в **ebx**, т.е. 2. Получаем 10.

2.3. Увеличиваем **eax**, равное 10, на 3.

2.4. Обнуляем значение, оставшееся с прошлых разов в регистре **edx**.

2.5. Записываем в **ebx** делитель - число 3.

2.6. Делим число **eax** (13) на **ebx**. Неполное частное от деления (4) остаётся в **eax**, а остаток уходит в регистр **edx**.

2.7. Для вывода нам нужен свободный регистр **eax**, поэтому мы переписываем полученное частное в другой - **edi**, дабы его не потерять.

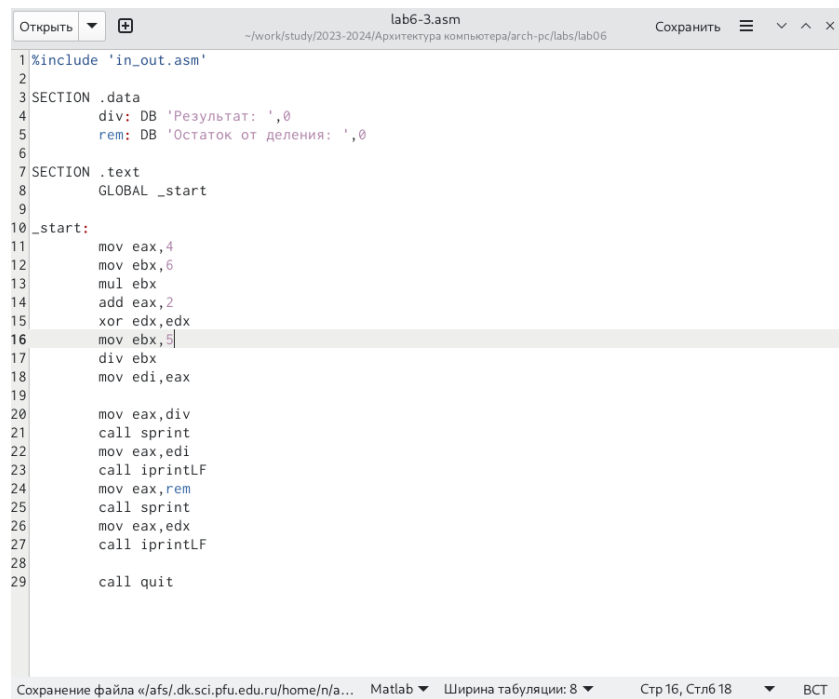
2.8. Последовательно выводим сообщения на экран и полученные значения.

3. Результат после компиляции.

```
10nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-3.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit lab6-3.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-3.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 4.2: Результат исполнения

4. Изменим код программы для вычисления другого выражения: $B : \frac{4 \times 6 + 2}{5}$.



```
1 %include 'in_out.asm'
2
3 SECTION .data
4     div: DB 'Результат: ',0
5     rem: DB 'Остаток от деления: ',0
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,6
13     mul ebx
14     add eax,2
15     xor edx,edx
16     mov ebx,5
17     div ebx
18     mov edi,eax
19
20     mov eax,div
21     call sprint
22     mov eax,edi
23     call iprintLF
24     mov eax,rem
25     call sprint
26     mov eax,edx
27     call iprintLF
28
29     call quit
```

Сохранение файла «/afs/dk.sci.pfu.edu.ru/home/n/a... Matlab Ширина табуляции: 8 Стр 16, Стр 18 ВСТ

Рис. 4.3: Программа вычисления В

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
    div: DB 'Результат: ',0
```

```
    rem: DB 'Остаток от деления: ',0
```

```
SECTION .text
```

```
    GLOBAL _start
```

```
_start:
```

```
    mov eax,4
```

```
    mov ebx,6
```

```
    mul ebx
```

```
    add eax,2
```

```
    xor edx,edx
```

```

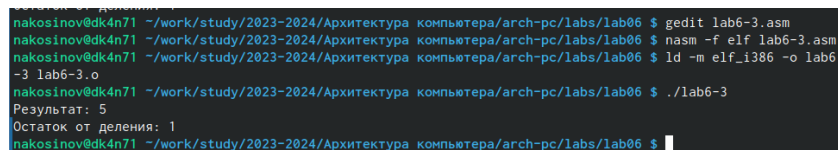
mov ebx,5
div ebx
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit

```

5. Скомпилируем и запустим. Убедимся, что программа выполнена корректно.



```

nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit lab6-3.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-3.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $

```

Рис. 4.4: Запуск менеджера

5 Применение вычислений для практических задач

Разберём вариант задачи, где вычисления нужны для какой-то практической цели. Для большинства таких задач нам необходимо посчитать не просто значение какого-то числа, но в зависимости от введённых пользователем данных.

1. Напишем программу, вычисляющую номер варианта работы по номеру студенческого билета, как остаток от деления этого номера на 20 (количество вариантов).

```
1 %include 'in_out.asm'
2
3 SECTION .data
4     msg: DB 'Введите № студенческого билета: ',0
5     rem: DB 'Ваш вариант: ',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     xor edx, edx
25     mov ebx, 20
26     div ebx
27     inc edx
28     mov eax, rem
29     call sprintf
30
31     mov eax, edx
32     call iprintf
33
34     call quit
```

Рис. 5.1: Программа вычисления варианта

```
%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
```

```
call atoi
```

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

```
mov eax,rem
```

```
call sprintf
```

```
mov eax,edx
```

```
call iprintLF
```

```
call quit
```

2. Разберём код:

2.1. Сохраняем стандартные сообщения для общения с пользователем в переменные **msg** и **rem**.

2.2. Выведем сообщение, предлагающие ввести номер билета.

2.3. Считываем номер билета, предварительно подготовив регистры **ecx** и **edx**.

2.4. Переносим билет в рабочий регистр **eax**, с которым и будем проводить операции. Функцией **atoi** из подключаемого файла преобразуем объект в

регистре из кода в число.

2.5. Считаем остаток от деления на 20. Он сохраняется в заранее обнулённом регистре **edx**.

2.6. Остаток при делении на 20 может принимать значения от 0 до 19, а варианты заданий - от 1 до 20. Чтобы нормировать эти два набора, увеличим остаток на единицу инструкцией **inc**.

2.7. Выведем на экран сообщение с результатом - номером варианта задания.

3. Скомпилируем и запустим. Ввожу номер своего студенческого билета: 1032189480.

```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch variant.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit variant.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf variant.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o variant variant.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./variant
Введите № студенческого билета:
1032189480
Ваш вариант: 1
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 5.2: Результат исполнения

6 Самостоятельная работа

При запуске предыдущей программы, мне по номеру билета выпал 1-й вариант. Действительно, 1032189480 делится нацело на 20. Поэтому самостоятельная работа состоит в вычислении значений выражения для $f(x) = \frac{10+2x}{3}$ с точностью до остатка для значений $x = 1$ и $x = 10$.

1. Напишем код программы. Добавим приглашающие сообщения и комментарии.

```

1 %include 'in_out.asm'
2
3 SECTION .data
4     polynom: DB 'Вариант1. f(x)=(10+2x)/3',0
5     zapros: DB 'Введите x: ',0
6     rez: DB 'f(x) = ',0
7
8 SECTION .bss
9     x: RESB 80
10
11 SECTION .text
12     GLOBAL _start
13
14 _start:
15     mov eax, polynom
16     call sprintf      ; Вывод формулы
17
18     mov eax, zapros
19     call sprintf      ; Запрос переменной
20
21     mov ecx, x
22     mov edx, 80
23     call sread        ; Ввод аргумента
24
25     mov eax, x
26     call atoi         ; Преобразование аргумента в число
27
28     mov ebx, 2
29     mul ebx
30     add eax, 10
31     xor edx, edx
32     mov ebx, 3
33     div ebx           ; Подсчёт выражения
34     mov edx, eax
35
36     mov eax, rez
37     call sprintf      ; Вывод сообщения
38
39     mov eax, edx
40     call iprintfLF    ; Вывод результата
41
42     call quit         ; Выход

```

Сохранение файла «/afs/dk.sci.pfu.edu.ru/home/n/a... Matlab Ширина табуляции: 8 Стр 4, Стлб 48 ВСТ

Рис. 6.1: Программа вычисления выражения $f(x)$

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
    polynom: DB 'Вариант1. f(x)=(10+2x)/3',0
```

```
    zapros: DB 'Введите x: ',0
```

```
    rez: DB 'f(x) = ',0
```

```
SECTION .bss
```

```
    x: RESB 80
```

```
SECTION .text
```

```
    GLOBAL _start
```

```

_start:
    mov eax, polynom
    call sprintf      ; Вывод формулы

    mov eax, zapros
    call sprintf      ; Запрос переменной

    mov ecx, x
    mov edx, 80
    call sread        ; Ввод аргумента

    mov eax, x
    call atoi          ; Преобразование аргумента в число

    mov ebx, 2
    mul ebx
    add eax, 10
    xor edx, edx
    mov ebx, 3
    div ebx            ; Подсчёт выражения
    mov edx, eax

    mov eax, rez
    call sprintf      ; Вывод сообщения

    mov eax, edx
    call sprintf      ; Вывод результата

```

`call quit` ; Выход

2. Скомпилируем и запустим. Убедимся, что результат выполнения верный.

Действительно, $\frac{10+2 \times 1}{3} = \frac{12}{3} = 4$ и $\frac{10+2 \times 10}{3} = \frac{30}{3} = 10$.

```
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit sr.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf sr.asm
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o sr s
r.o
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./sr
Вариант1. f(x)=(10+2x)/3
Введите x: 1
f(x) = 4
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./sr
Вариант1. f(x)=(10+2x)/3
Введите x: 10
f(x) = 10
nakosinov@dk4n71 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 6.2: Результат исполнения

7 Выводы

В ходе данной лабораторной работы мы научились пользоваться основными арифметическими операциями, выводить результат вычислений на экран в десятичной форме с помощью подключаемого файла, а также рассмотрели, как работает операция сложения для нечисленных объектов.