

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Аннагулыев Арслан Мухаммедович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	14
4.2.1	Ответы на вопросы по программе	17
4.3	Выполнение заданий для самостоятельной работы	18
5	Выводы	22
6	Список литературы	23

Список иллюстраций

4.1	Создание директории	8
4.2	Создание файла	8
4.3	Создание копии файла	8
4.4	Редактирование файла	9
4.5	Запуск исполняемого файла	9
4.6	Редактирование файла	10
4.7	Запуск исполняемого файла	10
4.8	Создание файла	10
4.9	Редактирование файла	11
4.10	Запуск исполняемого файла	11
4.11	Редактирование файла	12
4.12	Запуск исполняемого файла	12
4.13	Редактирование файла	13
4.14	Запуск исполняемого файла	13
4.15	Создание файла	14
4.16	Редактирование файла	14
4.17	Запуск исполняемого файла	14
4.18	Изменение программы	15
4.19	Запуск исполняемого файла	15
4.20	Создание файла	16
4.21	Редактирование файла	16
4.22	Запуск исполняемого файла	17
4.23	Создание файла	18
4.24	Написание программы	19
4.25	Запуск исполняемого файла	20
4.26	Запуск исполняемого файла	20

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного

результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты `cd`.

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab05/report$ cd
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ mkdir -p /work/study/2024-2025/Архитектура компьютера/arch-pc/lab06
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ cd
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 4.2).

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-1.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

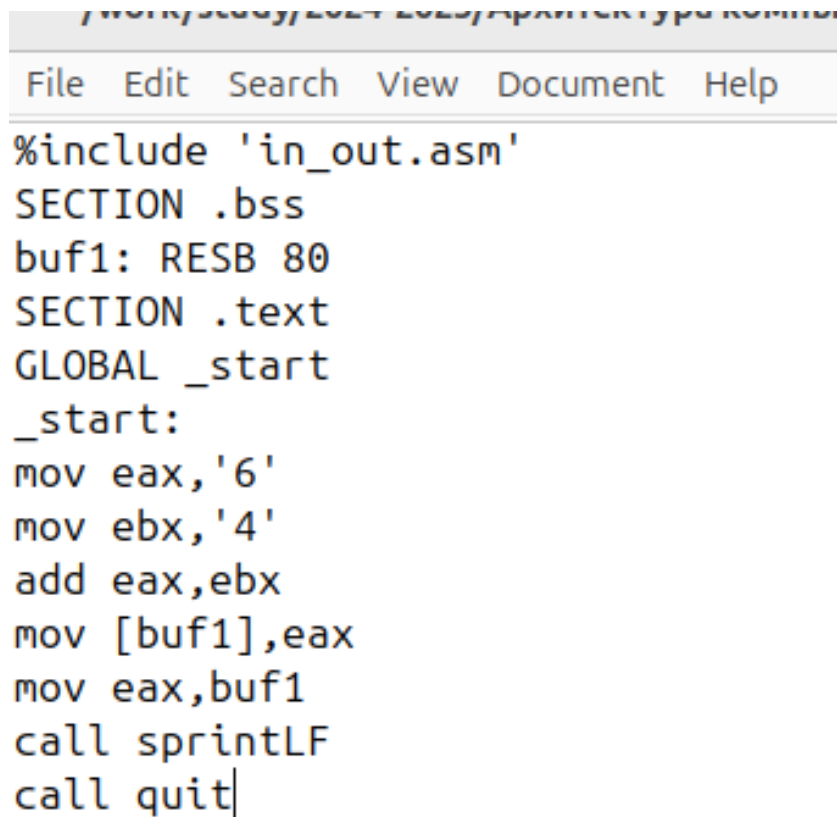
Рис. 4.2: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 4.3).

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ cp ~/Downloads/in_out.asm in_out.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$
```

Рис. 4.3: Создание копии файла

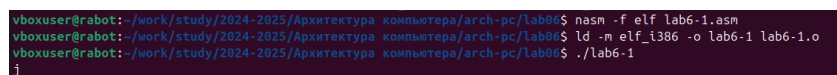
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.4).

A screenshot of a text editor window. The title bar at the top reads "/work/study/2024-2025/Архитектура компьютера". The menu bar includes "File", "Edit", "Search", "View", "Document", and "Help". The code content is as follows:

```
%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 sprintLF
call quit|
```

Рис. 4.4: Редактирование файла

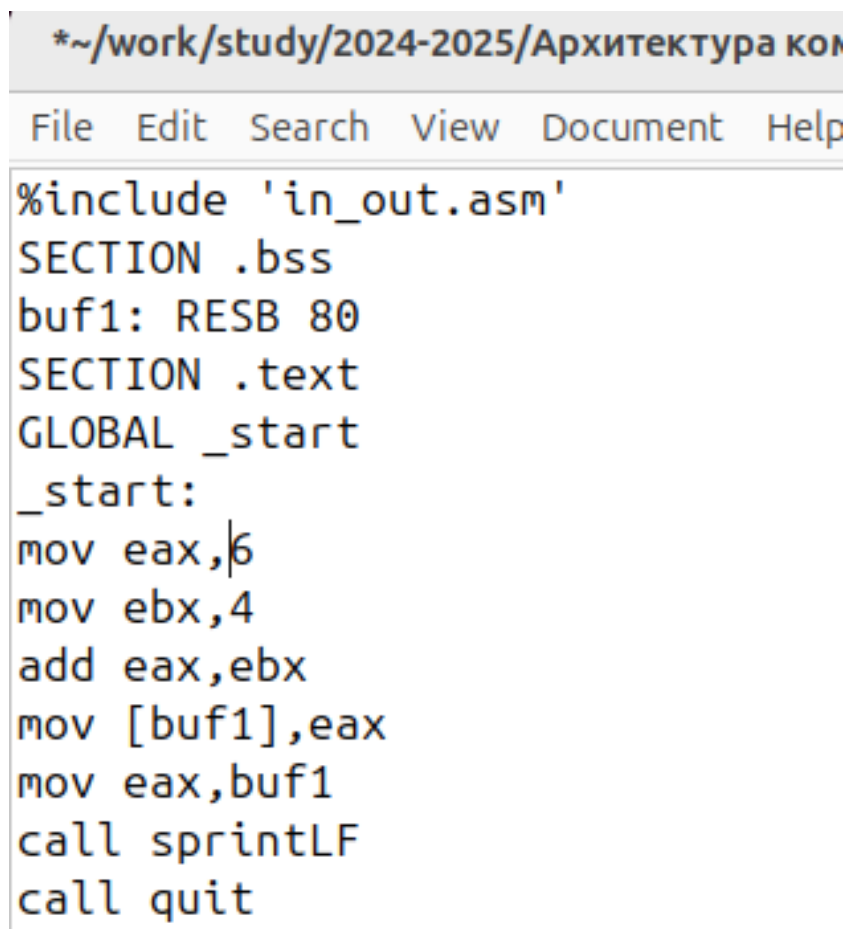
Создаю исполняемый файл программы и запускаю его (рис. 4.5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

A screenshot of a terminal window. The prompt is "vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab6". The commands and their outputs are:

```
nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
j
```

Рис. 4.5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.6).

A screenshot of a text editor window. The title bar at the top reads '*~/work/study/2024-2025/Архитектура комп'. Below the title bar is a menu bar with 'File', 'Edit', 'Search', 'View', 'Document', and 'Help'. The main text area contains the following assembly code:

```
%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 sprintLF
call quit
```

Рис. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

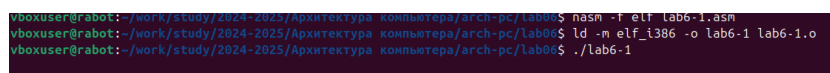
A screenshot of a terminal window with a dark background. The prompt is 'vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06\$'. The user enters three commands: 'nasm -f elf lab6-1.asm', 'ld -m elf_i386 -o lab6-1 lab6-1.o', and './lab6-1'. The output of the last command is a single line containing the number '10'.

Рис. 4.7: Запуск исполняемого файла

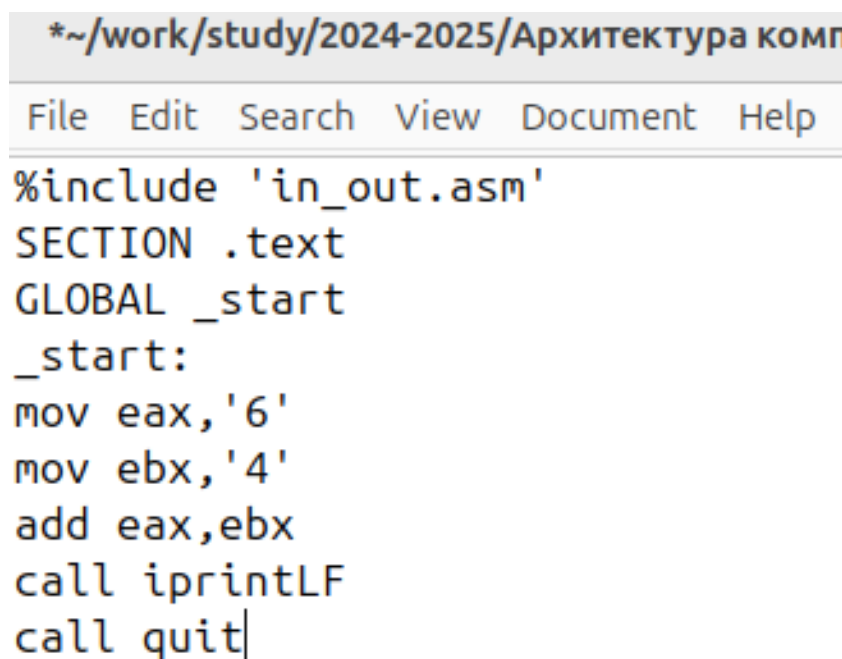
Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 4.8).

A screenshot of a terminal window with a dark background. The prompt is 'vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06\$'. The user enters the command 'touch lab6-2.asm'.

Рис. 4.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис.

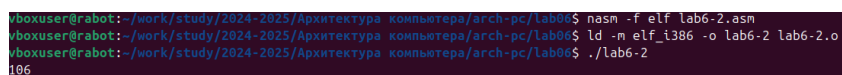
4.9).



```
*~/work/study/2024-2025/Архитектура комп  
File Edit Search View Document Help  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprintLF  
call quit|
```

Рис. 4.9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.10). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.



```
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm  
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2  
106
```

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.11).

```
~/work/study/2024-2025/Архит
File Edit Search View Docu
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.12).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-2.asm
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-2
10
```

Рис. 4.12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.13).

```
~/work/study/2024-2025/Арх  
File Edit Search View Do  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Рис. 4.13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.14). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ nasm -f elf lab6-2.asm  
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ld -n elf_i386 -o lab6-2 lab6-2.o  
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$ ./lab6-2  
10vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab6$
```

Рис. 4.14: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.15).

```
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch lab6-3.asm
```

Рис. 4.15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.16).

```
*~/work/study/2024-2025/Архитектура компьютер
File Edit Search View Document Help
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
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
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.17).

```
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
yboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.18).

```

*~/work/study/2024-2025/Архитектура комп
File Edit Search View Document Help
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, 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
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-3.asm
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.19: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.20).

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ touch variant.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ mousepad variant.asm
```

Рис. 4.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.21).

```
%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 sprintLF
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
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
```

Рис. 4.21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.


```

vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf variant.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o variant variant.o
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./variant
Введите № студенческого билета:
1032244499
Ваш вариант: 20

```

Рис. 4.22: Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

1. Строка “mov eax,rem” и строка “call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эти инструкции используются для чтения строки с вводом данных от пользователя. Начальный адрес строки сохраняется в регистре esx, а количество символов в строке (максимальное количество символов, которое может быть считано) сохраняется в регистре edx. Затем вызывается процедура sread, которая выполняет чтение строки.
3. Инструкция “call atoi” используется для преобразования строки в целое число. Она принимает адрес строки в регистре eax и возвращает полученное число в регистре eax.
4. Строка “xor edx,edx” обнуляет регистр edx перед выполнением деления. Строка “mov ebx,20” загружает значение 20 в регистр ebx. Строка “div ebx” выполняет деление регистра eax на значение регистра ebx с сохранением частного в регистре eax и остатка в регистре edx.
5. Остаток от деления записывается в регистр edx.
6. Инструкция “inc edx” используется для увеличения значения в регистре edx на 1. В данном случае, она увеличивает остаток от деления на 1.
7. Строка “mov eax,edx” передает значение остатка от деления в регистр eax. Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 4.23).

```
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютер  
а/arch-pc/lab06$ touch lab6-4.asm  
vboxuser@rabot:~/work/study/2024-2025/Архитектура компьютер  
а/arch-pc/lab06$ mousepad lab6-4.asm
```

Рис. 4.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $x^3 \cdot \frac{1}{3} + 21$ (рис. 4.24). Это выражение было под вариантом 8.

```
File Edit Search View Document H
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
div: DB 'Результат:',0
SECTION .bss
rez: RESB 80
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF

mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi

mov ebx,eax
mul eax
mul ebx
xor ebx,ebx
mov ebx,3
div ebx
xor ebx,ebx
add eax,21
mov[rez],eax

mov eax,div
call sprint
mov eax,[rez]
call iprintLF
call quit
```

Рис. 4.24: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.25). При вводе значения 1, вывод - 21.

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ nasm -f elf lab6-4.asm
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
Введите x:
1
Результат:21
```

Рис. 4.25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 4.26). Программа отработала верно.

```
vboxuser@rabort:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab06$ ./lab6-4
Введите x:
3
Результат:30
```

Рис. 4.26: Запуск исполняемого файла

****Листинг 4.1. Программа для вычисления значения выражения $x^3 + 1/3 + 21$.****

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
div: DB 'Результат:',0

SECTION .bss
rez: RESB 80
x: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call printf

mov ecx, x
mov edx, 80
call sread
```

```
mov eax,x
call atoi

mov ebx,eax
mul eax
mul ebx
xor ebx,ebx
mov ebx,3
div ebx
xor ebx,ebx
add eax,21
mov[rez],eax

mov eax,div
call sprint
mov eax,[rez]
call iprintLF
call quit
```

5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Таблица ASCII