РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

по лабораторной работе № 7

дисциплина: Архитектура компьютера

Студент: Клименко Кирилл Русланович

Группа: НММбд-02-24

МОСКВА

2025г.

Содержание

Список иллюстраций

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 №7 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты cd.

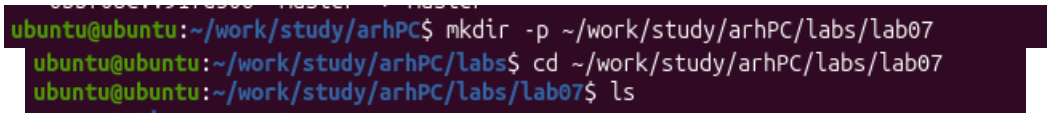

Рис. 4.1: Создание директории

С помощью утилиты touch создаю файл lab7-1.asm (рис. 4.2).

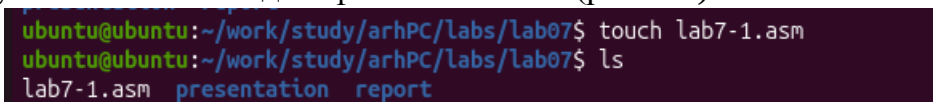

Рис. 4.2: Создание файла

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

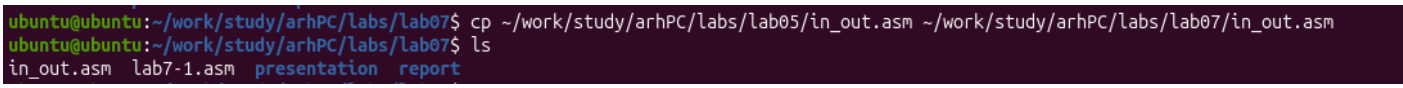

Рис. 4.3: Создание копии файла

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.4).

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.

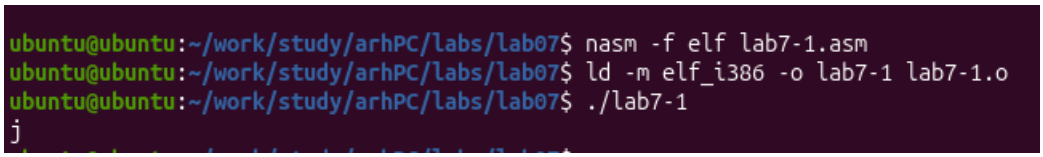

Рис. 4.5: Запуск исполняемого файла

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

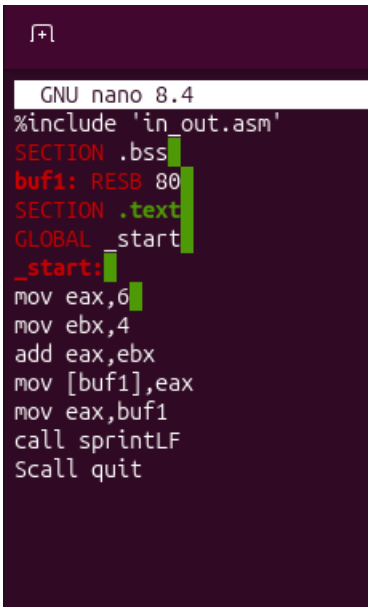

Рис. 4.6: Редактирование файла

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.

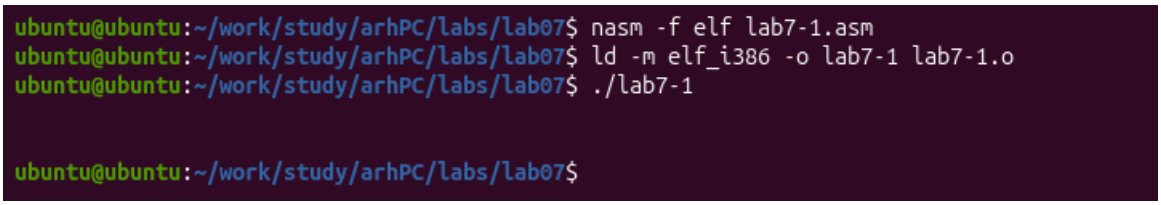

Рис. 4.7: Запуск исполняемого файла

Создаю новый файл lab7-2.asm с помощью утилиты touch (рис. 4.8).

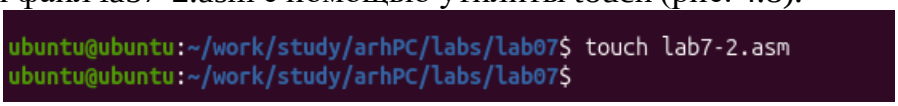

Рис. 4.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.9).

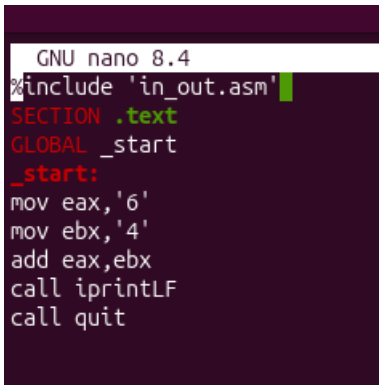

Рис. 4.9: Редактирование файла

Создаю и запускаю исполняемый файл lab7-2 (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”.

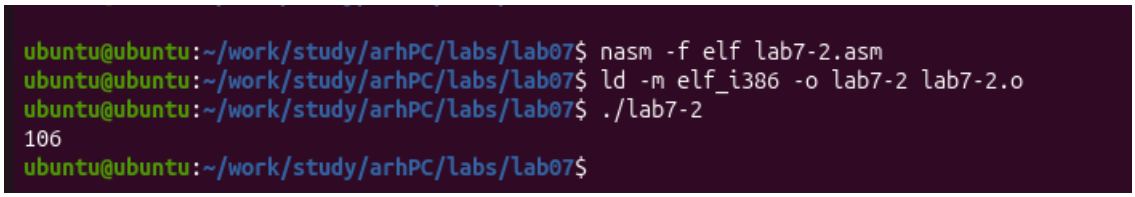

Рис. 4.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.11).

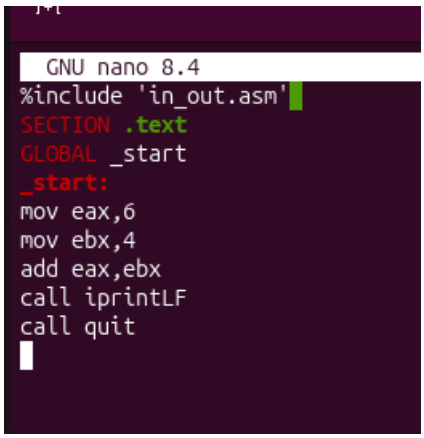

Рис. 4.11: Редактирование файла

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.

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ nasm -f elf lab7-2.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./lab7-2
10
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

Рис. 4.12: Запуск исполняемого файла

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

```
7 add eax,ebx
8 call iprint
9 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.

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ nasm -f elf lab7-2.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./lab7-2
10ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

Рис. 4.14: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab7-3.asm с помощью утилиты touch (рис. 4.15).

```
10ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ touch lab7-3.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ s
```

Рис. 4.15: Создание файла

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

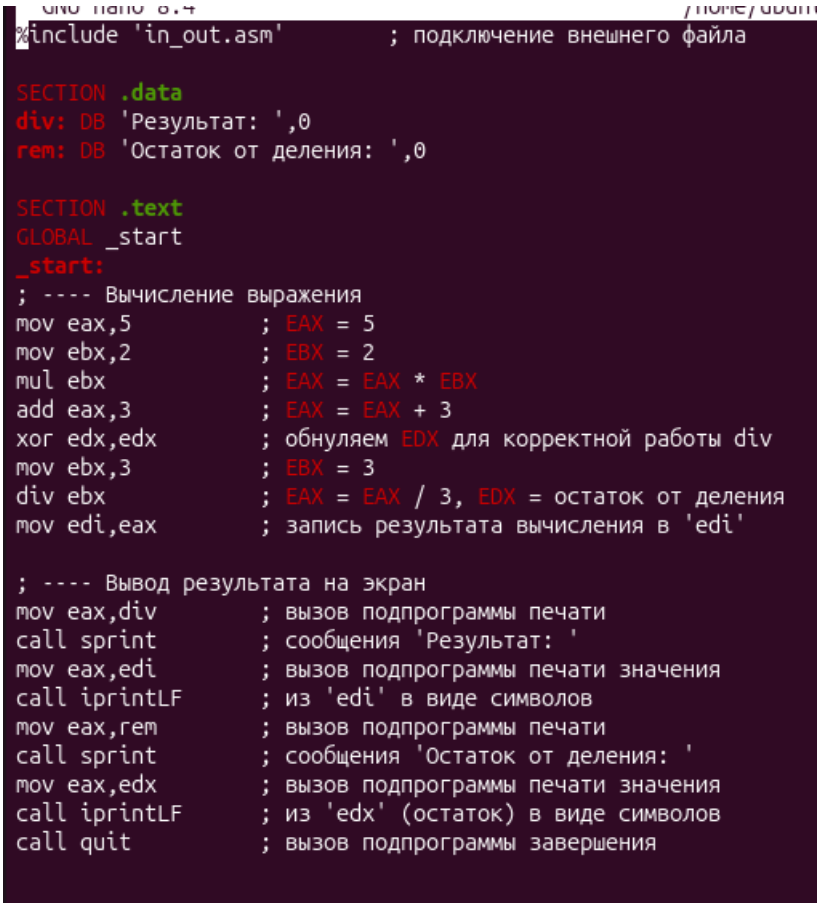

Рис. 4.16: Редактирование файла

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

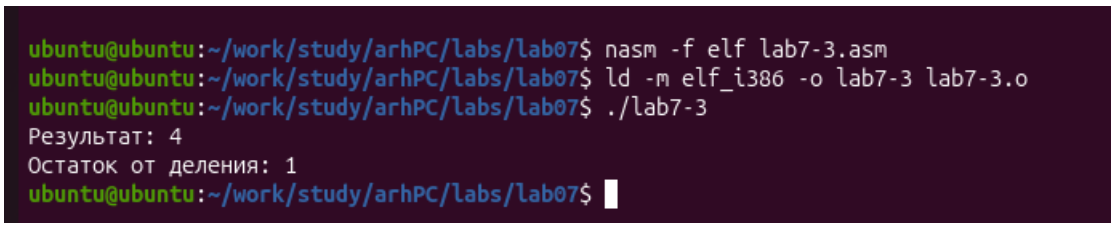

Рис. 4.17: Запуск исполняемого файла

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

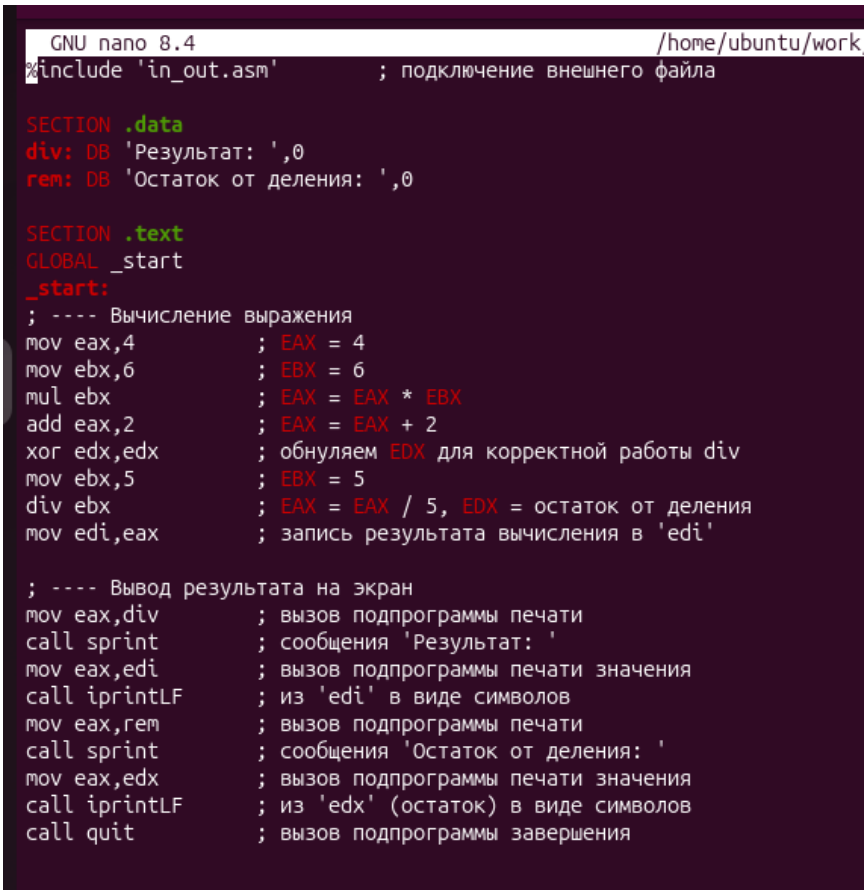

Рис. 4.18: Изменение программы

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

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ nasm -f elf lab7-3.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./lab7-3
Результат: 5
Остаток от деления: 1
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

Рис. 4.19: Запуск исполняемого файла

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

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ touch variant.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

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'                ; подключение внешнего файла

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          ; вывод приглашения
call sprintLF

mov ecx, x            ; буфер для ввода
mov edx, 80           ; длина буфера
call sread            ; читаем строку

mov eax, x            ; адрес введённой строки
call atoi             ; ASCII -> число, результат в eax

xor edx, edx
mov ebx, 20           ; делитель
div ebx               ; eax = номер варианта, edx = остаток

inc edx               ; варианты считаются с 1

mov eax, rem          ; печать сообщения
call sprint
mov eax, edx          ; печать номера варианта
call iprintLF

call quit             ; завершение программы
```

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.

```
ubuntu@ubuntu:~/work/study/arhPC$ cd ~/work/study/arhPC/labs/lab07
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ nasm -f elf variant.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ld -m elf_i386 -o variant variant.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./variant
Введите № студенческого билета:
1132246287
Ваш вариант: 8
```

Рис. 4.22: Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem
call sprint
```

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки:

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

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab7-4.asm с помощью утилиты touch (рис. 4.23).

```
Ваш вариант: 19
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ touch lab7-4
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

Рис. 4.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (11 + x) * 2 - 6 (рис. 4.24). Это выражение было под вариантом 8.

```
 GNU nano 8.4                                  /home/ubuntu/work/study/arhPC/labs/lab07/
%include 'in_out.asm'              ; подключение внешнего файла

SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss
x:   RESB 80                       ; переменная, значение которой будем вводить с клавиатуры

SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x                     ; вызов подпрограммы преобразования
call atoi                      ; ASCII кода в число, 'eax= x'

add eax, 11                    ; eax = eax + 11 = x + 11
mov ebx, 2
mul ebx                        ; EAX = EAX * EBX = (x + 11) * 2
add eax,-6                     ; eax = eax -6 = 2 * (x + 11) - 6

mov edi, eax                   ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран
mov eax, rem                   ; вызов подпрограммы печати
call sprint                    ; сообщение 'Результат: '
mov eax, edi                   ; вызов подпрограммы печати значения
call iprintLF                  ; из 'edi' в виде символов
                                                          [ Read 37 lines ]
```

Рис. 4.24: Написание программ

Создаю и запускаю исполняемый файл (рис. 4.25). При вводе значения 3, вывод - 22.

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ nasm -f elf lab7-4.asm
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./lab7-4
Введите значение переменной x: 3
Результат: 22
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

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.

```
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$ ./lab7-4
Введите значение переменной x: 1
Результат: 18
ubuntu@ubuntu:~/work/study/arhPC/labs/lab07$
```

Рис. 4.26: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения (11 + x) * 2 – 6.

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный ра
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимиого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax,11; eax = eax+11 = x + 11
mov ebx,2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax,-6; eax = eax-6 = (x+11)*2-6
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM, это оказалось крайне интересно.

6 Список литературы

1. Лабораторная работа №7
2. Таблица ASCII