МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) Институт информационных технологий

Кафедра Математического и программного обеспечения ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине Программирование на ассемблере

на тему Программирование на языке низкого уровня

Выполнил студент группы

1ПИб-01-1оп-21

01

*группа*

направления подготовки (специальности)

09.03.04 Программная инженерия

*шифр, наименование*

Трофимов Никита Владиславович

*фамилия, имя, отчество*

Руководитель

Виноградова Людмила Николаевна

*фамилия, имя, отчество*

Преподаватель

*должность*

Дата представления работы

«27» декабря 2022г. Заключение о допуске к защите

Оценка ,

количество баллов

Подпись преподавателя

Череповец, 2022

Аннотация

Курсовая работа по дисциплине «Программирование на ассемблере» на тему «Программирование на языке низкого уровня». В работе представлены материалы о развитии микропроцессоров, в частности Intel 8086, истории и особенностях языка ассемблера. Описание полного цикла разработки программы на данном языке, от технического задания до кодирования и тестирования, которая высчитывает выражение  при заданных пользователем через консоль N и j.

**ОГЛАВЛЕНИЕ**

[Введение 4](#_Toc122596602)

[1. Основная часть 7](#_Toc122596603)

[1.1. Изучение и описание предметной области 7](#_Toc122596604)

[1.2. Постановка задачи 8](#_Toc122596605)

[1.3. Выбор структур данных для решения поставленной задачи 8](#_Toc122596606)

[1.4. Логическое проектирование 9](#_Toc122596607)

[1.5. Физическое проектирование программного обеспечения 10](#_Toc122596608)

[1.6. Кодирование 10](#_Toc122596609)

[1.7. Тестирование 13](#_Toc122596610)

[Руководство пользователя 15](#_Toc122596611)

[Общие сведения о программе 15](#_Toc122596612)

[Описание установки 15](#_Toc122596613)

[Описание запуска 15](#_Toc122596614)

[Заключение 16](#_Toc122596615)

[Список литературы 17](#_Toc122596616)

[Приложение 1. Техническое задание 18](#_Toc122596617)

# Введение

Intel 8086 — первый [16-битный](https://intel.fandom.com/ru/wiki/16_%D0%B1%D0%B8%D1%82) микропроцессор компании [Intel](https://intel.fandom.com/ru/wiki/Intel), разрабатывавшийся с весны 1976 года и выпущенный 8 июня 1978 года. Реализованная в процессоре архитектура набора команд стала основой широко известной архитектуры [x86](https://intel.fandom.com/ru/wiki/X86). Процессоры этой архитектуры стали наиболее успешной линией процессоров Intel. Современные процессоры этой архитектуры сохраняют возможность выполнять все команды этого набора.

Рынок 8-разрядных микропроцессоров в конце 1970-х годов был переполнен, и Intel, оставляя попытки закрепиться на нём, выпускает свой первый 16-битный процессор. Процессор Intel 8086 представляет собой модернизированный процессор Intel 8080, и хотя разработчики не ставили перед собой цель достичь полной совместимости на программном уровне, большинство программ, написанных для Intel 8080, способны выполняться и на Intel 8086 после перекомпиляции. Новый процессор несёт в себе множество изменений, которые позволили значительно (в 10 раз) увеличить производительность по сравнению с предыдущим поколением процессоров компании.

Всего в процессоре Intel 8086 было 14 16-разрядных регистров: 4 регистра общего назначения (AX, BX, CX, DX), 2 индексных регистра (SI, DI), 2 указательных (BP, SP), 4 сегментных регистра (CS, SS, DS, ES), программный счётчик или указатель команды (IP) и регистр флагов (FLAGS, включает в себя 9 флагов). При этом регистры данных (AX, BX, CX, DX) допускали адресацию не только целых регистров, но и их младшей половины (регистры AL, BL, CL, DL) и старшей половины (регистры AH, BH, CH, DH), что позволяло использовать не только новое 16-разрядное ПО, но сохраняло совместимость и со старыми программами.

Расширение адресного пространства было одним из требований к новому процессору. В результате была выбрана компромиссная схема: исполнительный блок процессора оставлен 16-разрядным, а расширение адресного пространства сделано в блоке интерфейса шины (процессора путём реализации сегментной адресации памяти, увеличивающей разрядность шины адреса до 20 бит. Сегментная адресация памяти использует тот факт, что обращения к памяти со стороны процессора легко можно разделить на обращения к коду программы, обрабатываемым данным и стеку. Обращения к разным типам содержимого памяти отображаются на независимые области памяти в расширенном адресном пространстве — сегменты. В процессоре 8086 для такого отображения адресов используются четыре 16-битных сегментных регистра:

1. CS (Code Segment) — сегмент кода
2. DS (Data Segment) — сегмент данных
3. ES (Extra Segment) — дополнительный сегмент
4. SS (Stack Segment) — сегмент стека

Каждый сегментный регистр определяет адрес начала сегмента в памяти, при этом сегменты могут совпадать или пересекаться. По умолчанию регистр CS используется при выборке инструкций, регистр SS при выполнении операций со стеком, регистры DS и ES при обращении к данным. В случае обращения к данным сегментный регистр по умолчанию может быть изменён путём добавления перед кодом инструкции специального префикса замены сегмента.

При этом, в пределах сегмента используются 16-битные исполнительные адреса, хранящиеся в счётчике команд, указателе стека, либо вычисляемые в соответствии с видом адресации, заданном в коде инструкции. Фактически, процессору всегда доступны 4 области памяти размером 64 Кбайт каждая. Если такой объём оказывается недостаточен, в программу приходится включать логику управления сегментными регистрами. Эта логика может существенно замедлять обращение к памяти, поэтому компиляторы с языков высокого уровня для платформы x86 позволяют указывать модель памяти в соответствии с требуемыми объёмами кода и данных.

В 1981 году, на основе процессора 8086, Intel создала свой персональный компьютер IBM PC. Как раз начиная с этого аппарата можно написать программу, которая будет представлена ниже, на языке ассемблера и пользоваться ей.

# Основная часть

## Изучение и описание предметной области

Язык ассемблера появился 70 лет назад, в то время, когда ЭВМ стала позволять хранить в памяти, для того времени, большой объём информации. Тогда была разработана первая программа-транслятор, которая переводила в машинный код программы, написанные на понятном человеку языке. Эту программу назвали программой-сборщиком, а язык — языком ассемблера. Его можно считать языком второго поколения, если за первый принять машинный. Он работает непосредственно с процессором, и каждая его команда — это инструкция процессора, а не операционной или файловой системы. Перевод языка ассемблера в машинный код называется ассемблированием.

Команды ассемблера состоят из кодов операций и операндов. Операнды — это адреса, из которых процессор будет брать данные для вычислений и в которые будет помещать результат. Адресами могут быть ячейки оперативной памяти и регистры — память внутри процессора. Процессор работает с регистрами гораздо быстрее, чем с оперативной памятью.

Преимущества:

1. Позволяет максимально задействовать возможности процессора
2. Скорость работы программы
3. Размер программы
4. Доступность портов входа-выхода и особых регистров процессора
5. Возможность адаптировать под используемую платформу

Недостатки:

1. Большой код
2. Мало встроенных библиотек
3. Подходит для узконаправленных проектов

Применяют язык ассемблера для следующих задач:

1. Разработка операционных систем
2. Написание программ для микроконтроллеров и микропроцессоров
3. Разработка драйверов
4. Написание антивирусов
5. Разработка трансляторов для языков программирования

## Постановка задачи

Изучить на базовом уровне и уметь применять язык ассемблера. Познакомиться с программой Emu 8086. Выработать алгоритм разработки программы, которая должна вычислять значение произведения . Подпрограмма должна выполняться через вызов пользовательского прерывания (например, INT 60h). Адрес подпрограммы должен быть занесен в таблицу векторов прерываний при помощи функций DOS 25h и 35h. Подпрограмма должна выполнять действия, указанные в конкретном задании, при этом подпрограмме должны передаваться параметры N и j. Подпрограмма также должна возвращать результаты работы в регистрах общего назначения. После вызова подпрограммы программа должна восстановить адрес старого обработчика прерывания при помощи тех же функций DOS.

Параметры N и j могут передаваться в подпрограмму обработки прерывания через регистры общего назначения или через ячейки памяти. Значения параметров N и j не должны быть тривиальными (например, 1 или 0). Значение параметра N должно быть больше 1.

Написать программу с помощью эмулятора Emu 8086.

## Выбор структур данных для решения поставленной задачи

Структуры данных представлены в табл. 1.

Таблица 1

Структуры данных

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Количество итераций | N | Word |
| Переменная | i | Word |
| Переменная | j | Byte |

## Логическое проектирование

Алгоритм решения поставленной задачи:

1. Занести адрес подпрограммы в таблицу векторов прерываний при помощи функций DOS 25h и 35h
2. Сделать выполнение подпрограммы через вызов пользовательского прерывания INT 60h
3. Создать процедуру Func
4. Вывести на консоль через функцию DOS 40h текст с просьбой введения переменной j
5. Сделать ввод с консоли переменной j через функцию DOS 1h
6. Очистить консоль с помощью функции DOS 3h
7. Вывести на консоль через функцию DOS 40h текст с просьбой введения количества итераций N
8. Сделать ввод с консоли переменной N через функцию DOS 1h
9. Очистить консоль с помощью функции DOS 3h
10. Сохранить результат в переменной result
11. Вывести на консоль через функцию DOS 40h текст с 'Answer - '
12. Вернуть результат из переменной result в регистр общего назначения
13. Вывести посимвольно результат
14. Закончить процедуру Func
15. Инициализировать целочисленные и символьные переменные
16. Закончить выполнение программы

## Физическое проектирование программного обеспечения

Спецификация процедуры представлены в табл. 2.

Таблица 2

Спецификация модуля

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| Прерывание | Func | - | Занесение адреса подпрограммы в таблицу векторов прерываний и вызов пользовательского прерывания |
| Ввод | Func | - | Ввод значений |
| Вычисление | Func | N,j,i | Вычисление значения выражения |
| Вывод | Func | - | Вывод значений и символов |

## Кодирование

Программа рахзрабьотана на языке

Полный текст программы рприведён в приложении 2

CODE SEGMENT

ASSUME DS,CS:CODE

ORG 100H

Start:

mov ax, 3560h ;получение значения вектора прерывания.заносим в ah - 35h, заносим в al – 60h (номер вектора прерывания)

int 21h ;Вызов сервис DOS

mov si, bx

mov ax, 2560h ;установка значения вектора прерывания. заносим в ah - 35h,(заносим в al – 60h (номер вектора прерывания)

lea dx, Func ;Запись в dx адреса процедуры Func

int 21h ;Вызов сервис DOS

int 60h ;Вызов подпрограммы через прерывание

mov bx, si

mov bp, ax

mov ax, 2560h ;установка значения вектора прерывания. заносим в ah - 35h,(заносим в al – 60h (номер вектора прерывания)

mov dx, bx

mov di, ds

mov cx, es

mov ds, cx

int 21h ;Вызов сервис DOS

mov ds, di

mov ax, bp

RET

Func PROC ;Процедура обработки прерывания

mov ax,@data ;Вывод переменной msg1 на консоль

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg1

mov ah,40h

int 21h

mov ah,1h ; Ввод переменной j с консоли

int 21h

sub al,30h

mov j,al

cmp al,2 ; Проверка, чтобы j не была равна 2

je error

mov ax, 3h ;Очищение консоли

int 10h

mov ax,@data ;Вывод переменной msg2 на консоль

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg2

mov ah,40h

int 21h

mov ah,1h ;Ввод переменно N через консоль

int 21h

mov cx,ax

sub cx,30h

xor ch,ch

mov ax, 3h ;Очищение консоли

int 10h

m1: ;Вычисление выражения с помощью цикла loop

mov bl,j

mov ax,i

mov bh,2

sub bh,bl

idiv bh

imul result

mov result,ax

inc i

loop m1

mov result,ax ;Сохраняем результат вычислений в переменную result

mov ax,@data ;Вывод переменной msg3 на консоль

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg3

mov ah,40h

int 21h

mov ax,result ;Возвращение результата в регистр ax, для дальнейшего вывода на консоль

mov bl,10 ;Посимвольный вывод на консоль результата посредством деления на 10

div bl

mov cl,ah

cmp al,0

je m

cmp ah,4

je m2

div bl

mov bh,ah

m2:

mov dl,al

add dl,30h

mov ah,2h

int 21h

cmp cl,4

je m

mov dl,bh

add dl,30h

mov ah,2h

int 21h

m:

mov dl,cl

add dl,30h

mov ah,2h

int 21h

cmp ah,2

je end

error:

mov ax,@data ;Вывод переменной msg5 на консоль

mov ds,ax

mov bx,1

mov cx,31

mov dx, offset msg5

mov ah,40h

int 21h

cmp cx,31

je error1

end:

mov ax,@data ;Вывод переменной msg4 на консоль

mov ds,ax

mov bx,1

mov cx,32

mov dx, offset msg4

mov ah,40h

int 21h

error1:

IRET ;возврат из процедуры Func

Func ENDP

i dw 1 ;Инициализация целочисленных и символьных переменных

result dw 1

j db 1

msg1 db 'Enter j: '

msg2 db 'Enter N: '

msg3 db 'Answer - '

msg4 db 10,13,'Program completed successfully'

msg5 db 10,13,'Error - j not be 2.Try again!'

## Тестирование

Тестовые данные представлены в табл. 3.

Таблица 3

Тестовые данные

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| N=5; i=1; j=7 | Func | 0 |
| N=4; i=1; j=1 | Func | 24 |
| N=8; i=1; j=1 | Func | Переполнение регистра |
| N=6; i=1; j=1 | Func | 720 |

Результаты тестирования представлены в табл. 4.

Таблица 4

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирова-ния | Тести-руемый модуль или под-прог-рамма | Кто проводил тестирование | Опи-сание теста | Результаты тестирова-ния |
| 16.12.2022 20:00 | Func | Трофимов Н.В. | Запуск программы ввод j – 7,  Ввод N - 5 | 0 |
| 16.12.2022 20:01 | Func | Трофимов Н.В. | Запуск программы ввод j – 1,  Ввод N - 4 | 24 |
| 16.12.2022 20:02 | Func | Трофимов Н.В. | Запуск программы ввод j – 1,  Ввод N - 8 | Перепол-нение регистра |
| 16.12.2022 20:03 | Func | Трофимов Н.В. | Запуск программы ввод j – 1,  Ввод N - 6 | 720 |

# Заключение

В ходе изучения дисциплины «Программирование на ассемблере» изучил историю и строение микропроцессора Intel 8086. Освоил на базовом уровне язык ассемблера, выучив регистры, флаги, команды, прерывания.

Кульминацией обучения стала курсовая работа на тему «Программирование на языке низкого уровня», в которой описан весь процесс разработки собственной программы, высчитывающей выражение , при заданных с консоли значениях N и j и выводящую результат на экран.

# Список литературы

1. Ершов Е.В., д-р техн. наук, проф.; Виноградова Л.Н. и др. Методика и организация самостоятельной работы студентов − Коллектив авторов, ФГБОУ ВПО «Череповецкий государственный университет», 2012. − 208 с.
2. Assembler. Учебник для вузов. 2-е изд. / В. И. Юров — СПб.: Питер, 2003. — 637 с.
3. Язык Ассемблера для процессоров Intel : Кип Р. Ирвин ; [пер. с англ. С. Г. Тригуб]. - 4-е изд. - М. : Вильямс, 2005 (ГПП Печ. Двор). - 905 с.
4. Абель П. Язык Ассемблера для IBM PC и программирования. – М.: Высшая школа, 1992. – 447 с.

# Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Программирование на ассемблере

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ ,

д.т.н., профессор Ершов Е.В.

« » 2022 г.

Название Программирование на языке низкого уровня

Техническое задание на курсовую работу

Листов 7

Руководитель: Виноградова Л.Н.

Ф.И.О. преподавателя

Исполнитель:

студент 1Пиб-01-1оп-21

группа

Трофимов Н.В.

Фамилия, имя, отчество

1. год
2. Введение

Техническое задание для курсовой работы на тему «Программирование на языке низкого уровня» по дисциплине «Программирование на ассемблере».

1. Основание для разработки
   1. Задание на курсовую работу 1ПИб-01-1оп-21 – наименование документа на основании, которого ведется разработка.
   2. Череповецкий государственный университет – организация утвердившая данный документ.
   3. Дата утверждения – 25.10.2022
   4. Тема «Программирование на языке низкого уровня»
2. Назначение разработки

Раза должнв повышшать уровень компетеций

1. Требования к программе
   1. Требования к функциональным характеристикам

Программа должна вычислять значение произведения . Подпрограмма должна выполняться через вызов пользовательского прерывания (например, INT 60h). Адрес подпрограммы должен быть занесен в таблицу векторов прерываний при помощи функций DOS 25h и 35h. Подпрограмма должна выполнять действия, указанные в конкретном задании, при этом подпрограмме должны передаваться параметры N и j. Подпрограмма также должна возвращать результаты работы в регистрах общего назначения. После вызова подпрограммы программа должна восстановить адрес старого обработчика прерывания при помощи тех же функций DOS.

Параметры N и j могут передаваться в подпрограмму обработки прерывания через регистры общего назначения или через ячейки памяти. Значения параметров N и j не должны быть тривиальными (например, 1 или 0). Значение параметра N должно быть больше 1.

* 1. Требования к надежности

При вычислении произведений значения параметров N и j следует выбирать так, чтобы не происходило переполнения разрядной сетки для факториальных выражений. Для этого значением j должно быть число 1, так как из-за особенности выражения другие параметры j будут приводить к нулевому результату, в связи с тем, что результат первой итерации будет вещественное число между 0 и 1 (язык ассемблера не работает с вещественными числами, только с целыми), из-за этого ассемблер будет брать целую часть, то есть 0 и далее перемножать на него, а максимальным значением N – число 6.

* 1. Условия эксплуатации

1. температура окружающего воздуха от +10 до 35 °C
2. атмосферное давление от 630 до 800 мм ртутного столба
3. относительная влажность воздуха не более 80%
4. запыленность воздуха не более 0,75 мг/м3
5. в воздухе не должно быть паров агрессивных жидкостей и веществ, вызывающих коррозию
   1. Минимальные требования к составу и параметрам технических средств
6. Микропроцессор Intel 8086
7. Объем память – 1 Мб
8. Монитор
9. Клавиатура
10. Компьютерная мышь
    1. Требования к информационной и программной совместимости
11. Язык: Ассемблера
12. Среда программирования: Emu 8086
13. Операционная система: Window XP, Vista, 7, 8, 10
    1. Требование к маркировке и упаковке

Требования не предъявляются

* 1. Требования к транспортированию и хранению

Требования не предъявляются

* 1. Специальные требования

Требования не предъявляются

1. Требования к программной документации
   1. Содержание расчётно-пояснительной записки:

Титульный лист

Оглавление

Введение

1. Изучение предметной области
2. Постановка задачи
3. Выбор структуры данных
4. Логическое проектирование
5. Физическое проектирование
6. Кодирование
7. Тестирование

Заключение

Список литературы

Приложение

* 1. Требования к оформлению

Текстовые документы оформляются на белых листах формата А4. В соответствии с общими требованиями поля листа определяются следующим образом: левое – 30 мм, правое – 10 мм, верхнее – 20 мм, нижнее – 20 мм. Формат текста: Word for Windows, через полтора интервала (около тридцати строк на листе), шрифт – Times New Roman Cyr, размер шрифта – 14, отступ первой строки абзаца – 0,75 см. Количество знаков в строке, считая пробелы, – 60. Текст программы может быть расположен в две колонки, шрифт – Times New Roman Cyr, размер шрифта – 8.

Нумерация всех страниц (в том числе и приложений) сквозная. Номер проставляется в середине верхнего поля страницы арабской цифрой. Первая страница – титульный лист, вторая страница – аннотация, с третьей страницы начинается оглавление. Номера страниц на титульном листе, аннотации и оглавлении не проставляются.

Наименование разделов, подразделов, пунктов должно быть кратким и соответствовать содержанию. Каждая новая глава печатается с новой страницы. Это же правило относится и к другим основным структурным частям работы: аннотации, оглавлению, введению, заключению, списку литературы, приложениям.

Наименование разделов (основных частей) пишется прописными буквами по центру строки. Расстояние между заголовками и текстом, а также между заголовками разделов и подразделов должно быть равно двум интервалам.

Наименования подразделов и пунктов размещаются с абзацного отступа (0,75 см) и печатаются с прописной буквы, без подчеркивания и без точки в конце. Расстояние между последней строкой текста предыдущего раздела и последующим заголовком при расположении их на одной странице должно быть равно трем интервалам.

Разделы и подразделы нумеруются арабскими цифрами с точкой. Разделы имеют порядковые номера 1, 2 и т.д. Номер подраздела состоит из номера раздела и порядкового номера подраздела, входящего в данный раздел, разделенных точкой (например: 1.1, 2.5). При использовании ссылок на пункты, разделы и подразделы указывается порядковый номер раздела или пункта (например: «в разд. 2», «в п. 2.3.1»).

Перечисления нумеруются арабскими цифрами со скобкой с абзацного отступа. Для этого используются нумерованные списки. Допускается также применение маркированных списков одного вида.

1. Технико-экономические показатели

Требования не предъявляются

1. Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. 1.

Таблица 1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка о выполнении |
| 1 | 2 | 3 | 4 |
| Изучение предметной области | 01.11.2022 | Изучена предметная область |  |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Написание ТЗ | 15.11.2022 | Готовое ТЗ |  |
| Составление алгоритма | 20.11.2022 | Составлен алгоритм |  |
| Логическое проектирование | 25.11.2022 | Логически спроектировано |  |
| Физическое проектирование | 30.11..2022 | Физически спроектировано |  |
| Реализация кода | 05.12.2022 | Готовый код |  |
| Тестирование программы | 15.12.2022 | Пройденное тестирование |  |
| Написание РПЗ | 21.12.2022 | Готовое РПЗ |  |

1. Порядок контроля и приемки

Порядок контроля и приемки представлен в табл.2.

Таблица 2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Техническое задание | 25.10.2022 | Готовое ТЗ |  |
| Расчетно-пояснительная записка | 23.12.2022 | Готовое РПЗ |  |
| Защита курсовой работы | 27.12.2022 | Курсовая защищена |  |

Код программы

CODE SEGMENT

ASSUME DS,CS:CODE

ORG 100H

Start:

mov ax, 3560h

int 21h

mov si, bx

mov ax, 2560h

lea dx, Func

int 21h

int 60h

mov bx, si

mov bp, ax

mov ax, 2560h

mov dx, bx

mov di, ds

mov cx, es

mov ds, cx

int 21h

mov ds, di

mov ax, bp

RET

Func PROC

mov ax,@data

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg1

mov ah,40h

int 21h

mov ah,1h

int 21h

sub al,30h

mov j,al

cmp al,2

je error

mov ax, 3h

int 10h

mov ax,@data

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg2

mov ah,40h

int 21h

mov ah,1h

int 21h

mov cx,ax

sub cx,30h

xor ch,ch

mov ax, 3h

int 10h

m1:

mov bl,j

mov ax,i

mov bh,2

sub bh,bl

idiv bh

imul result

mov result,ax

inc i

loop m1

mov result,ax

mov ax,@data

mov ds,ax

mov bx,1

mov cx,9

mov dx, offset msg3

mov ah,40h

int 21h

mov ax,result

mov bl,10

div bl

mov cl,ah

cmp al,0

je m

cmp ah,4

je m2

div bl

mov bh,ah

m2:

mov dl,al

add dl,30h

mov ah,2h

int 21h

cmp cl,4

je m

mov dl,bh

add dl,30h

mov ah,2h

int 21h

m:

mov dl,cl

add dl,30h

mov ah,2h

int 21h

cmp ah,2

je end

error:

mov ax,@data

mov ds,ax

mov bx,1

mov cx,31

mov dx, offset msg5

mov ah,40h

int 21h

cmp cx,31

je error1

end:

mov ax,@data

mov ds,ax

mov bx,1

mov cx,32

mov dx, offset msg4

mov ah,40h

int 21h

error1:

IRET

Func ENDP

i dw 1

result dw 1

j db 1

msg1 db 'Enter j: '

msg2 db 'Enter N: '

msg3 db 'Answer - '

msg4 db 10,13,'Program completed successfully'

msg5 db 10,13,'Error - j not be 2.Try again!'

Приложение 2

# Руководство пользователя

## Общие сведения о программе

EMU8086 - бесплатный эмулятор для нескольких платформ. Он предоставляет пользователю возможность эмулировать старые процессоры 8086, которые использовались на компьютерах Macintosh и Windows с 1980-х и начала 1990-х годов. Он может эмулировать большое количество программного обеспечения, которое использовалось на этих микропроцессорах, но опытный пользователь может также программировать свой собственный код сборки для запуска на нем.

## Описание установки

1. Заходим по ссылке <https://emu8086.ru.malavida.com/windows/>
2. Нажимаем большую зеленую кнопку «Скачать для ПК»
3. Во вкладке загрузки на компьютере открываем файл emu8086.exe
4. Открывается меню загрузки, нажимаем кнопку «Next»
5. Открывается информация о загрузке, нажимаем кнопку «Next»
6. Выбираем место на компьютере, куда загрузится программа и нажимаем кнопку «Next»
7. Далее снова нажимаем кнопку «Next»
8. Переход на окно установки, нажимаем кнопку «Install»
9. Программа успешно установлена

## Описание запуска

1. Кликаем дважды по иконке Emu8086
2. Открывается программа, нажимаем кнопку «new»
3. Открывается меню выбора шаблона кода, выбираем «the emulator» и кликаем «OK»
4. Программа запущена, можно писать код