минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего 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-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Фатькина Алёна Дмитриевна |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Виноградова Людмила Николаевна |
| *фамилия, имя, отчество* |
| доцент |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_15\_\_»\_\_\_\_\_\_\_января\_\_\_\_\_2024 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024

*Год*

Аннотация

Данную курсовую работу по дисциплине «Программирование на ассемблере» на тему «Программирование на языке низкого уровня» выполнила студентка группы 1ПИб-02-3оп-22 Института информационных технологий Череповецкого государственного университета Фатькина Алёна Дмитриевна.

Целью курсовой работы является знакомство с принципами работы с языками низкого уровня, а именно с языком программирования Assembler для микропроцессора Intel 8086.

Курсовая работа содержит описание разработки программы, считающий сумму ряда (1):

Оглавление

[Введение 4](#_Toc154786891)

[1. Изучение и описание предметной области 6](#_Toc154786892)

[2. Постановка задачи 7](#_Toc154786893)

[3. Выбор структур данных 8](#_Toc154786894)

[4. Логическое проектирование 9](#_Toc154786895)

[5. Физическое проектирование 12](#_Toc154786896)

[6. Кодирование 13](#_Toc154786897)

[7. Тестирование 15](#_Toc154786898)

[Заключение 17](#_Toc154786899)

[Список литературы 18](#_Toc154786900)

[Приложение 1. Техническое задание 19](#_Toc154786901)

[Приложение 2. Руководство пользователя 27](#_Toc154786902)

[Приложение 3. Текст программы 32](#_Toc154786903)

# Введение

Программы имеют огромное значение в современном мире, поскольку они используются во всех сферах жизни - от бизнеса и науки до образования и развлечений.

Благодаря программам мы можем обрабатывать информацию, автоматизировать задачи, создавать и обрабатывать графику, управлять бизнес-процессами, разрабатывать новые технологии, а также выполнять множество других задач в различных сферах жизни.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [[3]](#_Источники).

Языки программирования разделяются на языки высокого уровня и языки низкого уровня. Уровень языка обозначает его удаленность от машинного кода целевой архитектуры процессора. Низкий уровень означает меньший масштаб преобразований, которые должен претерпеть код программы перед тем, как он может быть запущен. В данной курсовой работе будет рассматриваться язык программирования низкого уровня.

Одним из наиболее популярных языков программирования низкого уровня является Assembler (Ассемблер). Ассемблер представляет собой промежуточное звено между машинным кодом и языками программирования высокого уровня. Этот язык используется для написания программного обеспечения, которое управляет компьютером на более низком уровне, непосредственно взаимодействуя с аппаратным обеспечением [4].

Программы на языке ассемблера представляют собой набор инструкций, каждая из которых соответствует определенной команде процессора. Затем эти инструкции транслируются, или, по-другому, ассемблируются в машинный код – набор двоичных чисел, которые распознает центральный процессор и выполняет соответствующие введенным инструкциям операции.

При этом, для каждого процессора существует свой собственный язык ассемблера. В данной курсовой работе будет рассматриваться язык ассемблера для процессоров, созданных на базе архитектуры микропроцессора Intel 8086, выпущенного компанией Intel в 1978 году. В процессорах данной фирмы реализована преемственность, поэтому программы, написанные для младшей модели, могут быть без изменений запущены на более старшей.

Целью курсовой работы является знакомство с принципами работы с языками низкого уровня, а именно с языком программирования Assembler для микропроцессора Intel 8086.

В рамках курсовой работы необходимо написать программу на языке Ассемблер, которая высчитывает сумму ряда по формуле (1).

Для создания программы использовался эмулятор процессора 8086 - Emu8086.

# Изучение и описание предметной области

Язык низкого уровня – это язык программирования, который предоставляет более прямой и тесный доступ к аппаратным ресурсам компьютера по сравнению с высокоуровневыми языками. Программирование на языке низкого уровня чаще всего связано с близким взаимодействием с аппаратурой и машинным кодом. Два основных вида языков низкого уровня — ассемблер и машинный код:

Машинный код: это язык программирования, состоящий из двоичных кодов, которые напрямую исполняются процессором. Каждая команда в машинном коде соответствует определенной операции, выполняемой процессором.

Ассемблер: Этот язык представляет собой более читаемую форму машинного кода, где инструкции представлены мнемониками (например: MOV, ADD) и операндами. Ассемблер позволяет программистам использовать символические имена вместо бинарных кодов, что делает код более понятным.

Языки низкого уровня обычно ближе к аппаратуре и обладают высокой степенью контроля над ресурсами компьютера. Они могут использоваться для оптимизации производительности, написания драйверов устройств, создания операционных систем, и в других случаях, когда требуется тесное взаимодействие с аппаратурой. Однако, программирование на языках низкого уровня требует более глубокого понимания аппаратуры и может быть более трудным в сравнении с высокоуровневыми языками программирования.

# Постановка задачи

Для реализации программы необходимо написать код на языке ассемблера.

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 высчитывать сумму по формуле (1), 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.

# Выбор структур данных

Структуры данных представлены в табл. 1.

Таблица 1

Структуры данных

|  |  |  |
| --- | --- | --- |
| Описание | Обозначение | Тип данных |
| Количество итераций | N | Word |
| Переменная | result | Sword |
| Переменная | i | Word |
| Переменная | j | Sword |

# Логическое проектирование

Алгоритм работы программы:

1. Инициализация:

Установка сегментов данных и кода (ASSUME DS, CS:CODE).

Установка начального адреса программы (ORG 100H).

1. Ввод значения:

Загрузка значения 3560h в регистр AX.

Вызов прерывания 21h для ввода символа с клавиатуры.

1. Сохранение значения:

Сохранение значения из регистра BX в регистр SI.

Загрузка значения 2560h в регистр AX.

Загрузка адреса процедуры Func в регистр DX с использованием команды LEA.

Вызов прерывания 21h для выполнения процедуры Func.

Выход из программы с использованием прерывания 60h.

1. Возврат в основную программу:

Сохранение значения из регистра SI в регистр BX.

Сохранение значения из регистра AX в регистр BP.

Повторная загрузка значения 2560h в регистр AX.

Перемещение значений регистров DX (BX), DS (DI), и ES (CX).

Вызов прерывания 21h для завершения программы.

Перемещение значения N в регистр CX.

1. Определение процедуры Func:

Начало процедуры Func.

Начало цикла с меткой loop\_start.

Сравнение значения i с CX (N).

Переход к loop\_end, если i больше CX (N).

Перемещение значения i в регистр AX.

Умножение регистра AX на i, результат сохраняется в AX.

Умножение регистра AX на i, результат сохраняется в AX.

Умножение регистра AX на i, результат сохраняется в AX.

Умножение регистра AX на i, результат сохраняется в AX.

Умножение регистра AX на i, результат сохраняется в AX.

Вычитание из регистра AX значения j.

Прибавление значения AX к переменной result.

Инкрементация значения i.

Переход к loop\_start.

Метка loop\_end.

Сохранение результата в регистр AX.

Обнуление верхнего байта AX и вызов прерывания 21h для вывода результата.

1. Определение переменных:

Определение переменных i, j, N, и result с начальными значениями.

1. Завершение программы:

Завершение сегмента кода.

Окончание программы.

Блок-схема данной программы представлена на рис. 1.

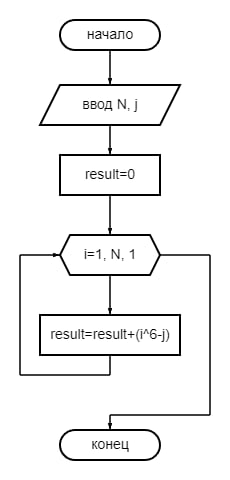


Рис. 1. Блок-схема программы

# Физическое проектирование

Программа состоит из одного файла KP.asm, в котором содержится код программы на языке ассемблера.

Команды, которые будем использовать в программе:

MOV – определяет адрес источника и записывает его в приемник;

LEA – определяет адрес источника во время выполнения программы и записывает его в приемник;

DW – определяет место под слово;

INT …h – вызов прерывания;

INC – увеличение на 1;

ADD – увеличение приемника на источник;

SUB – уменьшение приемника на источник;

MUL – умножение на источник;

CMP – сравнение 2 операндов со сменой флага нуля;

JG – если не ноль, то перейти к метке/адресу;

JMP – безусловный переход к метке/адресу;

AND – побитовая логическая операция;

RET – возврат из ближней процедуры.

# Кодирование

Для правильной работы программы добавим в начало:

CODE SEGMENT

ASSUME CS:CODE

ORG 100H

Start:

В начале программы описывается прерывание программы:

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

Цикл, который высчитывает значение суммы, пока значение i меньше значения N:

mov cx, N

loop\_start:

cmp i, cx

jg loop\_end

mov ax, i

mul i

mul i

mul i

mul i

mul i

sub ax,j

add result, ax

inc i

jmp loop\_start

loop\_end:

Заносим результат в регистр AX:

mov ax,result

В конец добавим объявление переменных в памяти и концовки

подпрограмм:

i dw 1

result dw 0

j dw …

N dw …

CODE ENDS

END Start

# Тестирование

Наборы тестовых данных представлены в табл. 2.

Таблица 2

Тестовые данные

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| 1 | N=2, j=2 | kp.asm | 61 |
| 2 | N=5, j=0 | kp.asm | 20515 |
| 3 | N=1, j=10 | kp.asm | -9 |
| 4 | N=3, j=100 | kp.asm | 494 |
| 5 | N=7, j=20000 | kp.asm | 44820 |
| 6 | N=4, j=20000 | kp.asm | -3110 |

Результаты тестов представлены в таблице 3.

Таблица 3

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемый модуль | Тестирование проводил | Описание теста | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №1. | Успех |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №2. | Успех |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №3. | Успех |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №4. | Успех |

Продолжение табл. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №5. | Успех |
| 26.12.2023 | kp.asm | Фатькина А.Д. | Тестирование с набором данных №6. | Успех |

# Заключение

В ходе выполнения курсовой работы была разработана программа, соответствующая требованиям технического задания и решающая математическое выражение (1), результат которого записывается в регистр AX.

В ходе разработки программы были освоены базовые возможности языка Assembler для процессоров на базе архитектуры микропроцессора Intel 8086, а также приобретены навыки работы с транслятором Turbo Assembler и компоновщиком Turbo Linker.

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ФГБОУ ВПО ЧГУ, 2015. – 243 с.
2. Виноградова, Л. Н. Системное программирование: Учеб. Пособие. – Череповец: ФГБОУ ВПО ЧГУ, 2016. – 210 с.
3. Язык программирования — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Язык_программирования>. Дата обращения: 20.12.2023.
4. Ассемблер — SkillFactory [электр.ресурс] https://blog.skillfactory.ru/glossary/assembler. Дата обращения: 20.12.2023.

# Приложение 1

МИНОБРАНАУКИ 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. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023 г.

Программирование на языке низкого уровня

Техническое задание на курсовую работу

Листов 8

Руководитель: Виноградова Л. Н.

Исполнитель: студент гр. 1ПИб-02-3оп-22

Фатькина А. Д.

2023 г.

Введение

Курсовая работа направлена на разработку программы на языке программирования низкого уровня.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Программирование на ассемблере», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 9 ноября 2023 года.

Наименование темы разработки: «Программирование на языке низкого уровня».

2. Назначение разработки

Основной задачей курсовой работы является создание программы, которая высчитывает сумму ряда по формуле (П1.1):

3. Требования к программе

* 1. Требования к функциональным характеристикам

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 вычислять сумму ряда по формуле (П1.1), 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.

* 1. Требования к надежности

Программа должна работать без ошибок. Для этого после создания она должна быть протестирована. При возникновении каких-либо ошибок они должны быть исправлены.

* 1. Условия эксплуатации

Для корректной работы программы необходимо:

* наличие любого устройства, на котором возможно запускать файлы формата .asm (компьютер, ноутбук);
* наличие на устройстве современной версии любой операционной системы.
  1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* оперативная память: не менее 1 Гб для 32-разрядной ОС, 2 Гб для 64-разрядной ОС;
* процессор на базе архитектуры микропроцессора 8086 не менее чем с 2 ядрами и тактовой частотой не ниже 1 ГГц;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие мыши и клавиатуры.
  1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* установленная операционная система (например, Windows XP, Vista, 7, 8, 8.1, 10, 11, macOS, Linux);
* установленный на устройстве транслятор «Turbo Assembler»;
* установленный на устройстве компоновщик «Turbo Linker»;
* установленный на устройстве эмулятор операционной системы MS-DOS «DOSBox».
  1. Требования к маркировке и упаковке

Требования к маркировке и упаковке программы не предъявляются.

* 1. Требования к транспортированию и хранению

Программа может храниться и транспортироваться в виде файлов, необходимых для ее работы, на носителе информации (флешка, CD-диск). Такую программу можно распространять, передавая его с одного компьютера на другой с помощью флешки или диска, либо отправляя ее по электронной почте.

* 1. Специальные требования

Специальные требования к программе не предъявляются.

4. Требования к программной документации

* 1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную

записку с содержанием:

1. Титульный лист
2. Аннотация
3. Оглавление
4. Введение
5. Основная часть
   1. Изучение и описание предметной области
   2. Постановка задачи
   3. Выбор структур данных
   4. Логическое проектирование
   5. Физическое проектирование
   6. Кодирование
   7. Тестирование
6. Заключение
7. Список литературы
8. Приложения
   1. Техническое задание
   2. Руководство пользователя
   3. Текст программы
   4. Требования к оформлению

Текстовые документы оформляются на белых листах формата А4, графический материал допускается представлять на листах формата A3. 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. 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 (например: 2), 3) и т. д.) с абзацного отступа. Для этого используются нумерованные списки. Допускается также применение маркированных списков одного вида. Таблицы в основном применяются для оформления цифрового материала. Шрифт – Times New Roman Cyr, размер шрифта – 14. Иногда возможен 10-й размер шрифта. Обозначения единиц физических величин необходимо применять в системе СИ. Номер таблицы размещается в правом верхнем углу над заголовком.

5. Технико-экономические показатели

Требования не предъявляются.

6. Стадии и этапы разработки

В данном пункте описаны стадии и этапы разработки программы (табл. П1.1).

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| 1 | 2 | 3 | 4 |
| Определение темы для курсовой работы | 09.11.2023 | Утверждена тема для разработки |  |
| Оформление технического задания | 17.11.2023 | Оформленное техническое задание |  |
| Изучение и описание предметной области | 21.11.2023 | Получены теоретические знания для разработки программы |  |
| Выбор структур данных | 28.11.2023 | Выбраны структуры данных |  |
| Логическое проектирование | 05.12.2023 | Подготовлен проект логической части программы |  |
| Физическое проектирование | 12.12.2023 | Подготовлен проект физической части программы |  |
| Написание программы | 18.12.2023 | Готовая программа |  |
| Тестирование и отладка | 24.12.2023 | Конечный вариант программы |  |
| Оформление сопроводительной документации | 27.12.2023 | Оформленная сопроводительная документация |  |

7. Порядок контроля и приемки

В данном пункте описан порядок контроля и приемки курсовой работы (табл. П1.2).

Таблица П1.2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| 1 | 2 | 3 | 4 |
| Сдача технического задания | 17.11.2023 | Согласованное техническое задания |  |
| Сдача расчетно-пояснительной записки | 27.12.2023 | Согласованная расчетно-пояснительная записка |  |
| Сдача курсовой работы | 27.12.2023 | Получение оценки за выполненную работу |  |

# Приложение 2

Руководство пользователя

1. Общие сведения о программе

Файл программы называется «KP.exe». При запуске программа высчитывает значение ряда суммы по формуле (П2.1):

1. Описание установки

Установка программы не требуется. Однако для ее запуска потребуется установить эмулятор процессора 8086 - Emu8086.

1. Описание запуска

Программу необходимо запускать в эмуляторе процессора 8086 - Emu8086.

процессора 8086 - Emu8086. Далее через эмулятор нужно будет открыть программу «КР.asm».

После открытия программы появляется окно с кодом и функциями (рис П2.1).

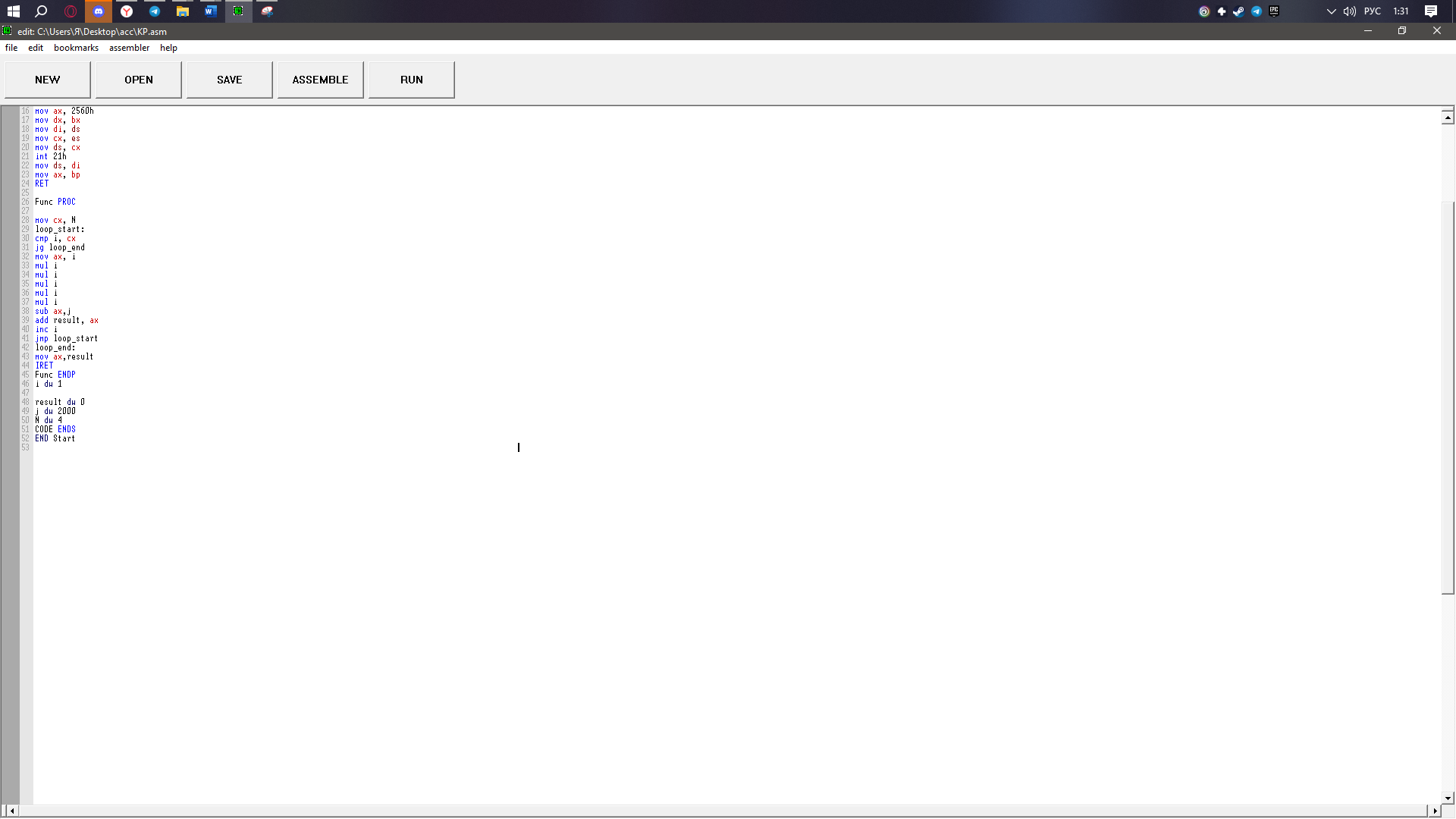


Рис П2.1. Вид программы emu8086

Для запуска эмулятора необходимо нажать кнопку «RUN» (рис. П2.2).

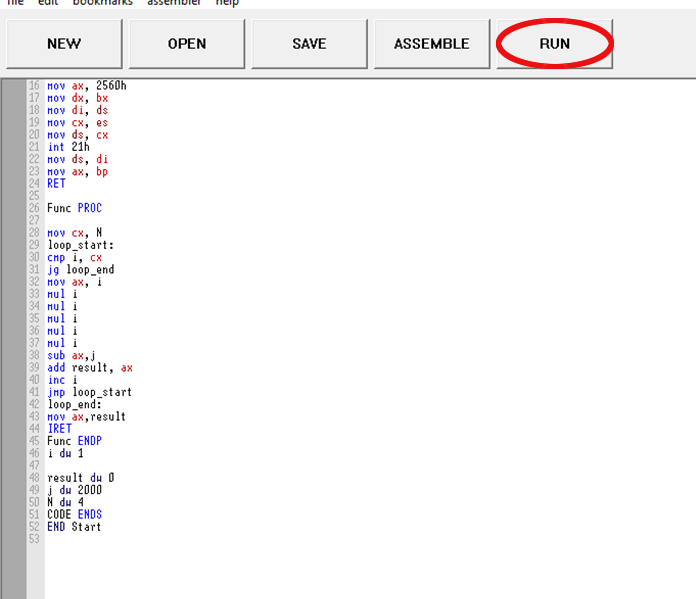


Рис П2.2. Кнопка «RUN»

В самом эмуляторе нажимаем на кнопку «RUN» (рис. П2.3), для начала работы программы.

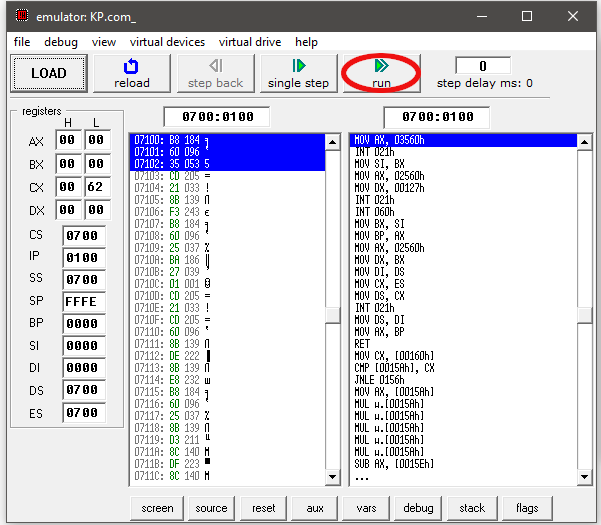


Рис П2.3. Кнопка «RUN» в эмуляторе

После завершения работы программа выведет сообщение об успешной обработке (рис. П2.4)

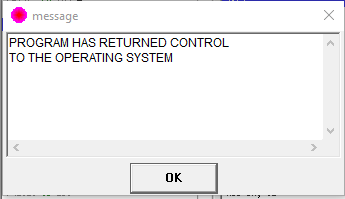


Рис П2.4. Сообщение об успешной обработке

Для того, чтобы узнать результат вычисления, нужно дважды нажать на регистр AX (рис П2.5).

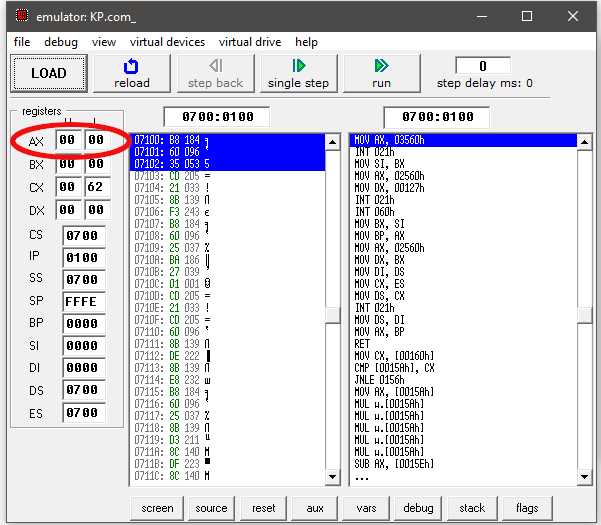


Рис П2.5. Регистр AX

После чего откроется окно (рис П2.6), в котором будет написан результат.

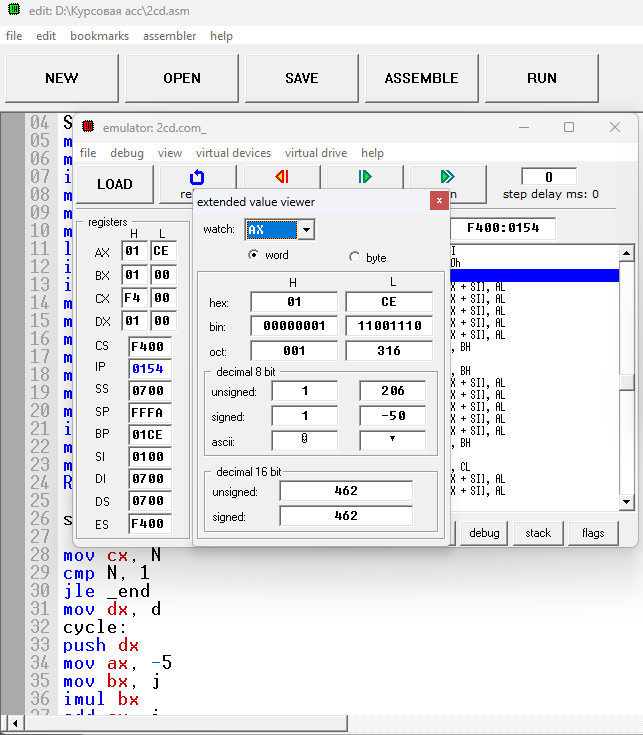


Рис. П.2.6. Итоговый результат

1. Инструкции по работе

Числа N и j при вводе должны соответствовать ограничениям. N является целочисленным беззнаковым типом WORD, j является целочисленным знаковым типом SWORD. Таким образом, допустимые значения для j [-32768; 2] ꓴ [0; 32767] и для N [2; 32767].

Лучше не вводить числа, при которых ответ может выйти за пределы диапазона значений [-32768; 32767]. В случае переполнения регистров программа может выдать неправильный результат.

# Приложение 3

Текст 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 cx, N

loop\_start:

cmp i, cx

jg loop\_end

mov ax, i

mul i

mul i

mul i

mul i

mul i

sub ax,j

add result, ax

inc i

jmp loop\_start

loop\_end:

mov ax,result

IRET

Func ENDP

i dw 1

result dw 0

j dw 2000

N dw 4

CODE ENDS

END Start