МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«**ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**»**

Институт (факультет)

Кафедра

Институт информационных технологий

Математическое и программное обеспечение ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине

на тему

Программирование на ассемблере

«Программирование на языке низкого уровня»

Выполнил студент группы

1ПИб-01-21оп

*группа*

Направления подготовки (специальности)

09.03.04 Программная инженерия

*шифр, наименование*

Акуличев Юрий Юрьевич

*фамилия, имя, отчество*

Руководитель

Виноградова Людмила Николаевна

*фамилия, имя, отчество*

Доцент

*должность*

Дата представления работы

«      » 2021 г.

Заключение о допуске к защите

Оценка                               ,

Количество баллов

Подпись преподавателя

Череповец, 2021

*год*

Аннотация

Курсовая работа по предмету: «Программирование на языке низкого уровня» выполнена студентом группы 1ПИб-01-21оп Института информационных технологий Акуличевым Юрием Юрьевичем.

При создании программы использовалось Техническое задание (приложение 1).

Курсовая работа содержит в себе описание созданного приложения.

Оглавление

[Введение 4](#_Toc89299473)

[1. Изучение и описание предметной области 6](#_Toc89299474)

[2. Постановка задачи 7](#_Toc89299475)

[3. Логическое проектирование 7](#_Toc89299476)

[4. Физическое проектирование программного обеспечения. 8](#_Toc89299477)

[5. Кодирование 8](#_Toc89299478)

[6. Тестирование 10](#_Toc89299479)

[Заключение 11](#_Toc89299480)

[Список литературы 12](#_Toc89299481)

[Приложение 1. Техническое задание. 13](#_Toc89299482)

[Приложение 2. Руководство пользователя 18](#_Toc89299483)

[Приложение 3. Текст программы. 29](#_Toc89299484)

Введение

Микропроцессоры компании Intel и персональные компьютеры на их основании прошли не очень долгий по времени, но значительный, по существу, путь развития, в течении которого всецело изменялись их возможности и даже сами принципы их архитектуры.

В то же время, привнося в микропроцессор принципиальные изменения, создатели были вынуждены непрерывно держать в виду потребность предоставления совместимости новых моделей со старыми, дабы не отпугивать потенциального покупателя возможностью абсолютной замены освоенного или разработанного им программного обеспечения. В результате нынешние микропроцессоры типа Pentium, обеспечивая такие возможности, как 32-битную адресацию практически безграничных размеров памяти, мультизадачный режим с одновременным выполнением нескольких программ, аппаратные средства защиты операционной системы и прикладных программ друг от друга, богатый набор дополнительных эффективных команд и способов адресации, в то же время могут работать (и часто работают) в режиме первых микропроцессоров типа 8086, используя всего лишь мегабайт оперативной памяти, 16-разрядные операнды (т.e числа в диапазоне до 2^6-1=65535) и ограниченный состав команд. Так как программирование на языке ассемблера напрямую касается аппаратных способностей микропроцессора, прежде всего, надлежит выяснить, в какой степени программист может использовать новые способности микропроцессоров в своих программах, и какие трудности программной несовместимости мо­гут при этом возникнуть.

Первые персональные компьютеры компании IBM, появившиеся в 1981 г. и получившие наименование IBM PC, использовали в качестве центрального вычислительного узла 16-разрядный микропроцессор с 8-разрядной внешней шиной Intel 8088. В дальнейшем в персональных компьютерах стал использоваться и другой вариант микропроцессора, 8086, который отличался от 8088 тем, что являлся полностью 16-разрядным. С тех пор его имя стало нарицательным, и в программах, использующих только возможности процессоров 8088 или 8086, говорят, что они работают в режиме 86-го процессора.

Целью выполнения данной курсовой работы является создание программы на языке ассемблера, которая будет выполнять подсчет произведения для определенных значений её элементов.

Основные этапы для решения поставленной задачи:

1. Проанализировать необходимую литературу, которая раскрывает возможности разработки программы на языке ассемблера.
2. Установить и использовать для разработки программы эмулятор микропроцессора 8086 “Emu8086”.
3. Выполнить логическое проектирование для создания программы.
4. Разработать программу, которая будет вычислять значение произведения , с помощью вызова обработчика прерывания.
5. Протестировать и убедиться в правильности работы программы с различными значениями элементов.

Данная программа может использоваться для различных математических расчетов, а также может быть внедрена в состав другой программы.

1. Изучение и описание предметной области

Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации. Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант и даёт возможность помечать строки программы метками с символическими именами с тем, чтобы к ним можно было обращаться (по именам, а не по адресам) из других частей программы (например, для передачи управления). Язык ассемблера обеспечивает использование символических меток вместо адресов ячеек памяти, которые при ассемблировании заменяются на автоматически рассчитываемые абсолютные или относительные адреса, а также так называемых директив.

Каждая модель (или семейство) процессоров имеет свой набор — систему команд и соответствующий ему язык ассемблера. Наиболее популярные синтаксисы языков ассемблера — Intel-синтаксис и AT&T-синтаксис.

Использование языка ассемблера предоставляет программисту ряд преимуществ перед программированием на языках высокого уровня. Большинство из них связано с близостью языка к аппаратной платформе.

Возможность непосредственного доступа к аппаратуре, и, в частности, портам ввода-вывода, конкретным адресам памяти, регистрам процессора.

Возможность максимально полного использования всех особенностей машины позволяет писать наиболее оптимизированный код для данного процессора.

Использование ассемблера практически не имеет альтернативы при создании: драйверов оборудования и ядра операционной программ, которые должны храниться в ПЗУ ограниченного объёма или выполняться на устройствах с ограниченной производительностью, платформо-зависимых компонентов компиляторов и интерпретаторов языков высокого уровня, системных библиотек и кода, реализующего совместимость платформ.

2. Постановка задачи

Целью данной курсовой работы является разработка алгоритма, реализующего подсчет значений в соответствии с заданием (суммы или произведения). Программа должна выполняться через пользовательские прерывания (например int60h). Адрес подпрограммы должен быть занесен в таблицу векторов прерываний при помощи DOS 25h и 35h. После чего подпрограмма должна возвращать значения в регистры общего назначения. При написании программы учитывать приоритеты операций и переполнение.

3. Логическое проектирование

Для решения поставленной задачи был разработан алгоритм.

Алгоритм:

1. Считываются параметры n, j, i.

2. Происходит получение вектора прерывания для INT 60h.

3. Сохраняется адрес прерывания.

4. Устанавливается вектор прерывания INT 60h на пользовательский обработчик.

5. Происходит вызов прерывания INT 60h.

6. В процедуре sum записывается значение n в CX, 0 в BX.

7. В цикле от 1 до n выполняется подсчет произведения  с помощью стандартных операций.

8. Возвращается адрес прерывания.

9. Вектор прерывания INT 60h устанавливается в изначальное состояние.

10. Выводится результат.

4. Физическое проектирование программного обеспечения.

Все переменные представлены в таблице 1, а функции в таблице 2.

Таблица 1

Переменные

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение переменной | Тип |
| Кол-во шагов цикла | n | Word |
| Параметр в выражении | j | Word |
| Счетчик | i | Word |

Таблица 2

Физическое проектирование

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| SUM | MAIN | - | Получение, хранение и установка вектора прерываний, вызов подпрограммы, вывод результата |
| SUM | SUM | n, j, i | Вычисление значения выражения |

5. Кодирование

Исходя из алгоритма, был написан код программы на языке программирования Assembler. Ниже представлен исходный код программы с пояснениями.

.MODEL SMALL

.CODE

ORG 100h

MAIN:

MOV AX, 3560h ; AH - 35H (дать вектор прерывания), AL - 60h (польз. прерывание)

INT 21h ; Вызываем сервис DOS, получаем ES:BX - значение вектора прерывания в 60h

MOV SI, BX

MOV AX, 2560h ; AH - 25H (уст. вектор прерывания), AL - 60h (польз. прерывание)

LEA DX, SUM ; Записываем в DX адрес процедуры обработки прерывания (подпрограммы)

INT 21h ; Вызываем сервис DOS, устанавливаем значение вектора прерывания 60h на нашу подпрограмму

INT 60h ; Вызываем подпрограмму через прерывание

MOV BX, SI

MOV BP, AX

MOV AX, 2560h ; AH - 25H (уст. вектор прерывания), AL - 60h (польз. прерывание - подпрограмма)

MOV DX, BX

MOV DI, DS

MOV CX, ES

MOV DS, CX

INT 21h ; Вызываем сервис DOS, устанавливаем значение вектора прерывания 60h равным DS:DX

MOV DS, DI

MOV AX, BP ; Возвращаем значение AX из BP (результат вычислений)

RET

SUM PROC ; Обработчик программного прерывания 60h

MOV CX, n

MOV BX, 1

CIKL:

MOV AX, 3

IMUL j

XOR AX,i

IMUL BX

MOV BX,AX

INC i

LOOP CIKL

IRET ; Возвращаемся из процедуры обработки прерывания

SUM ENDP

n DW 4

j DW 2

i DW 1

END MAIN Полный текст программы (приложение 3).

6. Тестирование

Тестовые данные (табл. 3.):

Таблица 3

Тестовые данные

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| N=4; i=1; j=2 | SUM | 01 18 |
| N=6; i=1; j=-2 | SUM | 08 C0 |
| N=15; i=1; j=7 | SUM | 1C 00 |

Результаты тестирования (табл. 3):

Таблица 3. Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемый модуль или подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.11.2021 10:00 | SUM | Аккуратный М.Д. | Использование отрицательного значения j = -4 | Ошибка (неверное имя регистра) |
| 15.11.2021 12:00 | SUM | Аккуратный М.Д. | Использование отрицательного значения j = -2 | Успех |
| 15.11.2021 14:00 | SUM | Акуличев Ю.Ю. | Использование положительного значения j = 7 | Успех |
| 15.11.2021 16:00 | SUM | Дыбаль А.К. | Использование положительного значения j = 3 | Успех |

Заключение

Языки низкого уровня в сегодняшних реалиях сохраняют свою актуальность и активно используются в разработке разнообразного ПО. Ассемблер является одним из них и остается востребованным, благодаря своим уникальным превосходствам в части производительности кода и прямой связи с аппаратной платформой. Данный язык применяется для создания драйверов оборудования и ядра операционной программ, платформо-зависимых компонентов компиляторов и интерпретаторов языков высокого уровня, системных библиотек и кода, реализующего совместимость платформ. Ассемблер зачастую используется, когда важно быстродействие и сохранение размера памяти, вследствие чего, его изучение остается востребованным для использования его функций в разработке ПО.

В ходе данной курсовой работы была изучена информация о построении программ, основных функциях и командах языка Ассемблер. Выполнено решение задачи на нахождение произведения с выполнением необходимых условий.

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. Пильщиков В.Н. Программирование на языке ассемблера IBM PC.
3. Фельдман С.К. Системное программирование на персональном компьютере – М:Бук-пресс, 2006.
4. Ирвин К. Язык ассемблера для процессоров Intel, 4-е издание.: пер. с англ. – М.: Издательский дом «Вильямс», 2005.
5. Абель П. Язык Ассемблера для IBM PC и программирования. – М.: Высшая школа, 1992. – 447 с.
6. Брэдли Д. Программирование на языке ассемблера для персональной ЭВМ фирмы IBM.– М.: Радио и связь, 1988. – 448 с.
7. Л.Н. Виноградова Учебное пособие “Системное программирование” 2016.
8. Нортон П., Соухэ Д. Язык ассемблера для IBM PC.– М.: Компьютер, 1992.
9. Голубь Н.Г. Искусство программирования на Ассемблере. Лекции и упражнения – М:DiaSoft, 2002.
10. В. И. Юров “Assembler” 2006.

ПРИЛОЖЕНИЕ 1

Техническое задание

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

Математическое и программное обеспечение ЭВМ

Программирование на ассемблере

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2020 г.

Программирование на языке низкого уровня

Техническое задание на курсовую работу

Листов 4

Руководитель: доцент Виноградова Л.Н.

Исполнитель: студент гр. 1ПИб-01-21оп

Акуличев Ю.Ю.

2021 г.

Введение

Тема курсовой работы: "Программирование на языке низкого уровня".

Цель курсовой работы: написание программы для вычисления значения выражения: 

Основная задача программы вычисление значения выражения.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу, выданное в первом семестре 2021/2022 учебного года на кафедре Математического и программного обеспечения ЭВМ по дисциплине: "Программирование на ассемблере".

2. Назначение разработки

Программа предназначена для студентов, изучающих информатику и программирование. Программа может быть использована для промежуточных математических вычислений. Основная задача данной курсовой работы - закрепление и применение на практике материала, полученного в ходе изучения дисциплины: "Программирование на ассемблере".

3. Требования к программе

3.1 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 производить расчёт по формуле: 

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.

При вычислении значений сумм с использованием логических операций под знаком суммы или произведения следует учитывать приоритеты логических и арифметических операций.

В процессе проверки работы программы в отладчике для выполнения в пошаговом режиме подпрограммы обработки прерывания необходимо команду INT выполнить в режиме пошагового выполнения команды (нажать комбинацию клавиш Alt+F7).

3.2 Требования к надежности

Параметры N и j могут передаваться в подпрограмму обработки прерывания через регистры общего назначения или через ячейки памяти. Их значения должны быть тривиальными и такими, чтобы результат был не больше 65535.

Значение параметра j должно быть в диапазоне [-5;5].

Значение параметра N должно быть в диапазоне [2;4].

3.3 Условия эксплуатации

Персональный компьютер должен эксплуатироваться в закрытом помещении, где имеется регулярное отопление. Температура воздуха должна составлять 20 +/-10 градусов, относительная влажность 60 +/- 15 %, атмосферное давление 84-107 КПа.

Источником питания компьютера является однофазная сеть переменного тока частотой 50-60 Гц и напряжением 187-242 В. Сопротивление заземляющего контура в соответствии с Правилами налаживания электроустановок не должно быть более 4 Ом.

3.4 Требования к составу и параметрам технических средств

1. 256 MB RAM и больше;
2. Дисплей 1280 × 768;
3. Мышь, трекпад или ручка + планшет.
   1. Требования к информационной и программной совместимости

Операционные системы Windows XP и новее. Текст программы реализуется на языке низкого уровня с использованием эмулятора микропроцессора Emu8086.

* 1. Требования к маркировке и упаковке

Требования не предъявляются.

* 1. Требования к транспортированию и хранению

Требования не предъявляются.

3.8. Специальные требования

Требования не предъявляются.

4. Требования к программной документации

Состав программной документации должен включать в себя: расчетно-пояснительную записку, текст программы, техническое задание, руководство пользователя.

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. П1.1

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Составление и доработка ТЗ | 10.11.2021-17.11.2021 | ТЗ, согласованное с преподавателем | Выполнено |
| Написание программы | 17.11.2021-01.12.2021 | Программа, проверенная на нескольких тестовых значениях | Выполнено |
| Тестирование программы | 02.12.2021-03.12.2021 | Программа протестирована и полностью готова к работе | Выполнено |
| Составление расчетно-пояснительной записки | 03.12.2021-14.12.2021 | Расчетно-пояснительная записка, проверенная преподавателем | Выполнено |
| Составление руководства пользователя | 15.12.2021 | Руководство пользователя | Выполнено |
| Доработка курсовой работы и защита | 16.12.2021-22.12.2021 | Курсовая работа, готовая для защиты | Выполнено |

# 6. Порядок контроля и приемки

Порядок контроля и приёма представлены в таблице П1.2.

Таблица П1.2

Порядок контроля и приёма

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приёмке результата контрольного этапа |
| 1 | 2 | 3 | 4 |
| Технические задание | 18.10.2021 | Оформленное техническое задание | Принято |

Продолжение табл. П1.2.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Теоретическая часть курсовой работы | 01.11.2021 | Оформленная теоретическая часть | Принято |
| Практическая часть курсовой работы | 19.11.2021 | Программа | Принято |
| Расчетно-пояснительная записка | 15.12.2021 | Оформленная РПЗ | Принято |
| Защита курсовой работы | 17.12.2021 | Получение итоговой оценки за курсовую работу | Принято |

ПРИЛОЖЕНИЕ 2

Руководство пользователя

1. Общие сведения

Emu8086 - исчерпывающий пакет программ, необходимых рядовому пользователю для изучения языка программирования Assembler, а также написания и отладки любых программ. Представленный пакет включает в себя редактор исходного кода, ассемблер и эмулятор микропроцессора (виртуальную машину) с отладчиком. Помимо того, в EMU8086 входят обучающие программы, что делает его особенно полезным для неопытных программистов. Предусмотрено в программе и пошаговое выполнение приложений на эмуляторе, наблюдение за содержимым регистров, значениями флагов и состоянием памяти. EMU8086 работает с любыми версиями операционной системы Windows.

1. Описание установки
   1. Возьмите диск, приложенный к данной курсовой работе, на котором находится установочный файл (рис.1.1).



Рис.1.1 Взятие диска

* 1. Вставьте взятый вами диск в дисковод (рис.1.2).



Рис.1.2 Вставка диска в дисковод

* 1. Нажмите на кнопочку закрытия дисковода и дождитесь его загрузки(рис.1.3). В открывшемся окне выберите вкладку “открыть папку для просмотра файлов” (рис.1.4).



Рис.1.3 Закрытие дисковода

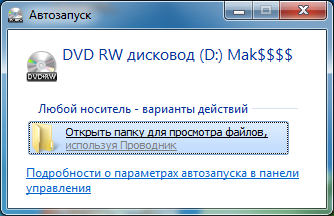


Рис.1.4 Открытие папки с файлами

* 1. В открывшейся папке с помощью двойного нажатия на левую кнопку мыши запустите установочный файл «setup.exe» (рис.1.5).



Рис.1.5 Установочный файл

* 1. В открывшемся окне нажмите кнопку «Next»(рис.1.6)

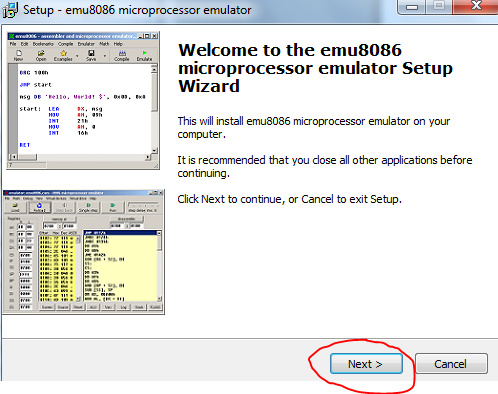


Рис.1.6 Начало установки

* 1. Повторите действие выше и снова нажмите кнопку «Next» (рис.1.7).

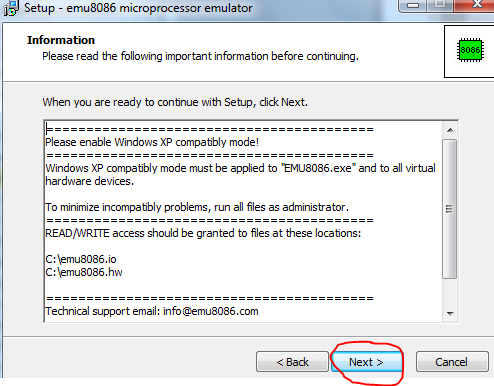


Рис.1.7 Продолжение установки

* 1. Выберите папку, в которой будет хранится программа для установки, с помощью кнопки «Browse» (1) и нажмите «Next» (2) (рис.1.8).

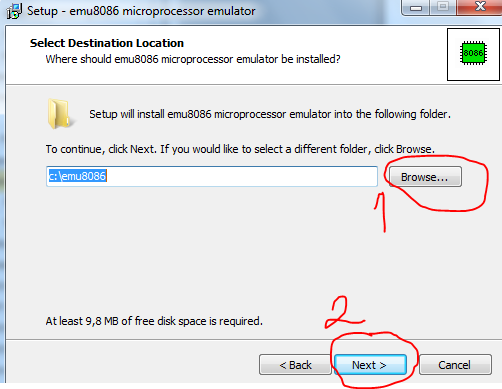


Рис.1.8 Выбор директории

* 1. Снова нажмите «Next» (рис.1.9).

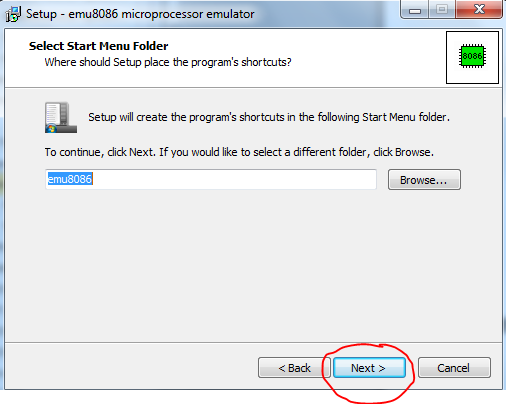


Рис.1.9 Выбор названия по умолчанию

* 1. Нажмите кнопку «Install» (рис.1.10).

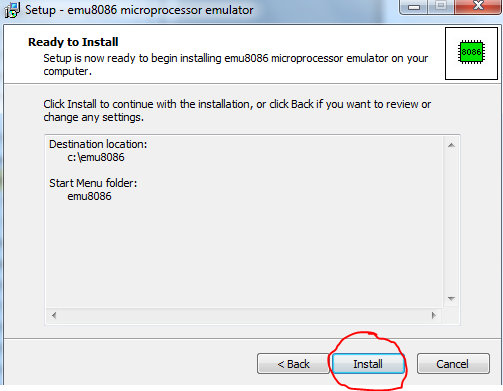


Рис.1.10 Нажатие кнопки «Install»

* 1. Дождитесь окончания установки и нажмите кнопку «Finish» (рис.1.11).

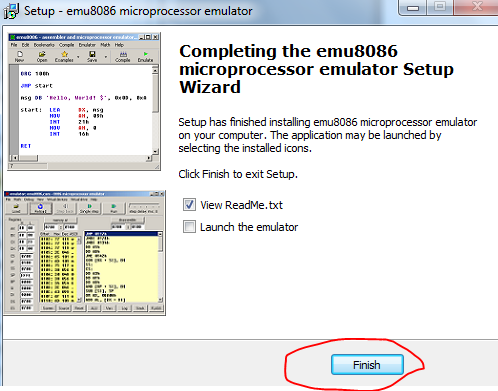


Рис.1.11 Конец установки

1. Описание запуска
2. Запуск Emu8086 осуществляется через ярлык на рабочем столе путём двойного нажатия по ярлыку с помощью левой кнопки мыши (рис.2.1).



Рис.2.1 Запуск программы

1. Программа предложит выбрать один из начальных вариантов. Следует нажать “New” (рис.2.2).

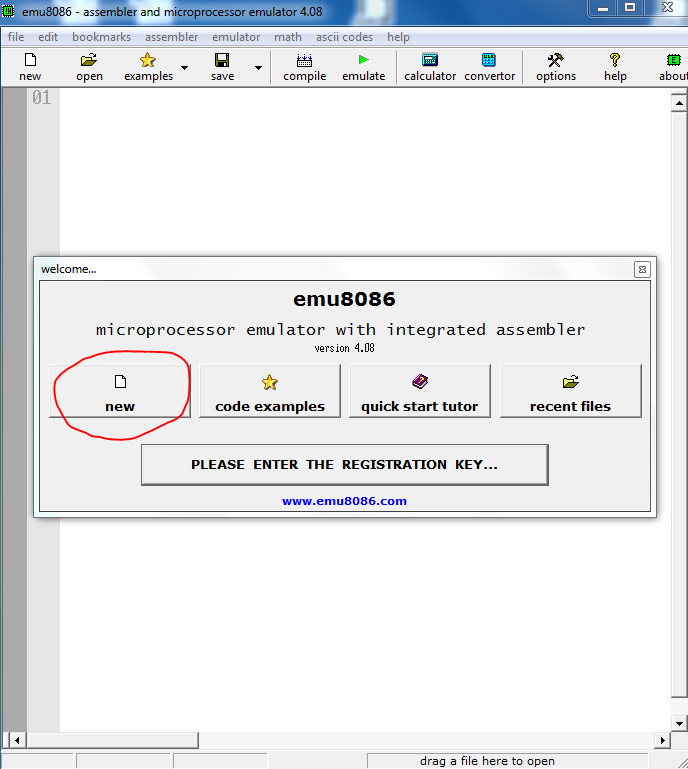


Рис.2.2 Создание нового файла

1. Открытие программы осуществляется через кнопку «open» (рис.2.3), где следует выбрать диск, на котором находится файл BestCourseworkByYura.asm, после чего выбрать его (рис.2.4).

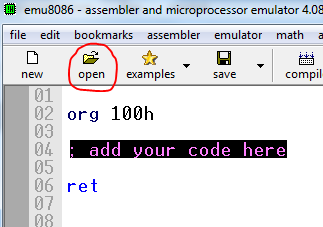


Рис.2.3 Открытие файла



Рис.2.4 Открытие файла “BestCourseworkByYura.asm”

1. Через пункт emulate запустить программу(рис.2.5).



Рис.2.5 Запуск программы

1. Описание пользовательского интерфейса

Программа не предусматривает пользовательский интерфейс поэтому для изменения данных программы необходимо изменение исходного кода (BestCourseworkByYura.asm), а именно параметров N, j и i в программе Emu8086. Как это сделать, описано ниже.

1. Спуститесь в самый низ программы и найдите там обозначение переменных N, j и i (рис.3.1).

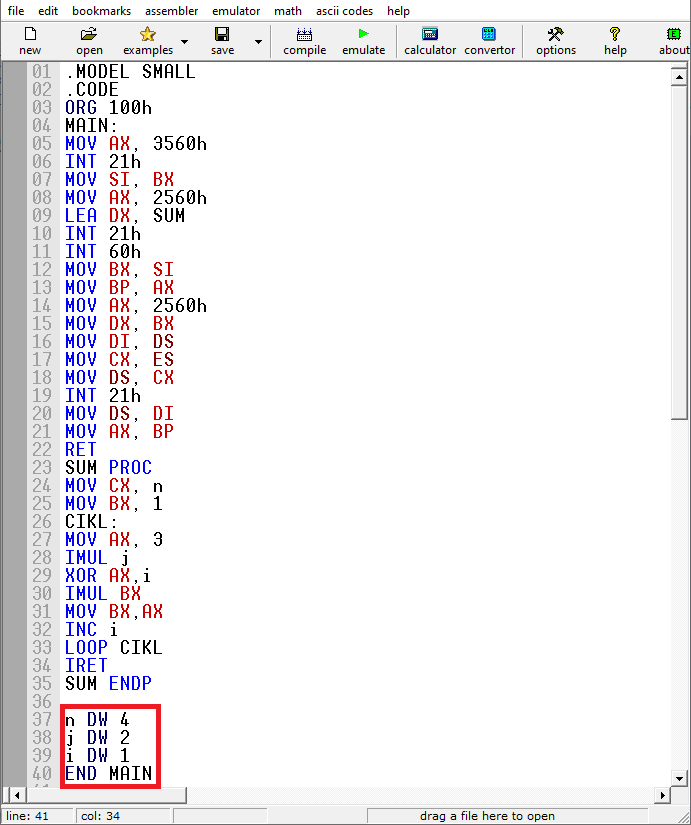


Рис.3.1 Переменные

1. Для того, чтобы изменить значения на те, которые вам необходимы, поменяйте цифру напротив каждой переменной на вашу. Например, как показано на рисунке ниже мы заменили n=4 на n=6. (рис.3.2-3.3)

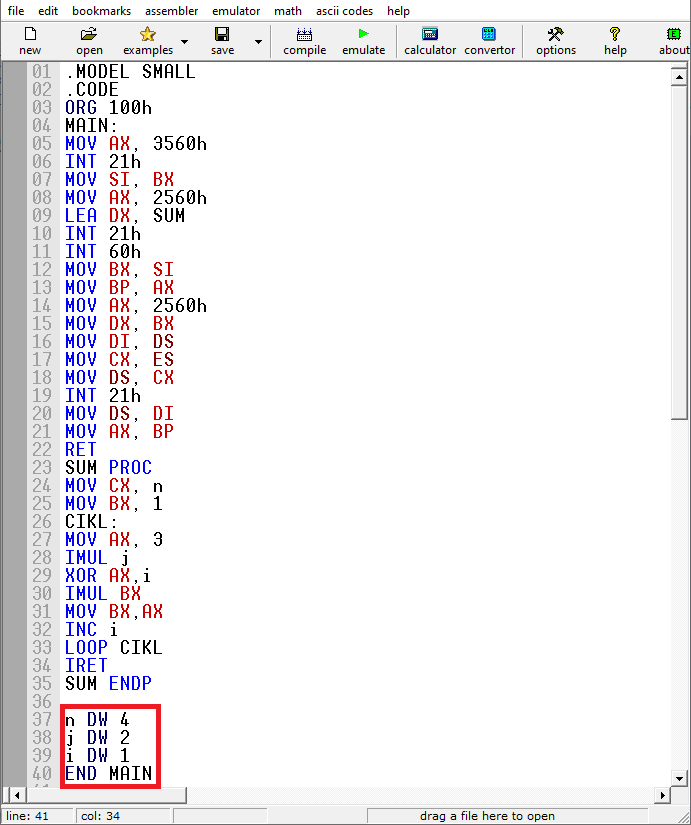


Рис.3.2 Замена переменных

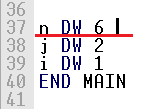


Рис.3.3 Замена переменной n

1. Переменные заменяйте в промежутках: N [2;4], J [-5;5] в противном случае, программа может выдать неверный результат или не запуститься.

ПРИЛОЖЕНИЕ 3

Текст программы

ORG 100h

MAIN:

MOV AX, 3560h

INT 21h

MOV SI, BX

MOV AX, 2560h

LEA DX, SUM

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

SUM PROC

MOV CX, n

MOV BX, 1

CIKL:

MOV AX, 3

IMUL j

XOR AX,i

IMUL BX

MOV BX,AX

INC i

LOOP CIKL

IRET

SUM ENDP

n DW 4

j DW 2

i DW 1

END MAIN

mov cx,n

mov bx,j

m1:

i dw 1

j dw 1

n dw 3

result dw 1

CODE ENDS

END START

mov ax,i

mov sp,2

sub sp,bx ;2-j

idiv sp ;i/(2-j)

imul result

mov result,ax

inc i

loop m1