МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Институт (факультет) | Институт информационных технологий | | | |
| Кафедра | Математическое и программное обеспечение ЭВМ | | | |
| КУРСОВАЯ РАБОТА | | | | |
| По дисциплине | | Программирование на ассемблере | | |
| на тему | | «Программирование на языке низкого уровня» | | |
|  | | | Выполнил студент группы | |
|  | | | 1ПИб-01-1оп-21 | |
|  | | | *группа* | |
|  | | | Направление подготовки (специальности) | |
|  | | | 09.03.04 Программная инженерия | |
|  | | | *шифр, наименование* | |
|  | | | Пикалова Анастасия Сергеевна | |
|  | | | *фамилия, имя, отчество* | |
|  | | | Руководитель | |
|  | | | Виноградова Людмила Николаевна | |
|  | | | *фамилия, имя, отчество* | |
|  | | | Доцент | |
|  | | | *должность* | |
|  | | | Дата представления работы | |
|  | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |
|  | | | Заключение о допуске к защите | |
|  | | |  | |
|  | |
|  | |
|  | | | Оценка \_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_ | |
|  | | |  | Количество баллов |
|  | | | Подпись преподавателя \_\_\_\_\_\_\_\_ | |
|  | | |  | |
|  | |  | | |
|  | | Череповец, 2022 | | |
| *год* | | | | |

Аннотация

Курсовая работа по предмету: «Программирование на языке низкого уровня» выполнена студенткой группы 1ПИб-01-1оп-21 Института информационных технологий Пикаловой Анастасии Сергеевной.

При создании программы использовалось Техническое задание (приложение 1). Курсовая работа содержит в себе описание созданного приложения.

Оглавление

[Введение 4](#_Toc121598419)

[1. Изучение и описание предметной области 6](#_Toc121598420)

[2. Постановка задачи 8](#_Toc121598421)

[3. Логическое проектирование 9](#_Toc121598422)

[4. Физическое проектирование 9](#_Toc121598423)

[5. Кодирование 10](#_Toc121598424)

[6. Тестирование 12](#_Toc121598425)

[Заключение 13](#_Toc121598426)

[Список литературы 14](#_Toc121598427)

[Приложение 1. Техническое задание 15](#_Toc121598428)

[Приложение 2. Текст программы 21](#_Toc121598429)

[Приложение 3. Руководство пользователя 26](#_Toc121598452)

Введение

Микропроцессоры корпорации Intel и персональные компьютеры на их основе прошли не очень долгий, но на самом деле значительный путь развития, в ходе которого их возможности и даже принципы их архитектуры кардинально изменились.

В то же время при кардинальных изменениях в микропроцессоре разработчики были вынуждены постоянно помнить о необходимости обеспечения совместимости новых моделей со старыми, чтобы не пугать потенциального покупателя перспективой полной замены освоенного или разработанного им программного обеспечения. В результате современные микропроцессоры Pentium, которые обеспечивают такие возможности, как 32-разрядная адресация практически неограниченного объема памяти, режим многозадачности с одновременным выполнением нескольких программ, аппаратная защита операционной системы и прикладных программ друг от друга, богатый набор других эффективных команд и методов адресации, могут одновременно работают (и часто срабатывают) в режиме первых микропроцессоров 8086, используя только мегабайт оперативной памяти, 16-битные операнды (т.е. j. числа в диапазоне до 2^6-1=65535) и ограниченный состав команды. Поскольку программирование на языке ассемблера напрямую влияет на аппаратные возможности микропроцессора, в первую очередь необходимо выяснить, в какой степени программист может использовать новые возможности микропроцессоров в своих программах, и какие проблемы с несовместимостью программного обеспечения могут возникнуть в этом случае.

Первые персональные компьютеры корпорации IBM, появившиеся в 1981 году и получившие название IBM PC, использовали в качестве центрального вычислительного узла 16-разрядный микропроцессор с 8-разрядной внешней шиной Intel 8088. Позже в персональных компьютерах использовалась другая версия микропроцессора - 8086, которая отличалась от 8088 тем, что была полностью 16-разрядной. С тех пор его название стало нарицательным, и в программах, которые используют только возможности процессоров 8088 или 8086, говорят, что они работают в режиме 86. processor.

Целью данной курсовой работы является приобретение практических навыков программирования на языке ассемблера.

Результатом курсовой работы является разработка программы, которая вычисляет значение суммы.

1. Изучение и описание предметной области

Процессор обрабатывает информацию, выполняя определенные команды. Таких команд может быть более тысячи. У каждой команды есть свой код. Коды всех команд процессора записаны в двоичной форме в специальном документе, который называется системой команд процессора. У каждого процессора своя система команд, поэтому один и тот же код для, разных процессоров может обозначать разные команды. Если же процессоры имеют ограниченную совместимость, то их рассматривают как семейство. Основной командой передачи данных является команда MOV, которая выполняет операцию присваивания: приемник MOV, источник команда MOV присваивает значение операнда получателя значению операнда источника. Универсальный регистр, сегментный регистр или ячейка памяти могут выступать в качестве приемника, константа, универсальный регистр, сегментный регистр или ячейка памяти могут выступать в качестве источника. Оба операнда должны быть одинакового размера.

Микропроцессор может передавать данные на порты ввода-вывода, поддерживаемые аппаратным обеспечением, и использовать линии ввода-вывода процессора, соответствующие их назначению. Адресное пространство аппаратного ввода-вывода процессора не является адресным пространством физической памяти. Адресное пространство ввода-вывода состоит из 64Кбайт индивидуально адресуемых 8-битных портов ввода-вывода, имеющих адреса 0…FFFFh. Адреса 0F8h…0FFh являются резервными. Любые два последовательных 8-битных порта могут быть объединены в 16-битный порт, 4 последовательных 8-битных порта – в 32-битный порт.

Команды преобразования типов предназначены для правильного изменения размера операнда, неявно установленного в регистре накопителя (EAX, AX, AL). Операнд не указывается сразу после сочетания клавиш.

Команды прерывания предназначены для управления программными прерываниями. Как правило, прерывание — это асинхронное выключение процессора, вызванное запуском устройства ввода-вывода. Исключением являются синхронные прерывания, которые возникают, когда во время выполнения команды определяются определенные предопределенные условия. При получении сигнала прерывания процессор останавливает выполнение текущей программы и переключается на выполнение обработчика прерываний, предварительно записанного для каждого прерывания. Архитектура IA-32 поддерживает 17 аппаратных векторов прерываний и 224 определяемых пользователем. Команда INT вызывает обработчик прерывания, указанный операндом (константой). Операнд определяет номер вектора прерывания BIOS от 0 до 255, представленный в виде 8-разрядного целого числа без знака. Когда вызывается обработчик прерываний, регистры EIP, CS и EFLAGS сохраняются в стеке. Нарушение переполнения вызывается отдельной командой INTO и имеет вектор 04h.

Команды организации цикла - по умолчанию используйте регистр ECX в качестве счетчика количества повторений цикла. Каждый раз, когда вы запускаете команду LOOPcc, значение реестра ECX уменьшается на 1, а затем сравнивается с 0. Если ECX=0, выполнение цикла заканчивается, и программный код, написанный после команды LOOPcc, продолжает выполнение. Если ECX содержит ненулевое значение, выполняется переход к адресу операнда команды LOOPcc.

2. Постановка задачи

Целью данной курсовой работы является разработка программы, вычисляющей значение суммы (рис. 1).

Рис. 1. Формула

3. Логическое проектирование

После запуска программы происходит ввод переменной N и переменной j, затем происходит занесение адреса подпрограммы в таблицу векторов прерываний при помощи функций DOS 25h и 35h, затем подпрограмма вызывается пользовательским прерыванием INT 60h и идёт расчёт суммы ряда: в цикле пока N не равно нулю осуществляется увеличение i на единицу, умножение i на семь и деление этого произведения на сумму j и 1, потом результат суммируется с другими результатами ряда и N уменьшается на единицу. Значение суммы всех результатов ряда заносится в регистр ВX. Восстанавливается старый вектор прерывания для пользовательского прерывания INT 60h и сумма выводится на экран.

4. Физическое проектирование

Программа состоит из файла Summa7.asm, в котором содержится код программы на языке ассемблера. Функции и их параметры представлены в таблице 1.

Таблица 1

Физическое проектирование

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Функция | Параметры | Выполняемое действие |
| Summa7.asm | 10H | MOV AX, 3H | Очистка экранного буфера |
| Summa7.asm | 21H/9H | MOV AH,9H  MOV DX,OFFSET X | Выводит строку из регистра DX |
| Summa7.asm | 21H/1H | MOV AH,01H | Ввод с клавиатуры в регистр AL |
| Summa7.asm | 21H/35H | MOV AH, 35H  MOV AL, 60H  MOV X, ES  MOV Y, BX | Получение старого вектора прерывания и сохранение его сегмента в X, а смещения в Y |
| Summa7.asm | 21H/25H | MOV DX, OFFSET X  MOV AX, SEG X  MOV DS, AX  MOV AH, 25H  MOV AL, 60H | Установка нового вектора прерывания для подпрограммы X |
| Summa7.asm | 21H/2H | MOV AH, 02H  MOV DX, X | Выводит символ X на экран |

5. Кодирование

Ввод с клавиатуры производится с помощью следующей последовательности инструкций:

vvodN:

push dx

push bx

push ax

mov cl,10

mov ah,9h

mov dx,offset enterN

int 21h

mov ah,01h

int 21h

cmp al,39h

ja vvodn

cmp al,30h

jb vvodn

sub al,30h

mov bl,al

moreN:

mov ah,01h

int 21h

cmp al,20h

je vvodNend

cmp al,39h

ja vvodN

cmp al,30h

jb vvodN

sub al,30h

xchg al,bl

imul cl

add bl,al

jmp moreN

vvodNend:

mov cx,bx

cmp cx,1

jle vvodN

vvodj:

mov ah,9h

mov dx,offset enterj

int 21h

mov dl,10

mov ah,01h

int 21h

cmp al,2dh

je vvodoj

cmp al,39h

ja vvodj

cmp al,30h

jb vvodj

sub al,30h

mov bl,al

morej:

mov ah,01h

int 21h

cmp al,20h

je vvodjend

cmp al,39h

ja vvodj

cmp al,30h

jb vvodj

sub al,30h

xchg al,bl

imul dl

add bl,al

jmp morej

vvodjend:

mov j,bx

jmp vvodexit

vvodoj:

mov ah,01h

int 21h

cmp al,39h

ja vvodj

cmp al,30h

jb vvodj

sub al,30h

mov bl,al

moreoj:

mov ah,01h

int 21h

cmp al,20h

je vvodojend

cmp al,39h

ja vvodj

cmp al,30h

jb vvodj

sub al,30h

xchg al,bl

imul dl

add bl,al

jmp moreoj

vvodojend:

neg bx

mov j,bx

vvodexit:

pop ax

pop bx

pop dx

Установка нового вектора прерывания производится с помощью следующей последовательности инструкций:

mov ah,35h

mov al,60h

int 21h

mov savecs,es

mov saveip,bx

push ds

mov dx,offset summ

mov ax,seg summ

mov ds,ax

mov ah,25h

mov al,60h

int 21h

pop ds

Подпрограмма для вычисления значения заданного выражения:

summa proc far

mov CX,0

mov Cl,N

L:mov ax,i

imul ml

mov BX, j

inc BX

CWD

IDIV BX

add i,1

add res, ax

loop L

exit:

iret

summa endp

Восстановление старого прерывания производится с помощью следующей последовательности инструкций:

push ds

mov dx,savecs

mov ax,saveip

mov ds,ax

mov ah,25h

mov al,60h

int 21h

pop ds

Вывод результата производится с помощью следующей последовательности инструкции:

mov ah,9h

mov dx,offset answer

int 21h

push bx

cmp bx,0

je nol

jl otr

jg pol

nol:

mov ah, 02h

mov dx, 30h

int 21h

jmp finish

otr:

mov ah, 02h

mov dx, 2dh

int 21h

neg bx

jmp pol

pol:

mov ax,bx

mov cx,0

cwd

savedigits:

mov bx,10

idiv bx

push dx

cwd

inc cx

cmp ax,0

jg savedigits

printdigits:

pop bx

add bx,30h

mov ah, 02h

mov dx, bx

int 21h

loop printdigits

finish:

pop bx

Полный текст программы находится в приложении 2.

6. Тестирование

В таблице 2 приведены тестовые данные, а в таблице 3 результаты тестирования

Таблица 2

Тестовые данные

|  |  |  |  |
| --- | --- | --- | --- |
| Исходные данные | Тестируемый модуль | Ожидаемые результаты | Номер теста |
| N = 2, j = 5 | Summa7.asm | 3 | 1 |
| N = 10, j = 8 | Summa7.asm | 38 | 2 |
| N = 5, j = 9 | Summa7.asm | 8 | 3 |
| N = 20, j = -5 | Summa7.asm | -360 | 4 |
| N = 10, j = -50 | Summa7.asm | -4 | 5 |

Таблица 3

Результаты тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата и время | Тестируемый модуль | Номер теста | Тестер | Описание теста | Результат тестирования |
| 25.11.2022 11:48 | Summa7.asm | 1 | Пикалова А. С. | Общий тест | Успех |
| 26.11.2022 12:15 | Summa7.asm | 2 | Пикалова А. С. | Тест с отрицательным результатом | Успех |
| 27.11.2022 12:19 | Summa7.asm | 3 | Пикалова А. С. | Тест с маленькими значениями | Успех |
| 28.11.2022 12:30 | Summa7.asm | 4 | Пикалова А. С. | Тест с отрицательным j | Успех |
| 4.12.2022 18:10 | Summa7.asm | 5 | Пикалова А. С. | Тест с большими значениями | Успех |

Заключение

В результате данной курсовой работы была спроектирована и разработана программа, вычисляющая значение суммы. К ней написано руководство (приложение 3)

В ходе разработки программы были освоены основные возможности языка ассемблер и эмулятора emu8086.

Список литературы

1. Ершов, Е.В. Методика и организация самостоятельной работы: учебное пособие. [Текст] / Ершов Е.В., Виноградова Л.Н., Селивановских В.В. // Череповец: ЧГУ, 2015. – 243 с.
2. Юров. В. И. Ю70 Assembler. Учебник для вузов. 2-е изд. – СПб.: Питер, 2007. – 637 с.: ил.
3. AMD x86-64 Architecture Programmer’s Manual Volume 2: System Programming [Книга по руководству программиста по архитектуре AMD x 86-64].
4. Лисицин, Д.В. Программирование на языке ассемблера / Лисицин Д.В., Петров Р.В., Полетаева И.А. // Новосибирск, 2011. – 100 с.
5. Смоленцев М.Ю.С 50 Программирование на языке Ассемблера для32/64-разряд-ных микропроцессоров семейства80x86: Учебное пособие в 3-х час-тях. Часть 1.– Иркутск: ИрГУПС, 2009.–192 с.
6. Пильщиков В.Н. Программирование на языке ассемблера IBM PC / Пильщиков В.Н. // Диалог-МИФИ, 2014. – 288 c.

Приложение 1.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

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

|  |  |  |  |
| --- | --- | --- | --- |
| Институт информационных технологий | | | |
| (наименование структурного подразделения) | | | |
| Кафедра математического и программного 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.  «\_\_\_» октября 2022 г. | |
|  | |  | |
| ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ НИЗКОГО УРОВНЯ | | | |
| Техническое задание на курсовой работу  Листов 6 | | | |
|  | Руководитель: Виноградова Л. Н. | | |
|  | Исполнитель: студентка гр. 1ПИб-01-1оп-21 | | |
|  | | | Пикалова А. С. |
| 2022 год | | | |

Введение

Курсовая работа направленна на разработку подпрограммы для процессора 8086. Данная подпрограмма позволит решать арифметическую операцию. В первую очередь, это будет актуально для дальнейшей разработки подпрограмм для процессора 8086.

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: 12 октября 2022 года.

Наименование темы разработки: Программирование на языке низкого уровня.

2. Назначение разработки

Данная подпрограмма позволит на конкретном примере разобраться как устроено программирование на ассемблере, а также может пригодиться при дальнейшем написании подпрограмм.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Подпрограмма должна:

* Выполняться через вызов пользовательского прерывания (например, INT 60h);
* Высчитывать сумму по формуле ;
* Иметь адрес подпрограммы, который должен быть занесен в таблицу векторов прерываний при помощи функций DOS 25h и 35h.;
* Выполнять действия, указанные в конкретном задании, при этом подпрограмме должны передаваться параметры N и j;
* Возвращать результаты работы в регистрах общего назначения;
* Восстановить адрес старого обработчика прерывания при помощи тех же функций DOS.

3.2. Требования к надежности

* Подпрограмма должна работать с процессором 8086;
* Подпрограмма должна обеспечивать бесперебойную работу пользователя;
* Подпрограмма должна выдавать сообщения о возникающих ошибках при неверном задании исходных данных;
* Система должна осуществлять контроль над вводимыми данными и обеспечить целостность хранимой информации. Этот контроль заключается в проверке на полноту и правильность форматов вводимой информации. Для параметра N должно быть ограничение: 90>N> 4. Для параметра j должно быть ограничение: 50>j> -50 не включая -1.

3.3. Условия эксплуатации

* Подпрограмма должна выполняться на процессоре 8086;
* Система должна быть устойчивой, то есть способной правильно выполнять запланированные действия.

3.4. Требования к составу и параметрам технических средств

* В состав технических средств должен входить IВМ-совместимый персональный компьютер (ПЭВМ), выполняющий роль сервера, включающий в себя:
* Процессор не младше чем 8086;
* Оперативную память объемом, 1Гигабайт, не менее;
* Операционную систему Windows 2000 Server или Windows 2003 и выше.

3.5. Требования к информационной и программной совместимости

* Windows 2003 или более поздней версии;
* Процессор не младше чем 8086;
* Использование Intel-синтаксиса.

3.6. Требования к маркировке и упаковке

Подпрограмма должна храниться в выделенной для нее памяти.

3.7. Требования к транспортированию и хранению

Подпрограмма должна при транспортировке храниться на носителе информации.

3.8. Специальные требования

Требования не предъявляются

4. Требование к программной документации

4.1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

* Титульный лист
* Оглавление
* Введение
* Изучение предметной области
* Постановка задачи
* Выбор структур данных
* Логическое проектирование
* Физическое проектирование
* Кодирование
* Тестирование
* Заключение
* Литература
* Приложения

4.2. Требования к оформлению

Оформление документов должно быть выполнено в едином стиле.

Документ:

Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.

Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.

Листы и диск в конверте вложены в пластиковую папку скоросшивателя.

Страницы:

Ориентация – книжная; отдельные страницы, при необходимости, альбомные.

Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см.

Абзацы:

Межстрочный интервал – 1,5. Отступ первой строки абзаца – 0,75 см. Расстояние между последней строкой текста предыдущего раздела и последующим заголовком при расположении их на одной странице должно быть равно трем интервалам.

Шрифты:

Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 8 (возможно в 2 колонки).

Рисунки:

Подписывается под ним по центру: Рис.Х. Название

В приложениях: Рис.П1.3. Название.

Таблицы:

Подписывается: над таблицей, выравнивание по правому: «Таблица Х».

В следующей строке по центру Название

Надписи в «шапке» (имена столбцов, полей) – по центру.

В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому.

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице П1.2.

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Получение задания | 12.10.2022 | Полученное задание |  |
| Разработка технического задания | 20.10.2022-24.10.2022 | Оформленное техническое задание |  |
| Разработка алгоритма | 28.10.2022 – 3.11.2022 | Готовый алгоритм |  |
| Написание программы | 4.11.2022 – 22.11.2022 | Написанная программа |  |
| Тестирование программы | 25.11.2022 – 5.12.2022 | Проверенная и отлаженная программа |  |
| Написание РПЗ | 10.12.2022-15.12.2022 | Оформленное РПЗ |  |

6. Порядок контроля и приёмки

Порядок контроля и приёма представлены в таблице П1.3.

Таблица П1.3

Порядок контроля и приёма

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приёмке результата контрольного этапа |
| Технические задание | 21.10.2022-25.10.2022 | Оформленное техническое задание |  |
| Теоретическая часть курсовой работы | 1.11.2022-4.11.2022 | Оформленная теоретическая часть |  |
| Практическая часть курсовой работы | 3.12.2022-6.12.2022 | Программа |  |
| Расчетно-пояснительная записка | 16.12.2022-18.12.2022 | Оформленная РПЗ |  |
| Защита курсовой работы | 19.12.2022-26.12.2022 | Получение итоговой оценки за курсовую работу |  |

Приложение 2.

Текст программы

.stack 100h

.data

savecs dw 0

saveip dw 0

message1 db 'Enter N: $'

message2 db 'Enter j: $'

message3 db 'Summ = $'

j DW 0

N Db 0

i dw 1

ml dw 7

res DW 0

negative DB 0

.code

start:

mov ax,@data

mov ds,ax

mov ax,3h

int 10h

call vvod

callvvod:

mov ah,35h

mov al,60h

int 21h

mov savecs,es

mov saveip,bx

push ds

mov dx,offset summa

mov ax,seg summa

mov ds,ax

mov ah,25h

mov al,60h

int 21h

pop ds

int 60h

push ds

mov dx,savecs

mov ax,saveip

mov ds,ax

mov ah,25h

mov al,60h

int 21h

pop ds

call vivod

vvod:

mov ah,09h

mov dx,offset message2

int 21h

mov ch,'0'

mov ah, 01

mov cl,13

cycle:

int 21h

cmp al,cl

je end\_cikl

cmp al,45

jne minus

mov negative,1

jmp cycle

minus:mov bl, al

sub bl, 48

mov ax, 10

mov ah,00h

imul j

add ax,bx

mov j,ax

mov ah,1h

jmp cycle

end\_cikl:

mov al,negative

cmp al,0

je minus2

neg j

minus2:

mov dl,10

mov ah,2h

int 21h

mov ah,09h

mov dx,offset message1

int 21h

mov ah,1h

int 21h

mov N,al

sub N,48

jmp callvvod

summa proc far

mov CX,0

mov Cl,N

L:mov ax,i

imul ml

mov BX, j

inc BX

CWD

IDIV BX

add i,1

add res, ax

loop L

exit:

iret

summa endp

vivod:

mov dl,10

mov ah,2h

int 21h

mov dl,13

int 21h

mov ah,09h

mov dx,offset message3

int 21h

mov bx,res

cmp bx, 1000000000000000b

jb minus3

mov dl,45

mov ah,2h

int 21h

neg bx

minus3:

mov ax,bx

call print\_ax

jmp exit2

print\_ax proc

cmp ax, 0

jne print\_ax\_r

push ax

mov al, '0'

mov ah, 0eh

int 10h

pop ax

ret

print\_ax\_r:

pusha

mov dx, 0

cmp ax, 0

je pn\_done

mov bx, 10

div bx

call print\_ax\_r

mov ax, dx

add al, 30h

mov ah, 0eh

int 10h

jmp pn\_done

pn\_done:

popa

ret

endp

exit2:

mov ax,0

int 21h

end start

Приложение 3.

Руководство пользователя

1. Общие сведения о программе

Файл программы называется summa7.asm. При запуске программа вычисляет значение суммы.

Рис. П.3.0. Формула

2. Описание установки

Программа не требует установки. Требуется только наличие доступа к файлу программы и установленный Emu8086.

3. Описание запуска

Двойным щелчком мыши запустить программу summa7.asm (рис. П.3.1).



Рис. П.3.1. Файл

После этого откроется окно emu8086 с программой (рис. П.3.2).

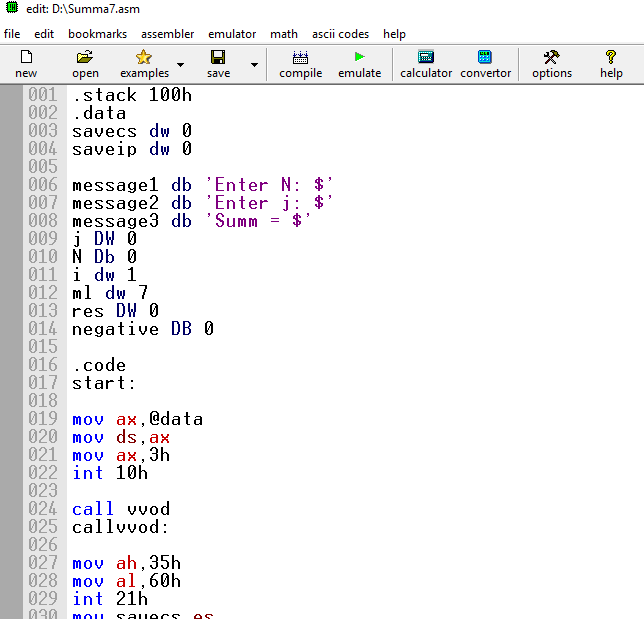


Рис.П3.2. Окно emu8086

Нажимаем кнопку “emulate” (на рис. П3.2 обведена красным). Появляется несколько окон (рис. П3.3).

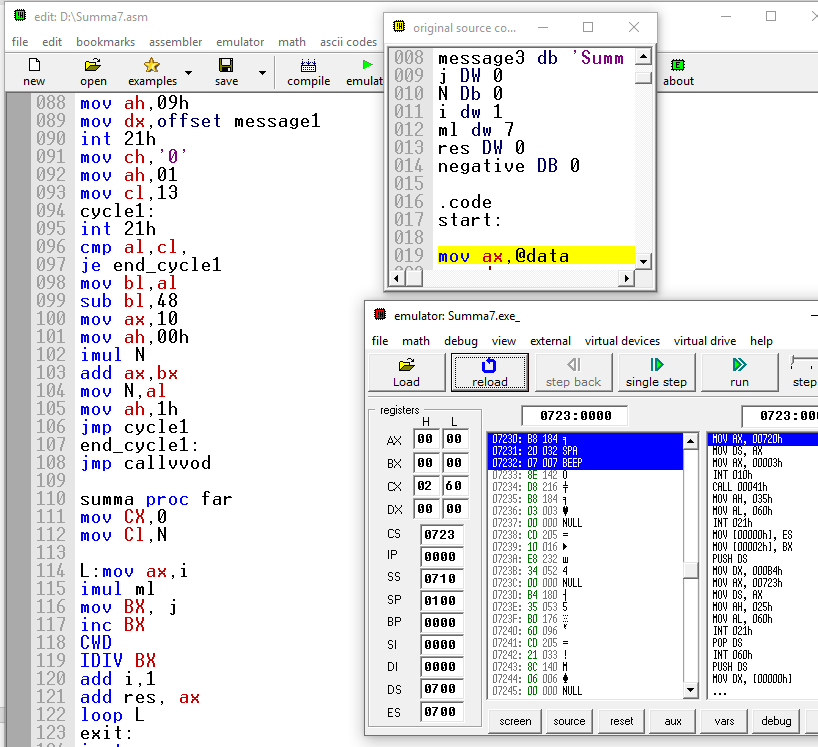


Рис. П3.3. Окна тестирования

Нажимаем кнопку “run” (на рис. П3.3 обведена красным). Появляется окно консоли. Программа попросит значения N и j. Введя значение j, нажимается пробел. Аналогично вводится значение N. После ввода значений программа выводит ответ (рис. П3.4)

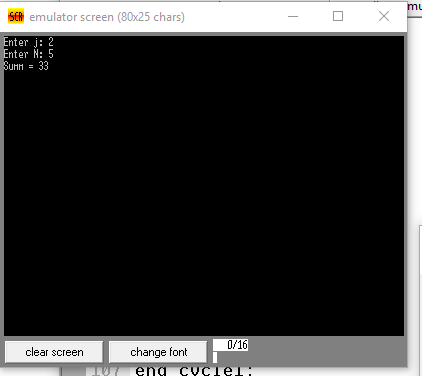


Рис.П3.4. Ввод переменных и ответ

4. Описание ошибок

При вводе значения N, которое равно -1, программа выведет сообщение об ошибке. Подобное будет происходить до тех пор, пока не будут введены допустимые значения. При вводе j допускается первым написать символ минус, тогда значение j будет обрабатываться как отрицательное число, все остальные символы кроме символов, которые являются цифрами, приведут к возобновлению ввода j. Программа правильно будет считать сумму ряда, если значение N находится в диапазоне от 4 до 90 всё включительно и если значение j находится в диапазоне от -50 до 50 не включая -1.