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-02-2оп-22 | |
|  | | | *группа* | |
|  | | | Направление подготовки (специальности) | |
|  | | | 09.03.04 Программная инженерия | |
|  | | | *шифр, наименование* | |
|  | | | Тихомиров Владислав Валерьевич | |
|  | | | *фамилия, имя, отчество* | |
|  | | | Руководитель | |
|  | | | Виноградова Людмила Николаевна | |
|  | | | *фамилия, имя, отчество* | |
|  | | | Доцент | |
|  | | | *должность* | |
|  | | | Дата представления работы | |
|  | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |
|  | | | Заключение о допуске к защите | |
|  | | |  | |
|  | |
|  | |
|  | | | Оценка \_\_\_\_\_\_ , \_\_\_\_\_\_\_\_\_\_\_ | |
|  | | |  | Количество баллов |
|  | | | Подпись преподавателя \_\_\_\_\_\_\_\_ | |
|  | | |  | |
|  | |  | | |
|  | | Череповец, 2023 | | |
| *год* | | | | |

Аннотация

Курсовая работа по предмету: «Программирование на языке низкого уровня» выполнена студентом группы 1ПИб-02-2оп-22 Института информационных технологий Тихомировым Владиславом Валерьевичем.

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 7](#_Toc121598421)

[3. Логическое проектирование 8](#_Toc121598422)

[4. Физическое проектирование 8](#_Toc121598423)

[5. Кодирование 9](#_Toc121598424)

[6. Тестирование 10](#_Toc121598425)

[Заключение 13](#_Toc121598426)

[Список литературы 14](#_Toc121598427)

[Приложение 1. Техническое задание 14](#_Toc121598428)

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

[Приложение 3. Руководство пользователя 25](#_Toc121598452)

Введение

Микропроцессоры американской компании Intel и персональные компьютеры на их основе прошли не очень длинный во времени, но значительный, по существу, путь развития, на протяжении которого кардинально изменялись их возможности и даже сами принципы их работы, например архитектура.

В то же время, внося в микропроцессор принципиальные изменения, разработчики были вынуждены постоянно иметь в виду необходимость обеспечения совместимости новых моделей со старыми, чтобы не отпугивать потенциального покупателя перспективой полной замены освоенного или разработанного им программного обеспечения. В результате современные микропроцессоры типа Intel core, обеспечивая такие возможности, как 64-битную адресацию почти неограниченных объемов памяти, многозадачный 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 в режиме первых микропроцессоров типа 8086, используя всего лишь мегабайт оперативной памяти, 16-разрядные операнды (т. е. числа в диапазоне до ) и ограниченный состав команд. Поскольку программирование на языке ассемблера напрямую затрагивает аппаратные возможности микропроцессора, прежде всего, следует выяснить, в какой степени программист может использовать новые возможности микропроцессоров в своих программах, и какие проблемы программной несовместимости могут при этом возникнуть.

Первые персональные компьютеры корпорации IBM, появившиеся в 1981 г. и получившие название 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 получение практических навыков работы программирования на языке ассемблера.

Итогом выполнения курсовой работы является разработка программы, вычисляющей значение произведения.

1. Изучение и описание предметной области

Основной командой передачи данных является команда MOV, осуществляющая операцию присваивания: MOV приемник, источник Команда MOV присваивает значению операнда приемника значение операнда источника. В качестве приемника могут выступать регистр общего назначения, сегментный регистр или ячейка памяти, в качестве источника могут выступать константа, регистр общего назначения, сегментный регистр или ячейка памяти. Оба операнда должны быть одного размера (байты, слова или двойные слова, главное одновременно оба).

Микропроцессор может передавать данные в порты ввода-вывода, которые поддерживаются аппаратно и используют соответствующие своим предназначениям линии ввода-вывода процессора. Аппаратное адресное пространство ввода-вывода процессора не является физическим адресным пространством памяти. Адресное пространство ввода-вывода состоит из 64 килобайт индивидуально адресуемых 8-битных портов ввода-вывода, имеющих адреса от 0 до FFFFh. Адреса от 0F8h до 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 корректного изменения размера операнда, заданного неявно в регистре-аккумуляторе (EAX, AX, AL). Непосредственно после аббревиатуры команды операнд не указывается.

Команды прерываний предназначены для управления программными прерываниями. Прерывание – это, как правило, асинхронная остановка работы процессора, вызванная началом работы устройства ввода-вывода. Исключением являются синхронные прерывания, возникающие при определении некоторых предопределенных условий в процессе выполнения команды. Когда поступает сигнал о прерывании, процессор останавливает выполнение текущей программы и переключается на выполнение обработчика прерывания, заранее записанного для каждого прерывания. Архитектура IA-32 поддерживает 17 векторов аппаратных прерываний и 224 пользовательских. Команда INT вызывает обработчик указанного операндом прерывания (константой). Операнд определяет номер вектора системного прерывания BIOS от 0 до 255, представленный в виде беззнакового 8-битного целого числа. При вызове обработчика прерывания в стеке сохраняются регистры EIP, CS и EFLAGS. Прерывание по переполнению вызывается отдельной командой INTO и имеет вектор 04h.

Команды организации циклов — используют регистр ECX по умолчанию в качестве счетчика числа повторений цикла. Каждый раз при выполнении команды LOOPсс значение регистра ECX уменьшается на 1, а затем сравнивается с 0. Если ECX=0, выполнение цикла заканчивается, и продолжает выполняться код программы, записанный после команды LOOPcc. Если ECX содержит ненулевое значение, то осуществляется переход по адресу операнда команды LOOPcc.

2. Постановка задачи

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



Рис. 1. Формула

3. Логическое проектирование

Программа сначала инициализирует сегменты для данных и стека, а также запоминает текущий обработчик прерывания INT 60h. Затем устанавливается новый обработчик, который выполняет расчёт формулы. Пользователь вводит два числа: N — количество итераций, и j — параметр формулы.

После этого программа вызывает пользовательское прерывание INT 60h, которое считает произведение ряда по формуле i \* (j+7) для каждого значения i от 1 до N. Результат вычислений сохраняется в регистре AX.

Затем этот результат преобразуется в строку с помощью отдельной подпрограммы, и строка выводится на экран. После завершения всех операций восстанавливается старый обработчик прерывания, и программа завершает свою работу.

4. Физическое проектирование

Программа будет состоять из одного файла, в котором будет содержаться код программы на языке ассемблера. Планируемые функции и их параметры представлены в таблице 1.

Таблица 1

Физическое проектирование

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Функция | Параметры | Выполняемое действие |
| Сalcstr.asm | 21H/09H | DX (адрес строки) | Выводит строку на экран |
| Calcres.asm | 21H/35H | AH = 35H | Получает текущий вектор прерывания 60H и сохраняет его |
| Calcres.asm | 21H/25H | DX (адрес CalcFormula) | Устанавливает новый вектор прерывания 60H |
| Calcres.asm | 60H | BX (2), CX (3) | Выполняет вычисления, вызывая обработчик прерывания 60H |
| Calcres.asm | 21H/16H | AX = 0 | |  | | --- | | Ждет нажатие клавиши |  |  | | --- | |  | |
| Calcres.asm | |  |  |  | | --- | --- | --- | | |  | | --- | | 4CH/00H |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | AH = 4CH |  |  | | --- | |  | |  |  | | --- | |  | | Завершает выполнение программы |

5. Кодирование

Сегмент данных:

data segment

strbuf db 10 dup(0)

Old\_60h dw 0,0

data ends

Сегмент стека, необходимый для работы программы:

stacks segment word stack 'stack'

dw 200 dup(?)

StkTOp LABEL word

stacks ends

Функция ConvertToString, которая преобразует число в строку, используя деление на 10 и сохранение цифр в обратном порядке:

ConvertToString proc

push bp

mov bp,sp

mov ax,[bp+4]

mov di,[bp+6]

xor dx,dx

xor si,si

mov bx, 10

\_convloop:

xor dx,dx

div bx

push dx

inc si

test ax,ax

jnz \_convloop

\_stacklopp:

pop dx

add dl,"0"

mov [di],dl

inc di

dec si

jnz \_stacklopp

mov byte ptr [di],'$'

pop bp

ret 4

ConvertToString endp

Функция CalcFormula, которая вычисляет произведение П=i⋅(j+7):

CalcFormula proc add bx,7

mov di,1

mov dx,di

start\_calc:

mov ax,di

mul bl

mul dx

mov dx,ax

inc di

cmp di,cx

jbe start\_calc

mov ax,dx ; iret

CalcFormula endp

Основная программа start:

start:

mov sp,StkTop

mov ax,data

mov es,ax

mov ds,ax

; Сохраняем старый обработчик прерывания 60h

mov ax, 3560h

int 21h

mov Old\_60h, bx

mov Old\_60h+2, es

; Устанавливаем новый обработчик прерывания 60h

cli

push ds

push cs

pop ds

mov ax, 2560h

lea dx, CalcFormula

int 21h

pop ds

sti

; Подготовка параметров для CalcFormula

mov bx,2

mov cx,3

int 60h

lea dx,[strbuf]

push dx

push ax

call ConvertToString

; Вывод результата на экран

lea dx,strbuf

mov ah,09h

int 21h ; Вызов DOS-прерывания

; Ожидание нажатия клавиши

xor ax,ax

int 16h

; Восстановление старого обработчика прерывания

cli

push ds

mov ax, Old\_60h+2

push ax

pop ds

mov dx, Old\_60h

mov ax, 2560h

int 21h

pop ds sti

mov ax,4c00h

int 21h

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 = 3 | calcres.asm | 200 | 1 |
| N = 3, j = 2 | calcres.asm | 4374 | 2 |
| N = 2, j = -7 | calcres.asm | 0 | 3 |
| N = 2, j = -1 | calcres.asm | 72 | 4 |
| N = 7, j = -6 | calcres.asm | 5040 | 5 |

Таблица 3

Результаты тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата и время | Тестируемый модуль | Номер теста | Тестер | Описание теста | Результат тестирования |
| 20.11.2023 10:10 | calcres.asm | 1 | Тихомиров В. В. | Общий тест | Успех |
| 20.11.2023 20:37 | calcres.asm | 2 | Тихомиров В. В. | Тест с маленькими значениями | Успех |
| 21.11.2023 14:41 | calcres.asm | 3 | Тихомиров В. В. | Тест с нулевым результатом | Успех |
| 25.11.2023 12:45 | calcres.asm | 4 | Тихомиров В. В. | Тест с отрицательным j | Успех |
| 27.11.2023 01:13 | calcres.asm | 5 | Тихомиров В. В. | Тест с большими значениями | Успех |

Заключение

В результате данной курсовой работы была спроектирована и разработана программа, вычисляющая значение произведения. 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. Виноградова Л.Н. Системное программирование: Учеб. пособие. – Череповец: ФГБОУ ВПО ЧГУ, 2016. – 210 с.

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 2023 г. |
|  | |  |
| ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ НИЗКОГО УРОВНЯ | | |
| Техническое задание на курсовой работу  Листов 6 | | |
|  | Руководитель: Виноградова Л. Н. | |
|  | Исполнитель: студент гр. 1ПИб-02-2оп-22 | |
| Тихомиров В. В. | | |
|  | | |
|  | | |
|  | | |
|  | | |

2023 год

Введение

Данная курсовая работа нацелена на написание программы на языке ассемблера, вычисляющей значение произведения(рис. П.1.1).



Рис. П.1.1. Формула

1. Основания для разработки

Разработка ведётся на основании задания на курсовую работу по дисциплине «Программирование на ассемблере», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Тема разработки: «Программирование на языке низкого уровня.»

2. Назначение разработки

Основная задача курсовой работы: освоить на практике материал, полученный в ходе изучения дисциплины «Программирование на ассемблере» и программу-эмулятор процессора Intel 8086 «emu8086».

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 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.

3.2. Требования к надёжности

Программа должна корректно работать для нескольких различных значений параметров N и j, исключая тривиальные (например, 1 или 0). Параметр N должен быть больше 1 и находиться в пределах от 2 до 50 включительно, а параметр j — в диапазоне от -8 до 9 включительно.

3.3. Условия эксплуатации

Персональный компьютер должен эксплуатироваться в закрытом помещении с постоянным контролем микроклимата, обеспечивающим его надежную работу. Температура окружающего воздуха должна находиться в пределах от +10°C до +35°C, а относительная влажность — от 30% до 80%, без резких колебаний. Атмосферное давление должно составлять от 84 до 107 кПа.

3.4. Требования к составу и параметрам технических средств

Минимальные системные требования:

* Операционная система: Windows (версии 7 и выше), Linux, macOS;
* Оперативная память: 1 ГБ и более;
* Архитектура процессора: x86-16 совместимая;
* 1 Гб свободного места на диске.

3.5. Требования к информационной и программной совместимости

Код программы написан на языке низкого уровня ассемблер Intel 8086 и скомпилирован в emu8086.

3.6. Требования к маркировке и упаковке

Отсутствуют.

3.7. Требования к транспортированию и хранению

Отсутствуют.

3.8. Специальные требования

Отсутствуют.

4. Требования к программной документации

4.1. Содержание расчётно-пояснительной записки:

* Введение;
* описание предметной области;
* описание созданного приложения;
* заключение;
* список литературы;
* техническое задание;
* руководство пользователя;
* текст программы.

4.2. Требования к оформлению

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений (в табл. П1.1).

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (20х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее — по 2 см, левое — 3 см, правое — 2 см. |
| Абзацы | Межстрочный интервал — 1,5, перед и после абзаца — 0. |
| 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.П.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 — выравнены 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 | 01.10.2023 | Полученное задание |  |
| Разработка технического задания | 11.10.2023-14.10.2023 | Оформленное техническое задание |  |
| Разработка алгоритма | 21.10.2023 – 29.10.2023 | Готовый алгоритм |  |
| Написание программы | 04.11.2023 – 13.11.2023 | Написанная программа |  |
| Тестирование программы | 20.11.2023 – 27.11.2023 | Проверенная и отлаженная программа |  |
| Написание РПЗ | 06.12.2023 | 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 | 12.10.2023-15.10.2023 | Оформленное техническое задание |  |
| Теоретическая часть курсовой работы | 01.11.2023-04.11.2023 | Оформленная теоретическая часть |  |
| Практическая часть курсовой работы | 03.11.2023-14.11.2023 | Программа |  |
| Расчетно-пояснительная записка | 03.12.2023-06.12.2023 | Оформленная РПЗ |  |
| Защита курсовой работы | 19.12.2023-21.12.2023 | Получение итоговой оценки за курсовую работу |  |

Приложение 2.

Текст программы

Name calcstr

data segment

strbuf db 10 dup(0) ; Буфер для строки (длина 10)

Old\_60h dw 0,0 ; Переменная для хранения состояния прерывания 60h

data ends

stacks segment word stack 'stack'

dw 200 dup(?) ; Стек размером 200 слов

StkTOp LABEL word

stacks ends

code segment

assume CS: code, DS: data, ES: data, SS: stacks

; Процедура ConvertToString преобразует число в строку

ConvertToString proc

push bp

mov bp, sp

mov ax, [bp+4] ; Загружаем число (word) в ax

mov di, [bp+6] ; Загружаем адрес строки (buffer) в di

xor dx, dx ; Очищаем dx

xor si, si ; Очищаем si

mov bx, 10 ; Устанавливаем основание системы счисления в 10

; Цикл преобразования числа в строку

\_convloop:

xor dx, dx

div bx ; Делим ax на 10, результат в ax, остаток в dx

push dx ; Сохраняем остаток на стеке

inc si ; Увеличиваем индекс символа

test ax, ax ; Проверяем, делится ли число на 10

jnz \_convloop ; Если результат не равен 0, продолжаем цикл

; Процесс перевода остаточных значений из стека в строку

\_stacklopp:

pop dx ; Извлекаем остаток из стека

add dl, "0" ; Преобразуем остаток в символ (число в ASCII)

mov [di], dl ; Записываем символ в строку

inc di ; Переходим к следующему символу

dec si ; Уменьшаем счетчик

jnz \_stacklopp ; Если есть еще символы, продолжаем

mov byte ptr [di], '$' ; Завершаем строку символом '$'

pop bp

ret 4

ConvertToString endp

; Процедура CalcFormula вычисляет выражение i\*(j+7) для всех i от 1 до N

CalcFormula proc

add bx, 7 ; j + 7

mov di, 1 ; Устанавливаем i = 1

mov dx, di ; Инициализируем dx значением i

start\_calc:

mov ax, di

mul bl ; Умножаем i на (j+7)

mul dx ; Умножаем результат на dx

mov dx, ax ; Сохраняем результат в dx

inc di ; Увеличиваем i на 1

cmp di, cx ; Сравниваем i с N

jbe start\_calc ; Если i <= N, продолжаем

mov ax, dx ; Результат вычислений в ax

iret ; Возврат из процедуры

CalcFormula endp

start:

mov sp, StkTop

mov ax, data

mov es, ax

mov ds, ax

; Сохраняем состояние прерывания 60h (для возврата к нему позже)

mov ax, 3560h

int 21h

mov Old\_60h, bx

mov Old\_60h+2, es

; Включаем прерывание 60h для дальнейшей обработки

cli

push ds

push cs

pop ds

mov ax, 2560h

lea dx, CalcFormula

int 21h

pop ds

sti

; Настроим параметры для выполнения вычислений

mov bx, 2 ; j

mov cx, 3 ; N

int 60h

; Преобразуем результат вычислений в строку

lea dx, [strbuf] ; Адрес буфера для строки

push dx ; Сохраняем адрес буфера

push ax ; Сохраняем результат вычислений

call ConvertToString ; Вызываем процедуру для преобразования числа в строку

; Выводим строку на экран

lea dx, strbuf ; Адрес строки

mov ah, 09h ; Устанавливаем функцию для вывода строки

int 21h

; Ожидаем нажатие клавиши

xor ax, ax

int 16h

; Восстанавливаем состояние прерывания 60h

cli

push ds

mov ax, Old\_60h+2

push ax

pop ds

mov dx, Old\_60h

mov ax, 2560h

int 21h

pop ds

sti

; Завершаем выполнение программы

mov ax, 4c00h

int 21h

code ends

end start

Приложение 3.

Руководство пользователя

1. Общие сведения о программе

Файл программы называется calcstr.asm. При запуске программа вычисляет значение произведения.



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

Вставить диск в привод компьютера. Двойным щелчком мыши запустить программу calcstr.asm из коренной папки. (рис. П.3.1).



Рис. П.3.1. Файл

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

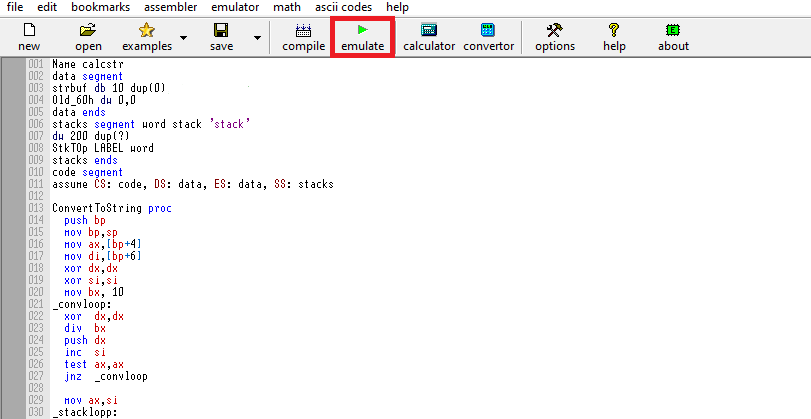


Рис.П3.2. Окно emu8086

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

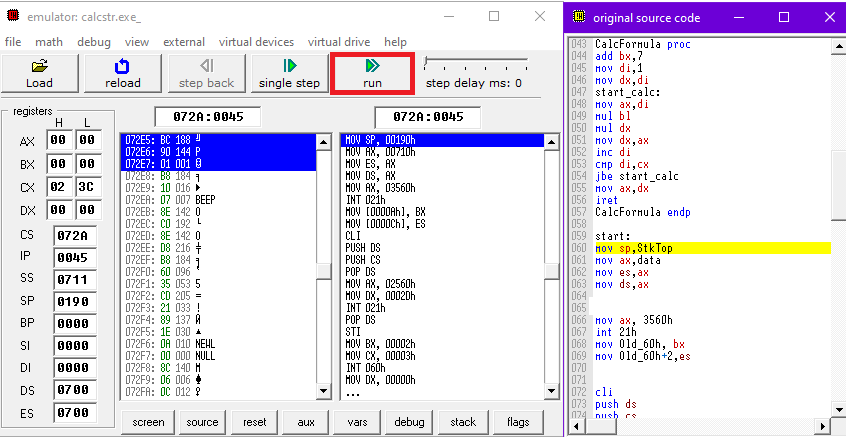


Рис. П3.3. Окна тестирования

Нажимаем кнопку “run” (на рис. П3.3 обведена красным). Появляется окно консоли. Программа выводит ответ (рис. П3.4)

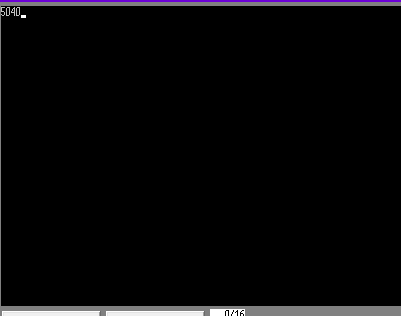


Рис.П3.4. Ответ

Для ввода N и j необходимо перейти к 83 и 84 строке кода и поменять там значения (рис. П3.5)

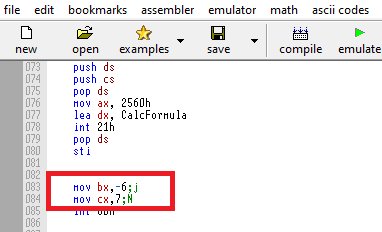


Рис.П3.5. Переменные