ИМИТАТОР РАБОТЫ ПРОЦЕССОРА

Введение.

В своей курсовой работе я попытался изложить общую схему работы процессора и проиллюстрировать ее с помощью программы-имитатора.

В первой части курсовой работы мною изложены основные теоретические сведения о логическом устройстве микропроцессора, его предназначении и принципах работы. При этом я не ориентировался на конкретный тип процессора.

Во второй части я рассматриваю руководство пользователя к своей программе-имитатору. Данная программа написана в среде Delphi. При написании данной программы я ориентировался на процессор типа PDP, как наиболее подходящего для изучения и наиболее часто рассматриваемого в учебниках информатики.

Микропроцессор как основа ЭВМ.

Внутренняя структура микропроцессора.

Любая ЭВМ предназначена для обработки информации причем, как правило, осуществляет эту обработку опосредовано – представляя информацию в виде чисел. Для работы с числами машина имеет специальную важнейшую часть – *микропроцессор*. Это универсальное логическое устройство, которое оперирует с двоичными числами, осуществляя простейшие логические и математические операции, и не просто как придется, а в соответствии с программой, т.е. в заданной последовательности. Для хранения этой заданной последовательности служат запоминающие устройства – ЗУ. ЗУ бывают постоянными – ПЗУ, в которых информация хранится, не изменяясь сколь угодно долго, и оперативными – ОЗУ, информация в которых может быть изменена в любой момент в соответствии с результатами ее обработки. Процессор общается с ОЗУ и ПЗУ через так называемое адресное пространство, в котором каждая ячейка памяти имеет свой адрес.

МП состоит из набора регистров памяти различного назначения, которые определенным образом связаны между собой и обрабатываются в соответствии с некоторой системой правил. *Регистр* – это устройство, предназначенное для хранения и обработки двоичного кода. К внутренним регистрам процессора относят: счетчик адреса команд, указатель стека, регистр состояний, регистры общего назначения.

Наличие счетчика команд было положено еще в работах фон Неймана. Роль счетчика состоит в сохранении адреса очередной команды программы и автоматическом вычислении адреса следующей. Благодаря наличию программного счетчика в ЭВМ реализуется основной цикл исполнения последовательно расположенных команд программы.

*Стек* – это особый способ организации памяти, при использовании которого достаточно сохранять адрес последней заполненной ячейки ОЗУ. Именно адрес последней заполненной ячейки ОЗУ и хранится в указателе стека. Стек используется процессором для организации механизма прерываний, обработки обращения к подпрограммам, передачи параметров и временного хранения данных.

В регистре состояний хранятся сведения о текущих режимах работы процессора. Сюда же помещается информация о результатах выполняемых команд, например: равен ли результат нулю, отрицателен ли он, не возникли ли в ходе операции ошибки и т.п. Использование и анализ в этом регистре происходит побитно, каждый бит регистра имеет самостоятельное значение.

Регистры общего назначения (РОН) служат для хранения текущих обрабатываемых данных или их адреса в ОЗУ. У некоторых процессоров регистры функционально равнозначны, в других назначение регистров строго оговаривается. Информация из одного регистра может предаваться в другой.

Система команд микропроцессора

Несмотря на бурную эволюцию вычислительной техники, основной набор команд довольно слабо изменился. Система команд любой ЭВМ обязательно содержит следующие группы команд обработки информации.

Команды передачи данных (перепись), копирующие информацию из одного места в другое.

Арифметические операции, к которым в основном относят операции сложения и вычитания. Умножение и деление обычно реализуется с помощью специальных программ.

Логические операции, позволяющие компьютеру производить анализ получаемой информации. Простейшими примерами команд рассматриваемой группы могут служить сравнение, а также известные логические операции *и, или, не*.

Сдвиги двоичного кода влево и вправо. В некоторых случаях сдвиги используются для реализации умножения и деления.

Команды ввода и вывода информации для обмена с внешними устройствами. В некоторых ЭВМ внешние устройства являются специальными служебными адресами памяти, поэтому ввод и вывод осуществляется с помощью команд переписи.

Команды управления, реализующие нелинейные алгоритмы. Сюда относят условный и безусловный переходы, а также команды обращения к подпрограмме (переход с возвратом). Часто к этой группе относят операции по управлению процессором типа останов или нет операции.

Любая команда ЭВМ обычно состоит из двух частей – операционной и адресной. Операционная часть называемая также кодом операции указывает, какое действие необходимо выполнить с информацией. Операционная часть имеется у любой команды. Адресная часть описывает, где используемая информация хранится и куда поместить результат. В некоторых командах управления работой машины адресная часть может отсутствовать, например, в команде останова.

Код операции можно представить себе как некоторый условный номер в общем списке команд. В основном этот список построен в соответствии с определенными внутренними закономерностями.

Адресная часть обладает значительно большим разнообразием. Основу адресной части составляет операнд. В зависимости от количества возможных операндов команды могут быть одно- и двухадресные. В двухадресных командах результат записывается либо в специальный регистр (сумматор), либо вместо одного из операндов.

Способы адресации данных

Способы (или методы) адресации не что иное, как способы указания на те или иные ячейки памяти, с которыми должен манипулировать оператор. Существует много различных методов адресации. Количество их зависит от типа процессора. Наличие большого количества способов адресации обеспечивает высокую гибкость в построении программ и является большим преимуществом системы команд данного типа ЭВМ. Способы адресации практически одинаковы для всех команд, в которых присутствуют операнды. В своей курсовой я подробно рассматриваю только три основных метода адресации, которые применяются почти во всех процессорах.

*Регистровая адресация*. При этом способе операндом является один из регистров общего назначения. Число хранится непосредственно в регистре. Записывается как Rn, где n – номер регистра.

*Косвенная адресация*. При этом способе адресации в одном из регистров общего назначения содержится не само число, с которым нужно работать, а его адрес, то есть номер ячейки памяти, в котором число находится. Записывается как (Rn), где n – номер регистра.

*Автоинкрементная адресация*. Этот вид адресации несколько сложнее двух предыдущих. Помимо основного действия (косвенного обращения к ячейке памяти), при использовании этого метода, происходит еще изменение адреса этого обращения. В данном случае увеличивается указатель адреса ячейки памяти, к которой мы обращаемся, то есть содержимое регистра, служащего указателем адреса. Данное увеличение происходит автоматически, без какой-либо команды. Записывается эта адресация как (Rn)+. То, что знак + стоит после имени регистра, намекает на порядок выполнения команды: сначала происходит операция с ячейкой, на которую указывает адрес помещенный в регистр Rn, а потом уже содержимое регистра увеличивается на 2 (если оператор работает со словом, то переход к адресу следующего слова), или на 1 (если оператор работает с байтом, переход к адресу следующего байта). Данный способ адресации применяется для работы с массивами и при использовании стека (например, при использовании подпрограмм).

Существует еще один особый способ адресации, который рассматривается отдельно. Речь идет о работе со стеком. *Стек* – неявный способ адресации данных, при котором информация записывается и считывается только последовательным образом с использованием указателя стека. Стек всегда имеет единственный вход и выход информации – для хранения его адреса и нужен указатель стека. При записи данных в стек процессор проделает следующее:

* уменьшит указатель стека на 2 (целое число занимает в памяти 2 байта);
* запишет данные по полученному адресу.
* При извлечении данных из стека процессор проделает следу шее:
* считает данные из стека;
* увеличит указатель на 2.

В командах работы со стеком адрес ОЗУ не фигурирует в явном виде. Но при этом молчаливо предполагается, что указатель стека уже задан. При задании указателя надо быть внимательным. Если указатель стека определен неправильно, то запись в стек может разрушить полезную информацию в ОЗУ.

Таким образом, мы рассмотрели способы адресации информации, которые существуют почти во всех типах процессора и которые я использовал в своем имитаторе.

Руководство пользователя.

Описание программы

Изложенную выше структуру микропроцессора я попытался реализовать в своей программе.

Основную часть программы составляет *Редактор*. Он представляет собой место, где вводится программа, которую выполняет процессор. Каждая команда вводится в свою ячейку, имеющую адрес. Размер адресного пространства редактора составляет 1Кб. Начальная ячейка имеет адрес 1000, конечная – 2024. В качестве операнда в одно- и двухадресных командах выступает один из регистров общего назначения. Я выделил следующую систему команд.

Одноадресные команды.

Представлены в следующей форме: *Операция операнд ОП1.*

Очистить ОП1– обнуляет значение операнда.

Увеличить на 1 ОП1 – увеличивает значение операнда.

Уменьшить на 1 ОП1 – уменьшает значение операнда.

Двухадресные команды.

Представлены в следующей форме: *Операция первый операнд ОП1 второй операнд ОП2.*

Переслать ОП1 в ОП2 – пересылает значение первого операнда во второй операнд.

Добавить ОП1 к ОП2– добавляет значение первого операнда ко второму операнду, результат во втором операнде.

Вычесть ОП1 из ОП2 – вычитает значение первого операнда из второго операнда, результат во втором операнде.

Сравнить ОП1 с ОП2 – сравнивает разность второго и первого операнда с нулем, значения операндов не меняются, результат влияет на состояние регистра состояний.

Безадресные команды.

Возврат из подпрограммы – осуществляет возврат из подпрограммы в ячейку, следующую за командой вызвавшей эту подпрограмму, используется только в подпрограммах

Стоп – команда останова, ставится обязательно в конце программы, после ее выполнения ни какие команды не выполняются.

Команды перехода.

Переход на К слов – безусловный переход – осуществляет переход на К слов.

Вызов подпрограммы по адресу – переход на адрес К с запоминанием адреса возврата для команды возврата из подпрограммы.

Если <> переход на К слов – переход на К слов, если результат <>0.

Если = переход на К слов – переход на К слов, если результат =0.

Если >= переход на К слов – переход на К слов, если результат >=0.

Если > переход на К слов – переход на К слов, если результат >0.

Если < переход на К слов – переход на К слов, если результат <0.

Если <= переход на К слов – переход на К слов, если результат <=0.

Работа команды условного перехода осуществляется следующим образом: если анализируемое условие справедливо, то переход происходит. В противном случае никаких действий не производится, а значит, переход игнорируется и процессор, как обычно, выбирает следующую команду. Справедливость анализируемого условия определяется по состоянию регистра состояний, которое в свою очередь зависит от результата команды сравнения.

Таким образом, выше изложена операционная часть команд, которые выполняет мой имитатор. Посмотрим, какими способами могут быть представлены операнды ОП1 и ОП2. Для своего проекта я отобрал три наиболее простых из имеющихся в реальных процессорах метода адресации. Это регистровый, косвенный и автоинкрементный способы адресации, подробно они рассматриваются в первой части моей курсовой работы.

Не менее важной частью программы является часть *Регистры,* в которой расположены регистры используемые в моем проекте. Для своего имитатора процессора я решил использовать восемь шестнадцати разрядных регистров общего назначения (c R0 по R7) и особого регистра, в котором отражается текущее состояние процессора – регистра состояний PS.

Любой из регистров общего назначения может использоваться в командах на равных основаниях. Вместе с тем имеется два выделенных регистра, содержимое которых моя программа использует для собственных нужд. Прежде всего это регистр R7, выполняющий роль счетчика команд. Другим выделенным регистром является указатель стека R6. Остальные 6 регистров, обозначаемые R0 - R5, пользователь может использовать по своему усмотрению. Уместно заметить, что выделенные регистры программы R6 и R7 могут использоваться в любой команде наряду с обычными регистрами общего назначения R0 - R5. Как уже упоминалось регистры обшего назначения шестнадцати разрядные, то есть диапазон принимаемых значений от -32768 до 32767.

Регистр состояний PS в моем проекте является двухбитным. Его особенностью является то, что каждый его бит имеет самостоятельное значение и может использоваться процессором отдельно от других. Используемые биты регистра обозначаются N и Z. Управляющие признаки N (Negative) характеризует наличие знака минус у результата операций: если число отрицательое, то N=1, в случае неотрицательного значения N=0. Бит Z (Zero) говорит о равенстве или неравенстве нулю результата: в первом случае Z=1, иначе Z=0. Значение этих битов зависит от результата операции сравнения и влияет в свою очередь на операцию условного перехода. Регистр состояний управляется только программно и недоступен пользователю.

Третью часть программы составляет *Память.* Эта часть моей программы реализована в отдельном окне, которое можно убрать. Если окно открыто, то оно всегда находится поверх главного окна. Окно Память предназначено для демонстрации работы процессора с памятью ЭВМ. В памяти хранится программа, которая написанная пользователем, представленная в машинных кодах. В моем проекте коды реализованы в десятичной системе счисления. Также в памяти можно хранить данные в тех ячейках, которые не отводятся под программу. При использовании косвенной и автоинкрементной адресаций обращение идет к данным, которые хранятся именно в памяти.

Таковы основные части моей программы. Рассмотрим теперь состав главного меню программы. Меню состоит из трех пунктов: файл, правка, пуск. Рассмотрим каждое из них по отдельности.

Пункт меню Файл позволяет работать с написанной пользователем программой как с отдельным файлом и включает следующие подпункты:

Новый – создает новый файл, очищая редактор и регистры.

Открыть – открывает файл.

Сохранить – сохраняет в файл с расширением pdp и текущим именем.

Сохранить как – сохраняет в файл с расширением pdp и новым именем.

Выход – выход из имитатора.

Пункт меню Правка включает пункты подменю, которые позволяют редактировать текст программы написанной пользователем. В нем имеются следующие подпункты:

Очистить – полностью удаляет содержимое либо только редактора, либо только регистров, либо редактора и регистров одновременно. Имя файла при этом не меняется (в этом отличие от пункта меню Новый).

Вставить – вставляет в редакторе пустую строку перед текущей.

Удалить – удаляет текущую строку в редакторе.

Пункт меню Пуск предназначен для исполнения написанной пользователем программы. Включает следующие подпункты:

Пуск – запускает написанную пользователем программу.

По шагам – позволяет выполнять каждую команду по отдельности.

Стоп – осуществляет прерывание запущенной программы.

Пауза – позволяет сделать паузу при выполнении запущенной программы

Продолжить – продолжает исполнение программы, после нажатия паузы.

Некоторые подпункты этого пункта доступны только при исполнении программы.

Некоторые пункты главного меню вынесены как кнопки. Также вы можете увидеть кнопку Память, при нажатии на которую появляется окно Память.

Итак, вы познакомились со структурой моего имитатора. Теперь можно приступить к написанию программы.

Написание программы

Основная программа, как уже было сказано выше, пишется в редакторе, который по структуре напоминает таблицу. При запуске имитатора редактор пуст и курсор находится в начале первого столбца редактора. Перемещаться по редактору можно с помощью клавиш управления курсора, либо с помощью мыши. Выбор команды осуществляется только, когда курсор находится в первом или четвертом столбце. При нажатии клавиши Enter или нажатии левой кнопки мыши появится список команд, из которого вы можете выбрать необходимую команду. Все команды пишутся в первом столбце редактора. Дальнейшие ваши действия зависят от того, какую команду вы выбрали.

Если вы выбрали двухадресную команду, то во втором и четвертом столбце появятся операнды, которые будут у команды по умолчанию. Если вы хотите задать свои операнды, то необходимо переместится в то место редактора, где находится нужный вам операнд. Перемещение производится с помощью клавиш управления курсором, либо с помощью мыши. Для выбора необходимого вам операнда следует нажать клавишу Enter, либо нажать левую кнопку мыши. При нажатии появляется список доступных операндов, из которого вы выбираете необходимый.

Если вы выбрали одноадресную команду, то операнды появятся только во втором столбце редактора. Выбор нужного вам операнда осуществляется также как для двух адресной команды.

Если вы выбрали команду перехода, то во втором столбце редактора появится количество слов, на которое осуществляется переход (по умолчанию 0). Для того чтобы задать свое количество шагов необходимо переместится в то место редактора, где находится значение перехода, нажать Enter, либо кнопку мыши, и ввести свое значение. При выборе команды "Вызов подпрограммы" в нижней строке появится вторая часть этой команды – "по адресу 0". Во втором столбце этой строки необходимо указать адрес подпрограммы.

При выборе безадресных команд появится только команда в первом столбце.

При заполнении редактора или изменении программы в памяти происходят изменения. В соответствующей ячейке появляется код новой команды. В ячейки, не содержащие программу, заносятся нули.

Когда вы написали все команды нужно приступить к заполнению регистров. Для заполнения регистров необходимо перевести указатель мыши на часть Регистры и нажать левую кнопку мыши, либо нажать клавишу Tab. При заполнении регистров стоит вспомнить, что максимальное число, которое может храниться в регистре, равно 32767, а минимальное -32768. Поэтому если вы попытаетесь ввести число больше максимального или меньше минимального, то цифра разряда, которую вы ввели, будет равна максимальной цифре, которая может находится в данном разряде числа. Аналогично при исполнении программы, если число заносимое в регистр будет меньше минимального или больше максимального, то значение соответствующего регистра будет минимальным, либо максимальным. Также необходимо помнить что, регистр R7 является счетчиком команды, поэтому туда необходимо занести адрес ячейки начала программы. При использовании подпрограммы в регистр R6 необходимо занести адрес начала стека.

В последнюю очередь при написании программы необходимо занести данные в память. Но это необходимо сделать, только если вы используйте в своей программе косвенный или автоинкрементный способ адресации. Для занесения данных в память необходимо активизировать окно Память. Это можно сделать, нажав кнопку Память (если окна Память нет), либо направив курсор мыши на окно Память нажать левую кнопку мыши. Следует помнить, что в те ячейки памяти, которые отведены под программу, данные заносить нельзя.

Таким образом, следуя вышеизложенным рекомендациям, вы сможете написать свою программу. Советую написанную программу сохранить, используя пункты меню Сохранить и Сохранить как.

Исполнение программы

Теперь рассмотрим, как написанная вами программа исполняется. Здесь я затрону только общие моменты исполнения вашей программы. Конкретно исполнение каждой команды можно посмотреть в разделе описания.

Исполнение любой программы начинается с нажатия клавиши F9 или пункт меню Пуск. Вначале считывается значение регистра R7 (счетчика команд). Значение R7 увеличивается на 2. Исполняется команда, находящаяся по адресу, считанному из счетчика команд. Эти три операции повторяются пока не встретится команда Стоп или пока не возникнет ошибка исполнения. Исполнение может быть остановлено самим пользователем (кнопка Стоп или пункт меню Стоп). Также пользователь может приостановить исполнение (кнопка Пауза или пункт меню Пауза) и затем продолжить приостановленное исполнение (кнопка Продолжить или пункт меню Продолжить).

Выполнить можно также только одну команду. При нажатии клавиши F8 исполняется команда, находящаяся по адресу указанному в R7, при этом значение счетчика команд увеличивается на 2.

При исполнении вашей программы могут возникнуть следующие ошибки:

*Неверное значение счетчика команд* – значение регистра R7 не попадает в диапазон 1000…2024.

*Пустая строка, не могу выполнить* – нет команды в ячейке, адрес которой указан в R7, либо нет команды Стоп в конце программы.

*Неверный адрес* – адрес, указанный в регистре, при использовании косвенной адресации не попадает в диапазон 1000…2024.

*Данные заносятся в адрес, отведенный под программу* – при использовании косвенной адресации данные заносятся в память по адресу, который используется программой.

*Данные не должны содержать букв* – в памяти хранятся данные, которые содержат посторонние символы (не цифры).

Переполнение диапазона чисел – число, хранимое в памяти очень большое.

*Неверный адрес перехода* – значение адреса перехода не принадлежит диапазону 1000…2024.

*Неверные символы в команде перехода* – адрес перехода в команде перехода содержит посторонние символы (не цифры).

*Нет адреса подпрограммы* – команда Вызов подпрограммы по адресу не содержит адреса начала подпрограммы.

*Неверный адрес подпрограммы* – адрес начала подпрограммы не принадлежит диапазону 1000…2024, либо адрес подпрограммы содержит посторонние символы (не цифры).

*Неверный адрес возврата* – адрес возврата не принадлежит диапазону 1000…2024.

При исполнении могут возникнуть и другие ошибки, которые мне не удалось предусмотреть.

Особое внимание следует уделить исполнению программы с подпрограммой (см. пример программы с подпрограммой). При исполнении команд обращения к подпрограмме и возврата из подпрограммы происходит обращение к стеку, адрес начала которого хранится в регистре R6. Принцип работы стека изложен в первой части курсовой, в разделе способы адресации.

Примеры программ

Простая программа, использующая регистровую адресацию.

Найти сумму всех чисел от А до В.

Число А хранится в R0, число В в R1, сумма в R3. Пусть А=1, В=5.

|  |  |
| --- | --- |
| Регистры | Редактор |
| R0 = +00001 | 1000 Переслать R0 в R2 |
| R1 = +00005 | 1002 Сравнить R2 c R1 |
| R2 = +00000 | 1004 Переход на 3 слов |
| R3 = +00000 | 1006 Добавить R2 к R3 |
| R7 = +01000 | 1008 Увеличить на 1 R2 |
|  | 1010 Переход на -5 слов |
|  | 1012 Стоп |

В данной программе не используются регистры R4, R5, R6.

Программа, использующая косвенную адресацию.

Найти сумму всех чисел от А до В.

Число А хранится в ячейке 1016, число В в ячейке 1018, результат в ячейке 1020. Регистр R0 хранит адрес первого числа, R1 – второго числа, R3 – результата.

|  |  |  |
| --- | --- | --- |
| Регистры | Редактор | Память |
| R0 = +01016 | 1000 Переслать (R0) в R2 | 1016 1 |
| R1 = +01018 | 1002 Сравнить R2 c (R1) | 1018 5 |
| R2 = +00000 | 1004 Переход на 3 слов | 1020 0 |
| R3 = +01020 | 1006 Добавить R2 к (R3) |  |
| R7 = +01000 | 1008 Увеличить на 1 R2 |  |
|  | 1010 Переход на -5 слов |  |
|  | 1012 Стоп |  |

Программа работы с массивом данных.

Посчитать число положительных элементов массива и занести его в R5.

В R0 хранится адрес начала массива 1030. В R1 – адрес конца массива 1036.

|  |  |  |
| --- | --- | --- |
| Регистры | Редактор | Память |
| R0 = +01016 | 1000 Сравнить (R0)+ c R3 | 1030 -1 |
| R1 = +01018 | 1002 Если < переход на 1 слов | 1032 2 |
| R5 = +00000 | 1004 Увеличить на 1 R5 | 1034 3 |
| R7 = +01000 | 1006 Сравнить R0 с R1 | 1036 -4 |
|  | 1008 Если <= переход на -5 слов |  |
|  | 1010 Cтоп |  |

Программа с подпрограммой.

Даны два массива одинаковой размерности. Произвести обмен максимальными элементами. В качестве подпрограммы оформить нахождение максимального элемента массива.

Адрес начала первого массива 1050 хранится в R0. Адрес конца первого массива 1060 в R1. Адрес начала второго массива 1070 в R2. Адрес конца второго массива 1080 в R3. Начало стека 1200 в R6.

|  |  |  |
| --- | --- | --- |
| Регистры | Редактор | Память |
| R0 = +01050 | 1000 Вызов подпрограммы | 1050 1 |
| R1 = +01060 | 1002 по адресу 1022 | 1052 2 |
| R2 = +01070 | 1004 Переслать R2 в R0 | 1054 3 |
| R3 = +01080 | 1006 Переслать R3 в R1 | 1056 -2 |
| R6 = +01200 | 1008 Переслать R4 в R2 | 1058 4 |
| R7 = +01000 | 1010 Переслать R5 в R3 | 1060 1 |
|  | 1012 Вызов подпрограммы | … |
|  | 1014 по адресу 1022 | 1070 -2 |
|  | 1016 Переслать R2 в (R5) | 1072 5 |
|  | 1018 Переслать R4 в (R3) | 1074 6 |
|  | 1020 Стоп | 1076 -3 |
|  | 1022 Очистить R4 | 1078 7 |
|  | 1024 Переслать R0 в R5 | 1080 -7 |
|  | 1026 Переслать (R0)+ в R4 |  |
|  | 1028 Сравнить (R0) с R4 |  |
|  | 1030 Если < переход на 2 слов |  |
|  | 1032 Переслать R0 в R5 |  |
|  | 1034 Переслать (R0) в R4 |  |
|  | 1036 Увеличить на 1 R0 |  |
|  | 1038 Увеличить на 1 R0 |  |
|  | 1040 Сравнить R0 с R1 |  |
|  | 1042 Если <= переход на -8 слов |  |
|  | 1044 Возврат |  |

В данной программе регистры R4 и R5 используются как вспомогательные.

Заключение.

Разработанная программа-имитатор позволяет наглядно иллюстрировать логическую структуру процессора и показать как он исполняет программы написанные пользователем. Фактически мой имитатор во многом повторяет написанную для ПЭВМ "Yamaha" программу-имитатор PDP.

Программа предназначена для изучения основных принципов устройства ЭВМ, а именно, устройства микропроцессора как основной части практически любой ЭВМ. Данный имитатор позволяет пользователю написать свою программу, используя данную систему команд, и проследить, как процессор выполняет заданную программу. В систему команд имитатора были отобраны наиболее важные инструкции, предназначенные для обработки целых чисел.

Как уже было сказано данная программа написана в среде Delphi и хранится на диске в файле pdp.exe. Примеры предложенные в тексте курсовой работы вы также сможете найти на диске в файлах test1.pdp, test2.pdp, test3.pdp, test4.pdp. Программа предназначена для работы на компьтерах PC AT 486 и выше. На компьютере также необходимо наличие операционной системы WINDOWS'95/98. Рекомендуется также минимальное разрешение экрана 800 на 600.