ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Вычислительная техника»

МОДЕЛИРОВАНИЕ

Методические указания

Самара
Самарский государственный технический университет
2010

Печатается по решению редакционно-издательского совета СамГТУ

УДК 519.213, 519.216, 519.22, 004.421

Моделирование: метод. указ. / сост. Б.*В. Мартемьянов*. – Самара: Самар. гос. техн. ун-т, 2010. – 18 с.: ил.

Изложено содержание заданий на курсовую работу по дисциплине «Моделирование». Приведены основные требования к содержанию и оформлению пояснительной записки и исходных текстов программ. Рассмотрены методика выполнения и пример организации интерфейса среды моделирования.

Предназначены для студентов специальности 230101 «Вычислительные машины, комплексы, системы и сети» всех форм обучения.

УДК 519.213, 519.216, 519.22, 004.421

Рецензент д-р техн. наук П.К. Кузнецов

ВВЕДЕНИЕ

Приводимое ниже задание по курсовому проекту соответствует рабочей программе по курсу «Моделирование» для студентов специальности 230101.

Цель задания на курсовой проект – дать студентам навыки по самостоятельному проектированию программ имитационного моделирования компонентов и систем цифровой вычислительной техники.

Выбранная тема проекта позволяет закрепить знания, полученные студентами при изучении данной дисциплины («Моделирование»), а также курсов «Программирование», «Схемотехника», «Теория цифровых автоматов», «Теория проектирования ЭВМ». Таким образом, настоящий курсовой проект оказывается связующим звеном при освоении нескольких профилирующих учебных дисциплин специальности «Вычислительные машины, комплексы, системы и сети».

Задания предполагают достаточно свободное владение студентами одним из языков программирования высокого уровня и соответствующей средой разработки и отладки программ, знания арифметических и логических основ цифровой техники, а также перечисленных выше профилирующих дисциплин.

1. ТЕМА КУРСОВОЙ РАБОТЫ И ИСХОДНЫЕ ДАННЫЕ

1.1. ТЕМА И ЦЕЛЬ КУРСОВОЙ РАБОТЫ

Тема курсовой работы – «Разработка среды для имитационного моделирования операционного устройства».

Цель курсовой работы – формирование практических навыков разработки имитационных моделей компонент и систем вычислительной техники с использованием инструментальных средств языка программирования высокого уровня.

В курсовой работе необходимо разработать интерактивную среду для моделирования поведения конкретного операционного устройства (ОУ), определяемого исходной микропрограммой выполнения арифметической операции, заданной в виде ГСА.

В вариантах заданий варьируются исходные ГСА, задающие микропрограммы выполнения арифметических операций, поэтому при общем шаблоне задания на курсовую работу получаются различные по содержанию программные реализации моделей.

1.2. ИСХОДНЫЕ ДАННЫЕ

Исходными данными к курсовой работе являются:

– индивидуальный вариант микропрограммы выполнения арифметической операции, выдаваемый студентам в виде ГСА (пример на рис. 1.1);

– требования к реализуемым уровням и режимам моделирования объекта;

– требования к обязательным возможностям интерфейсных средств среды моделирования;

– требования к оформлению пояснительной записки и особенностям программной реализации.

Операционное устройство считается состоящим из управляющего автомата (УА) и операционного автомата (ОА).

Структура ОА не проектируется. Основу модели ОА составляет набор процедур, моделирующих выполнения микроопераций над кодами данных и вычисления логических функций, задающих выходные сигналы ОА о состояниях кодов данных.

Форматы разрядных сеток для представления данных определены в каждой ГСА. Для представления этих данных в модели студент должен выбрать наиболее подходящие типы данных используемого языка программирования.

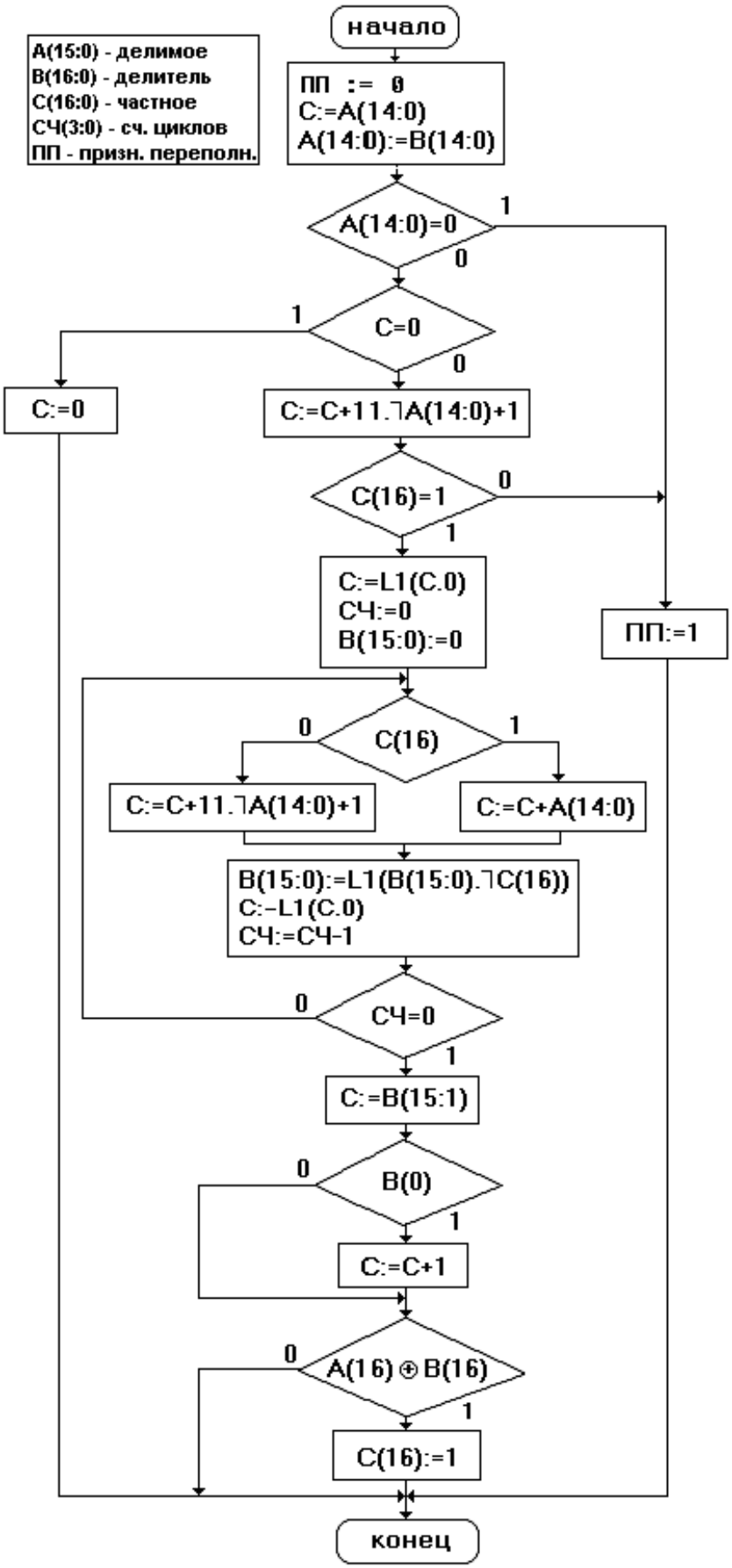

Рис. 1.1. Пример микропрограммы выполнения арифметической операции

В интерактивной среде моделирования реализуются два уровня моделирования ОУ:

– моделирование на уровне микропрограммы;

– моделирование на уровне взаимодействия УА и ОА;

Моделирование ОУ на уровне микропрограммы. Ветвления в алгоритме программируются средствами условного оператора if. Модель отдельной микрокоманды представляет собой последовательный вызов соответствующих процедур, моделирующих выполнения микроопераций. Эта последовательность закладывается в разрабатываемый программный код модели.

В результате такого моделирования ОУ можно проверить правильность микропрограммы выполнения арифметической операции.

Моделирование ОУ на уровне взаимодействия УА и ОА. Модель УА строится на основании результатов проектирования УА как автомата Мили на жесткой логике.

Модель УА формирует вектор Y управляющих сигналов, детализирующий команды для ОА до уровня микроопераций. Модель ОА в результате анализа состояния вектора Y вызывает процедуры выполнения указанных вектором микроопераций и затем вычисляет вектор X значений выходных сигналов, поступающих на вход УА.

В модели этого режима запрещается закладывать в программу какую-либо последовательность вызовов процедур моделирования микроопераций. Каждая такая процедура может быть вызвана только как следствие обнаружения конкретного (например единичного) состояния определенной компоненты вектора Y.

Модель УА должна выделять следующие его компоненты:

– память состояний (ПС) УА на D триггерах;

– дешифратор кодов (ДК) состояний УА (это необязательный структурный компонент УА);

– комбинационные схемы (КС), формирующие вектор выходных сигналов Y, и вектор D сигналов управления состояниями разрядов ПС УА;

– память на D триггерах для запоминания некоторых компонент вектора логических условий.

Модели КС строятся на функциональном уровне, т.е. как последовательность булевых выражений, вычисляющих компоненты векторов Y и D.

В интерактивной среде на каждом из уровней моделирования реализуются два режима моделирования ОУ:

– пошаговый режим выполнения микропрограммы;

– автоматический режим выполнения микропрограммы.

В первом из режимов предполагается наличие на форме кнопки с надписью типа «Такт». Каждое нажатие такой кнопки должно вызывать продвижение по микропрограмме на 1 такт. Причем этот такт должен соответствовать понятию такта работы ОУ при проектировании УА на основе модели цифрового управляющего автомата типа Мили.

В автоматическом режиме предполагается наличие на форме кнопки с надписью типа «Выполнить МП». При нажатии на эту кнопку микропрограмма выполняется с начала до конца без вмешательства со стороны пользователя, т.е. автоматически.

Интерфейсные средства среды моделирования должны позволять:

– наблюдать на экране ГСА, размеченную состояниями автомата модели Мили;

– выбирать любой из описанных выше уровней и режимов моделирования:

– задавать начальные значения операндов операции;

– просматривать процесс выполнения микропрограммы по шагам (по тактам) с отображением на форме всех используемых в микропрограмме кодов;

– выполнять микропрограмму автоматически;

– отображать все изменения кодов переменных, участвующих в микропрограмме;

– при моделировании на уровне взаимодействия УА и ОА отображать на форме состояния векторов, задающих все входные, выходные и внутренние векторы УА в предположении, что УА спроектирован как автомат модели Мили на жесткой логике с элементами памяти

на D триггерах, а также отображать на ГСА графическую метку, задающую текущее состояние УА.

При разработке только модели уровня моделирования микропрограммы максимальная оценка – «***удовлетворительно***». Для ее получения ***эту часть задания необходимо выполнить и защитить на «отлично»***.

2. ОБЩИЕ ТРЕБОВАНИЯ К КУРСОВОЙ РАБОТЕ И ЕЕ ОФОРМЛЕНИЮ

2.1. ЯЗЫК ПРОГРАММИРОВАНИЯ И ГРАФИЧЕСКИЙ РЕЖИМ

Программная часть курсовой работы выполняется с использованием среды ***Delphi-6*** (или более поздних версий) или среды какого-либо иного языка программирования высокого уровня, предназначенной для создания Windows приложений. Использованный графический режим видеосистемы должен быть не хуже, чем TrueColor с пространственным разрешением 1024*768 пикселей.

2.2. ОТЧЕТНЫЕ МАТЕРИАЛЫ ПО КУРСОВОМУ ПРОЕКТУ

В качестве результата выполнения курсового проекта *представляются*:

– какой-либо носитель информации;

– пояснительная записка.

Носитель информации должен содержать:

– исходные тексты всех разработанных программных единиц;

– комплект файлов, достаточный для запуска программы из соответствующей операционной системы (ОС) и среды разработки;

– «бумажную» и электронную версии пояснительной записки, выполненную в текстовом процессоре Word версии 2000 или выше с соблюдением требований ГОСТов на оформление технических текстов;

– файл readme с информацией об авторе, языке программирования и об ОС.

Пояснительная записка должна:

– отображать процесс проектирования программного продукта;

– содержать описание программной реализации;

– содержать инструкции пользователю с описанием интерфейсных средств.

2.3. СТРУКТУРА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Пояснительная записка должна иметь следующую структуру.

1. Титульный лист (рис. 2.1).

Федеральное агентство по образованию Российской Федерации
Государственное образовательное учреждение высшего профессионального образования
«Самарский государственный технический университет»

Кафедра ***"Вычислительная техника"***

КУРСОВАЯ РАБОТА
по дисциплине **"Моделирование"**
на тему **" Разработка среды для имитационного моделирования операционного устройства** ".

Выполнил(а) _____
Проверил _____
Оценка _____

Самара 2010

Рис. 2.1. Форма титульного листа записки

2. Задание, содержащее как исходные данные, так и требования, изложенные в данных методических указаниях.

3. Авторскую оценку соответствия качества проекта объявленным требованиям.

4. Математическую постановку задачи:

– описание структур разрядных сеток, используемых для представления всех исходных, результирующих и вспомогательных кодов

с указанием типа кодов (прямой, обратный, дополнительный, модифицированный);

– описание особенностей алгоритма, заданного в виде ГСА. Пример такого описания приведен ниже;

– привлекаемые для решения задачи методы (например методы теории цифровых автоматов; методы моделирования цифровых схем с использованием двоичного (или иного) алфавита моделирования; теорию и методы булевой алгебры, автоматный подход к проектированию программ и т.д.), ссылки на основные алгоритмы, описанные в последующих пунктах пояснительной записки.

5. Описание структурной схемы УА с объяснением назначений ее компонент и логическим содержанием всех кодов. Поскольку УА должен быть автоматом модели Мили, то в общем случае в структурной схеме УА должна быть память (регистр) для хранения значений логических условий. Необходимо объяснить назначение этой памяти и аргументировать необходимость хранения каждого из записываемых в нее логических условий. При отсутствии такой необходимости значение логического условия не следует хранить в такой памяти.

6. Описание типов моделей компонент УА и ОА.

7. Описание процесса проектирования. Описание процесса проектирования моделей компонент УА и ОА должно содержать таблицы переходов УА, описание кодирования микроопераций и логических условий, описание разметки состояний УА на ГСА, все необходимые логические выражения.

Временные диаграммы, иллюстрирующие процессы распространения сигналов в схемах УА и ОА.

Описание содержания модели одного такта работы ОУ с указанием последовательности обращений к моделям компонент ОУ.

8. Описание программной реализации. Описание программной реализации должно содержать:

– требования к операционной системе (Windows 2000, …);

– требования к составу и характеристикам оборудования (например: IBM PC совместимый компьютер класса Pentium 2 или

выше с видеосистемой, поддерживающей графический режим True Color с пространственным разрешением на хуже, чем 1024×768 пикселей с видеопамятью объемом не менее 32 Мбайт и т.п.);

– указание использованного языка программирования;

– описание комплекта файлов, необходимых для запуска программы (*.exe файлы, драйверы и др.) и структуры размещения файлов по папкам;

– описание структуры программы в виде состава модулей и процедур с кратким описанием их назначения, состава параметров, взаимодействия;

– описание структуры данных;

– описание основных алгоритмов с привязкой к описанным ранее методам, модулям, процедурам и структурам данных;

– листинг основных структур данных и подпрограмм: тексты программ в записке лучше набирать шрифтом типа Courier.

9. Описание пользовательского интерфейса и инструкция по инсталляции и запуску программы.

10. Библиографический список.

11. Содержание записки.

2.4. ОСНОВНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОЙ РЕАЛИЗАЦИИ

1. Необходимо полностью учесть содержание задания.
2. Выбрать среду реализации в соответствии с п. 2.1.
3. Программа не должна завершаться аварийно при любых действиях пользователя и при любых исходных данных.
4. Использовать модульный принцип программирования.
5. Текст программы должен содержать комментарии к используемым модулям, процедурам, структурам данных, основным переменным и шагам алгоритма. Строки текста писать с отступом от начала строки, подчеркивая подчиненность процедурам, функциям, операторам цикла и операторным скобкам.

6. Модели микропрограммы во всех случаях должны давать правильный результат арифметической операции. Если в исходной ГСА имеются какие-либо логические ошибки, то обнаружить их и исправить – задача автора курсовой работы.

3. МЕТОДИКА ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Для сокращения последующего текста введем некоторые условные сокращения терминов, касающихся микропрограмм:

МП – микропрограмма;

МК – микрокоманда;

МО – микрооперация.

При выполнении курсовой работы рекомендуется придерживаться следующей последовательности действий.

1. Изучить (вспомнить) теоретический материал, изложенный в [2, 3]. При этом можно ограничиться кругом вопросов, связанных с выбранным вариантом арифметической операции.

2. В результате внимательного изучения заданного варианта МП выявить основные признаки реализованного в МП алгоритма и описать их. Например, на рис. 3.1 показана МП выполнения операции деления. Перечислим основные признаки этой МП.

2.1. МП вычисляет частное $C := A / B$.

2.2. В МП предусмотрен исход «Переполнение разрядной сетки». Переполнение фиксируется, если $|A| \geq |B|$. Это означает, что частное должно быть по модулю строго меньше 1. Поскольку вычисленное частное в следующей команде программы может оказаться делимым или делителем, то естественно принять, что операнды A и B также должны быть по модулю строго меньше 1. Последнее замечание определяет структуру разрядной сетки для представления операндов и результата операции.

2.3. В МП предусмотрен ускоренный вариант получения частного в случае, когда делимое равно 0.

2.4. В МП операнды предполагаются представленными в прямых кодах.

2.5. В МП количество повторений цикла всегда на 1 больше количества вычисляемых разрядов частного, поэтому при завершении циклической части выполняются МО по округлению вычисленного частного.

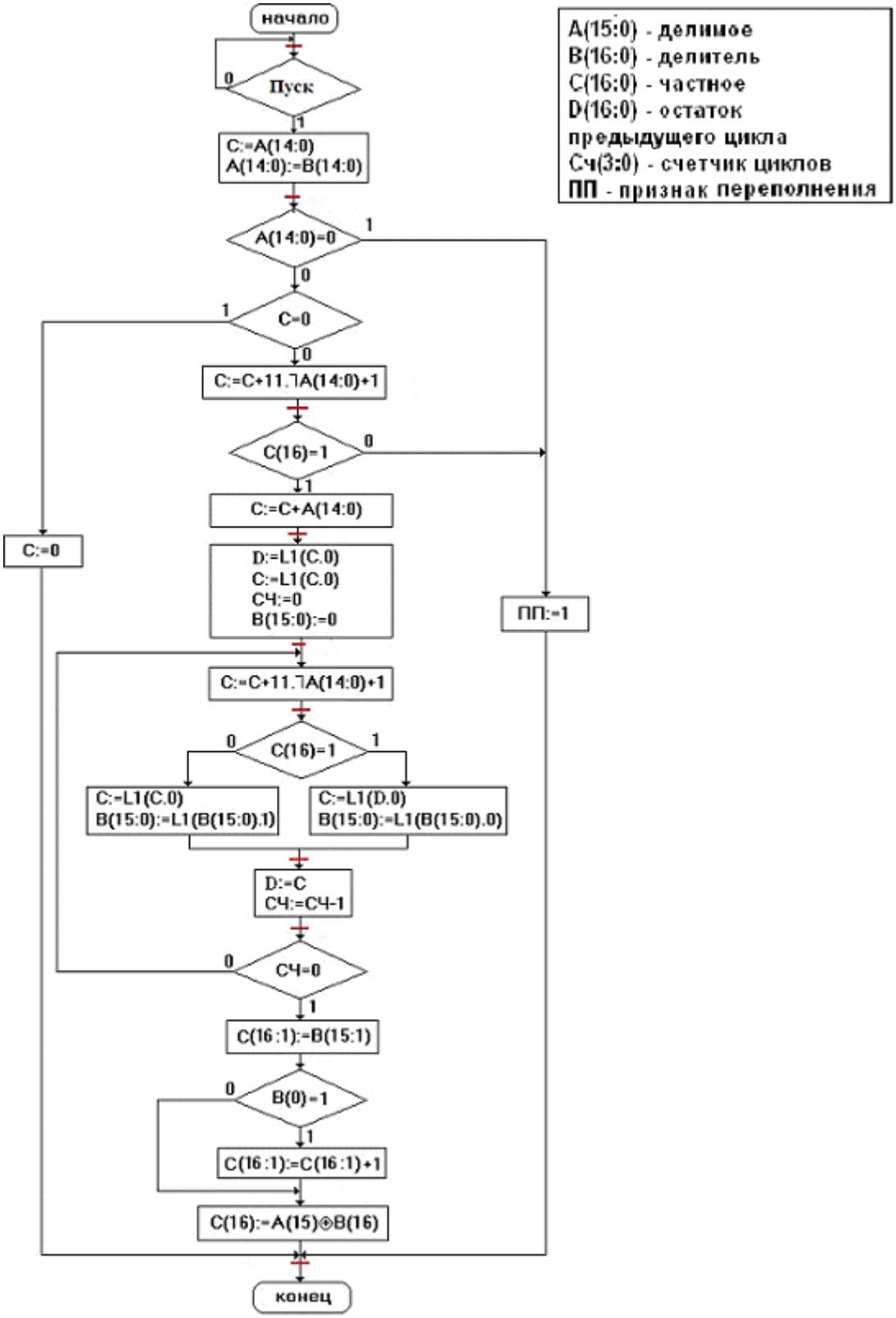

Рис. 3.1. Пример МП
выполнения арифметической операции

2.7. В МП реализован алгоритм деления с восстановлением остатка. Но восстановление остатка выполняется не за счет МО вида *A*:= *A* + *B* («Остаток»:= «Остаток» + «Делитель»), а за счет запоминания текущего значения остатка во вспомогательной переменной *D* и использования на следующем цикле в МО сдвига кода остатка либо только что вычисленного нового остатка, хранящегося в переменной *C*, либо предыдущего остатка, запомненного в переменной *D*.

2.8. В МП использован алгоритм деления со сдвигом кода остатка влево.

Выявленные особенности реализованного алгоритма дают более адекватное представление о микропрограмме и позволяют организовать процесс разработки моделей в правильном направлении.

3. Выполнить разметку состояний на ГСА, соответствующую состояниям управляющего автомата модели Мили. Этот этап выполнения работы требует привлечения знаний, полученных при изучении курса «Прикладная теория автоматов». Размеченные состояния автомата необходимы как для проектирования модели УА, так и для организации режима пошагового выполнения МП.

4. Разработать ***концептуальную модель*** проекта. На уровне концептуальной модели ***проектные решения*** должны носить общий характер. Например:

– позволять или не позволять пользователю вводить исходную строку с помощью клавиатуры;

– в какой форме указывать пользователю очередной шаг алгоритма;

– если программа указывает конкретное состояние на ГСА, то отображаемые на форме значения всех кодов равны их значению до выполнения МО, инициируемых данным состоянием, или после выполнения этих МО;

– отображать или не отображать десятичные эквиваленты кодов;

– в структуре УА использовать или не использовать дешифратор кодов состояний;

– предусматривать модульную структуру или ограничиться одним модулем;

– интерфейсные средства, соответствующие двум уровням моделирования МП, представить на разных закладках или все расположить на одной (рис. 3.2) и т.п.

4. На основе проектных решений концептуального характера разработать интерфейсные средства программы. Для отображения кодов рекомендуется использовать компоненты типа String Grid.

5. Разработать модели всех МО. При этом в качестве моделей обрабатываемых кодов использовать только беззнаковые целочисленные типы данных языка программирования.

При разработке моделей МО следует учитывать, что МО выполняются соответствующими операционными элементами, поэтому подход к содержанию МО с точки зрения программиста часто может привести к неверному ее толкованию, к ошибкам в моделировании. Так, например, в МП (см. рис. 3.1) имеется МО «Сч:= Сч – 1», которая первый раз выполняется при значении Сч=0. С позиций программиста после вычитания 1 из 0 получается число -1, но операционный элемент типа «счетчик» из состояния 0 при вычитании 1 перейдет в состояние, соответствующее самому большому числу, которое может отобразить этот счетчик. В приведенной МП счетчик имеет 4 разряда, поэтому из состояния 0 при вычитании 1 он перейдет в состояние 15. В качестве модели этого счетчика следует взять переменную типа byte, поэтому указанный переход счетчика должен привести к установке этой переменной в состояние, задаваемое следующим двоичным кодом: 00001111.

6. Имея модели МО и размеченную ГСА, можно приступать к программированию модели МП.

7. Выполнить знакомые по курсу «Прикладная теория цифровых автоматов» шаги проектирования УА, ограничившись получением логических выражений, описывающих функционирование регистров и комбинационных схем, входящих в состав УА. Раскрывать структуру этих схем не требуется. Таким образом, компоненты УА моделируются на уровне функциональных моделей.

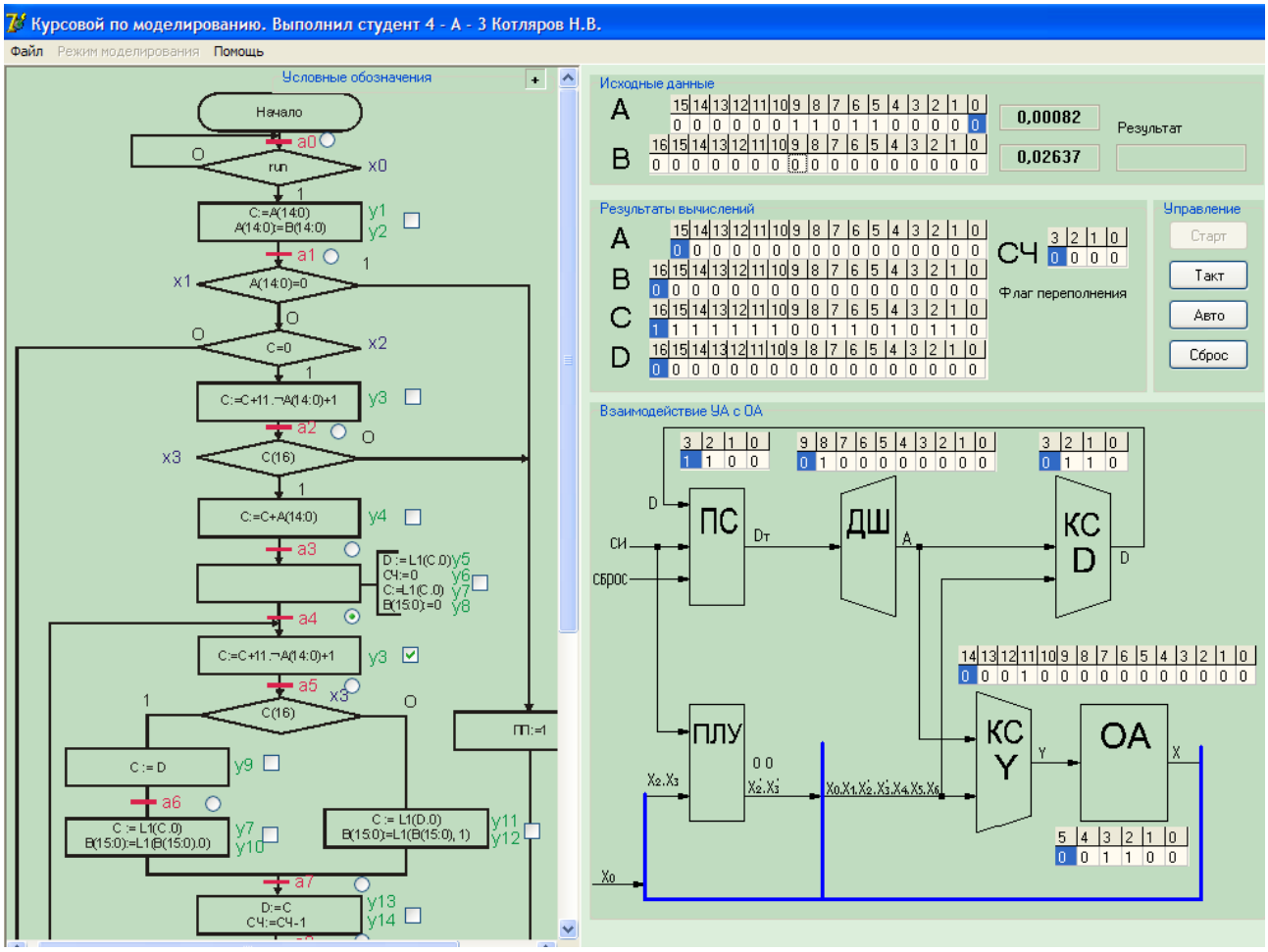

Рис. 3.2. Пример интерфейса среды моделирования МП

8. Понимая последовательность прохождения сигналов через схемы УА и моменты востребованности кодов, формируемых различными компонентами УА, разработать последовательность обращений к моделям компонент УА и ОА. Выполняя этот этап проектирования, студент должен продемонстрировать свое понимание изученных курсов «Схемотехника», «Прикладная теория цифровых автоматов», «Теория проектирования ЭВМ». Готовый «рецепт» здесь не дается. Неправильно спроектированная последовательность вызова моделей компонент может сделать бессмысленным наличие в структуре УА памяти логических условий. Бессмысленной потому, что при ее отсутствии в структуре УА поведение модели УА будет точно таким же, как при наличии этой памяти в составе УА. Поскольку такая память может оказаться необходимой по объективным причинам, то безразличие модели к ее присутствию либо отсутствию в структуре УА говорит о недостаточной адекватности модели моделируемому объекту.

9. При моделировании ОУ на уровне МП модель отдельной МК, содержащей несколько МО, может представлять собой заданную программистом последовательность вызовов моделей конкретных МО. При моделировании на уровне отображения взаимодействия УА и ОА модель УА формирует вектор *Y* выходных сигналов. Модель ОА должна выполнять анализ содержимого этого вектора, просматривая значения его компонент. При обнаружении компоненты, равной 1, модель ОА должна инициировать обращение к модели соответствующей МО. При разработке этих моделей следует принять меры к тому, чтобы конечный результат исполнения модели МК не зависел от относительных значений номеров, присвоенных отдельным МО при выполнении их кодирования. Иными словами, любые изменения последовательности вызовов моделей МО, составляющих данную МК, не должны как-либо влиять на конечный результат исполнения МК. В реальном ОА все МО данной МК выполняются одновременно различными операционными элементами в составе ОА, поэтому результат исполнения любой из таких МО не может быть использован на этом же такте другими МО данной МК. В модели имеет место совершенно иная ситуация. Процессы, протекающие одновременно, любая программа на однопроцессорной вычислительной системе может моделировать только последовательно во времени. В результате в моделях может проявиться зависимость результатов моделирования МО в составе МК от последовательности обращений к моделям отдельных МО в составе МК. Это возможно, например, в МК, состоящей из следующих МО (пример из МП на рис. 3.2):

C[14..0]:= *A*[14..0];

A[14..0]:= *B*[14..0].

При непродуманной модели МК изменение последовательности вызова моделей приведенных МО может привести к тому, что получатся следующие значения кодов: *C*[14..0] = *A*[14..0] = *B*[14..0]. Ошибочность подобного совпадения трех кодов станет очевидной, если учтем, что в микрооперациях в правой части записаны коды до выполнения МК, а в левой части – коды после выполнения МК. Поэтому

содержание МО более точно передают несколько иные их описания. Обозначим индексами t коды до выполнения МК, а индексами t+1 – коды после выполнения МК. Тогда приведенные МО будут описаны несколько иначе:

$C_{t+1}[14..0]:= A_t[14..0];$

$A_{t+1}[14..0]:= B_t[14..0].$

Теперь ясно, что в составе этой МК указаны два совершенно разных кода, наполняющих регистр *A*, поэтому код с регистра *B* в принципе не может через регистр *A* попасть на регистр *C*. Эту особенность поведения реального ОА необходимо правильно смоделировать. Решение этой проблемы за счет какой-либо особой нумерации МО (первой из МО в составе МК присвоим меньший номер, чем второй из них, в результате эти две МО будут моделироваться в последовательности, исключающей передачу кода из регистра *B* в регистр *C* через регистр *A*) будет оценено как неудовлетворительное. Повторим еще раз обязательное требование к логике моделирования:

любые изменения номеров, присвоенных отдельным МО на этапе их кодирования, не должны как-либо влиять на результаты работы модели ОА.

10. При разработке моделей МО необходимо заботиться о том, чтобы части кодов, не указанные в правой части МО, не изменяли своего значения в результате выполнения этой МО. Например, МО

```
Am(31:15):=A(14:0)
```

предполагает, что часть `A(14:0)` кода на регистре остается неизменной. Модель этой МО в виде процедуры может иметь следующий вид:

```
procedure Y1; {Am(31:15):= A(14:0)}
 begin
  Am:=((A and $00007FFF) shl 15) or (Am and $00007FFF)
 end;
```

11. Проверить работу разработанной среды при всех значимых вариантах исходных данных.

12. После отладки значимого фрагмента программного кода можно приступать к составлению пояснительной записки в части, отражающей отлаженный фрагмент программы.

4. ГРАФИК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Студент должен заниматься курсовым проектированием в течение всего семестра, не откладывая работу на «потом», поэтому необходимо придерживаться примерно следующего графика выполнения курсовой работы (табл. 4.1).

Таблица 4.1

Примерный процент выполнения КР	Срок выполнения
Программная реализация: – 25-30% выполнения; – 50% выполнения; – 100% выполнения Пояснительная записка: – 25% выполнения; – 50% выполнения; – 100% выполнения.	 8 – 10 недели; 11-12 недели; 15-16 недели 12-13 недели; 13-14 недели; 15-16 недели.
Защиты курсовой работы	16 – 18 недели

Соблюдение графика проектирования контролируется преподавателем-руководителем курсовой работы. Контроль соблюдения графика осуществляется на лабораторной работе, ближайшей к указанному в графике сроку.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кнут, Д.* Искусство программирования для ЭВМ. Т. 1 Получисленные алгоритмы/ *Д. Кнут*; пер. с англ. под ред. *Ю.М. Баяковского*. – М.: Мир, 1978. – 839 с.
2. *Орлов, С.П.* Арифметика ЭВМ и логические основы переключательных функций: учеб. пособ. / *С.П. Орлов, Б.В. Мартемьянов*. – Самара: СамГТУ, 2004 – 197 с.
3. *Орлов, С.П.* Арифметика ЭВМ и логические основы переключательных функций: учеб. пособ. / *С.П. Орлов, Б.В. Мартемьянов*. 3-е изд., испр. и доп. – М.: Машиностроение – 1, 2005. – 256 с.

Моделирование

Составитель *МАРТЕМЬЯНОВ Борис Викторович*

Редактор *С.И. Костерина*
Компьютерная вёрстка *И.О. Миняева*
Выпускающий редактор *Н.В. Беганова*

Подп. в печать 25.02.10.
Формат 60×84 1/16. Бумага офсетная.
Усл. п.л. 1,06. Уч.-изд. л. 1,04.
Тираж 100 экз. Рег.№ 403/09.

Государственное образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»
443100. г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
443100. г. Самара, ул. Молодогвардейская, 244. Корпус №8