Арифметические основы вычислительной техники и элементы микропрограммного управления. Лабораторный практикум

М.М. Шихов

13 января 2020 г.

Оглавление

Введение

В данном пособии описана лабораторная установка, предназначенная для закрепления изученных алгоритмов выполнения арифметических операций.

У обучающегося формируются знания, умения и навыки в следующих областях: системы счисления, форматы представления чисел, методы вычислений.

Для успешного освоения курса студент должен обладать базовыми знаниями основ информатики. Знания, полученные в ходе освоения данного курса необходимы для последующего изучения завершающих обучение профильных дисциплин: математическая логика и теория алгоритмов, программирование, теория автоматов, схемотехника, исследование операций, проектирование ЭВМ, технология программирования, защита информации.

Для углубленного изучения авторы рекомендуют [1, 2]

1 «Микропрограммное управление» — цикл лабораторных работ

Теорию построения вычислительных устройств рассмотрим на примере устройства, позволяющего выполнить тот или иной алгоритм умножения.

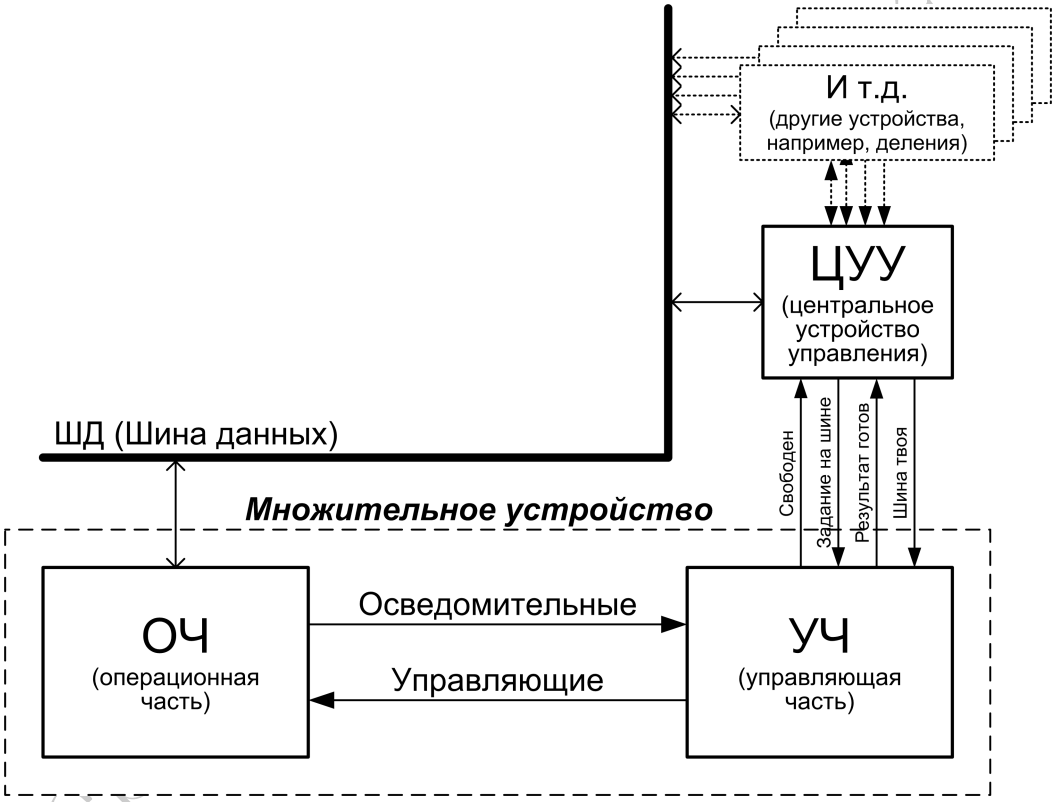

Рис. 1.1: Структура вычислительного устройства

Структурно устройство разделяется на три компонента (см. рисунок 1.1):

- центральное устройство управления (ЦУУ);
- операционная часть множительного устройства (ОЧ);
- управляющая часть множетельного устройства (УЧ).

ЦУУ выполняет *программу* пользователя, состоящую из отдельных *команд*. Одной из команд является команда выполнить умножение. ЦУУ не выполняет эту операцию самсостоятельно, а дает задание множительному устройству. Множительное устройство, состоящее из операционной (ОЧ) и управляющей частей (УЧ), является лишь одним из многих устройств, которым ЦУУ перепоручает выполнить сложые операции.

Операция умножения, например, сводится к последовательности операций сложения, которые являются элементарными. Все остальные сложные операции также сводятся к выполнению элементарных операций по определенному алгоритму. *Элементарная* операция выполняется некоторым элементом оборудования[1] операционной части (ОЧ) сразу, за фиксированный (и относительно короткий) промежуток времени, называемый *тактом*. В составе любого цифрового устройства имеется таймер, отсчитывающий такты[2]. Поэтому, когда говорится «сложная операция», то подразумевается операция, выполняемая за несколько тактов.

При отсутствии сбоев, между центральным управляющим устройством (ЦУУ) и управляющей частью множительного устройства (УЧ) происходит следующий обмен сигналами и данными:

1. **УЧ→ЦУУ:«Свободен»**.

   УЧ выдает сигнал в ЦУУ «Свободен» всегда, когда не выполняет задачу. Отсутствие сигнала «Свободен» от УЧ трактуется ЦУУ как признак того, что множительное устройство занято решением ранее поставленной задачи.

2. **ЦУУ→УЧ:«Задание на шине», ШД=«Операнды»**.

   Убедившись, что множительное устройство готово принять задание (получив от УЧ сигнал «Свободен»), ЦУУ выдает в УЧ сигнал «Задание на шине». Передача задания по шине в общем случае может занять один или несколько тактов. Это зависит от принятых соглашений на формат передачи задания — в общем случае это серия передач по шине данных. Получив сигнал «Задание на шине», УЧ должна в следующем такте снять сигнал «Свободен», и одновременно начать считывать с шины первый фрагмент задания.

3. **УЧ→ЦУУ:«Результат готов»**.

   УЧ с помощью ОЧ выполняет умножение. Этот процесс может длиться довольно долго, и в это время ЦУУ может декодировать следующую команду и, если это возможно, начать её выполнять параллельно с работой

[1] Например, сложение выполняется логической схемой — сумматором. См. описание этого элемента в разделе 1.1

[2] Отсюда термин *тактовая частота*: $1/T$, где T — продолжительность такта

мнжительного устройства. Так как при этом по шине данных (ШД) могут передаваться данные, получив результат, УЧ не должно выдавать его на шину сразу. УЧ выдает в ЦУУ осведомительный сигнал «Результат готов» и держит его до тех пор, пока ЦУУ не освободит шину.

4. **ЦУУ→УЧ:«Шина твоя»**.

   ЦУУ, получив от УЧ сигнал «Результат готов»: завершает (если шина занята) текущую передачу данных по шине; выдает сигнал в УЧ «Шина твоя» и снимает его в следующем такте.

5. **УЧ→ЦУУ:ШД=«Результат»**.

   Получив от ЦУУ сигнал «Шина твоя», УЧ следующем такте должна:

   - снять сигнал «Результат готов»;
   - выдать на шину первый фрагмент результата, и если необходимо, продолжить в следующих тактах выдачу остальных фрагментов результата.

УЧ множительного устройства, выполняя алгоритм умножения, в каждом такте выдает в ОЧ необходимые управляющие сигналы. Управляющие сигналы формируются УЧ в зависимости от своего внутреннего состояния и осведомительных сигналов, часть которых поступает из ОЧ, а часть из ЦУУ. Аппаратура ОЧ содержит набор необходимых утройств, чтобы выполнить все необходимые элементарные операции.

ОЧ составлена из базовых схемотехнических цифровых элементов: логических элементов, сумматоров, мультиплексоров, триггеров, регистров, операционных усилителей и т.д. Необходимые сведения о работе этих элементов приводятся в разделе 1.1.

1.1 Элементы функциональной схемы

В данном разделе приводится необходимый минимум сведений для изображения и понимания функциональных схем. За исчерпывающей информацией обращайтесь к стандарту [3].

1.1.1 Сигналы, шины, жгуты

Отдельная *сигнальная линия* соответствует проводнику (проводу), который передает логические значения 0 или 1. Сигнальная линия на схеме изображается тонкой линией, соединяющей выход некоторого элемента со входом некоторого элемента.

Множество внешних (интерфейсных) сигнальных линий принято объединять в *шины*. Шина на схеме изображается толстой линией и обязательно именуется: сверху над линией в удобном месте подписывается имя шины (на рисунке 1.2 изображена шина X). В общем случае шин на схеме может быть несколько. В качестве имени шины обычно выбирают заглавную литеру: X, Y, Z, P, а если шины разделены по функциональному назначению, то они могут быть названы: ШД — шина данных, ША — шина адреса, ШУ — шина управления и т.д.

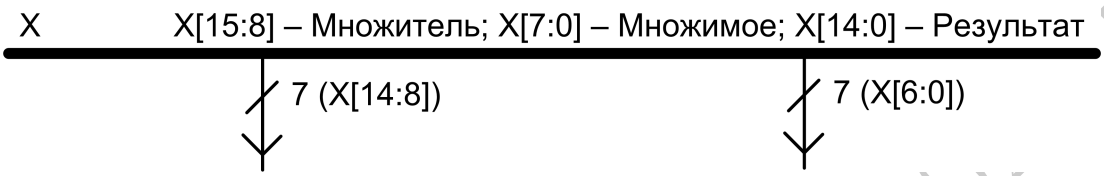

Рис. 1.2: Изображение шин и жгутов

Чтобы не рисовать множество параллельных сигнальных линий, в случае, когда передается группа бит, на схеме изображают *жгут*. Жгут также изображается тонкой линией, на которой в произвольном месте (но не на концах) тонкими линиями рисуется тупоугольная стрелка, указвающая направление передачи, сверху от стрелки жгут перечеркивается косой чертой и справа указывается разрядность жгута. В случае когда требуется указать, какие разряды шины отведены в жгут, в скобках после разрядности перечисляются обозначения групп сигнальных линий в составе шины. На рисунке 1.2 от шины X отведены два жгута по 7 разрядов каждый.

Сигналы разделяют на осведомительные и управляющие. Управляющие сигналы обычно обозначаются: $Y0$, $Y1$, $Y2$,..., а осведомительные — $P0$, $P1$, $P2$,... Если управляющий сигнал подается на вход элемента, а не на управление, то стрелку не рисуют. На рисунке 1.3 изображен 8-и разрядный регистр (подробнее о регистрах см. раздел 1.1.3), который управляется сигналами $Y0$, $Y1$. Седьмой разряд выхода регистра используется как осведомительный сигнал $P0$. На входные разряды с 0 по 3 подаются соответственно сигналы с 3 по 6 шины X. На вход $RG1[7:4]$ подается двоичное значение $(0011)_2$.

Управляющие сигналы подводятся стрелками, входящими в управляемый блок справа. Осведомительные отводятся с выходов стрелкой, направленной вниз или влево.

Входы большинства элементов на схеме подводятся сверху, а выходы — снизу. Исключением являются элементы логической схемы и триггеры (см. рисунок 1.4), в которой входы подаются слева, а выходы снимаются справа. В любом случае недопустимо разворачивать или переворачивать блоки.

Если о разрядности входов можно судить по разрядности выходов, то допустимо подписывать только разрядности выходов элементов.

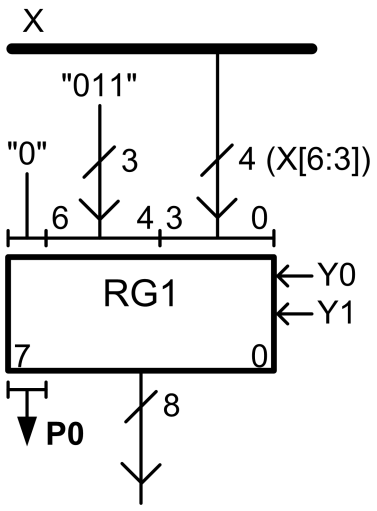

Рис. 1.3: Входы, выходы и управляющие сигналы регистра

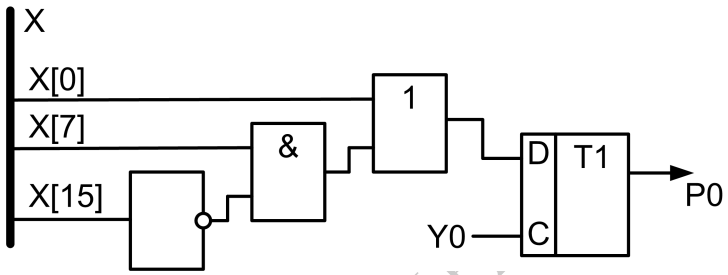

Рис. 1.4: Логическая схема

1.1.2 Элементы логики

Элементы логики отличаются от элементов памяти тем, что представляют собой реализацию однотактной функции: выход формируется за один такт времени и зависит только от информации на входах. У элементов логики нет внутреннего состояния. Типичные элементы логики это: логические схемы (И, ИЛИ, НЕ, XOR, И-НЕ, и т.д.), мультиплексоры, сумматоры, шинные формирователи и т.д.

Многовходовые логические функции

На схемах элементы, реализующие многовходовую ассоциативную и комутативную логическую функцию (И, ИЛИ, XOR), допустимо изображать одним элементом, как на рисунке 1.5.

Выход такого элемента — одноразрядный.

Инвертор

На схемах многоразрядный инвертор изображается как рисунке 1.6.

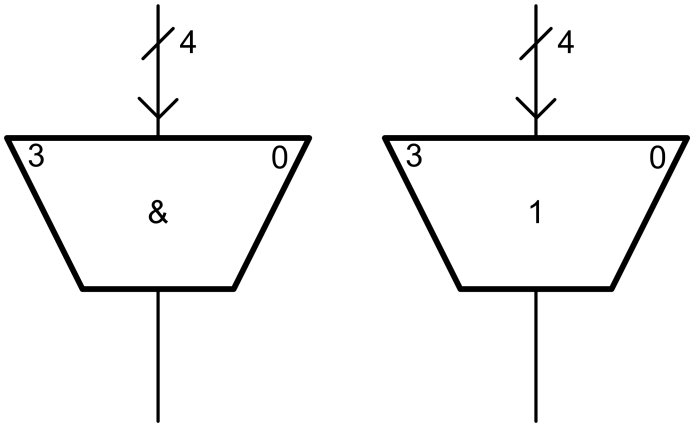

Рис. 1.5: 4-х входовые функции И, ИЛИ

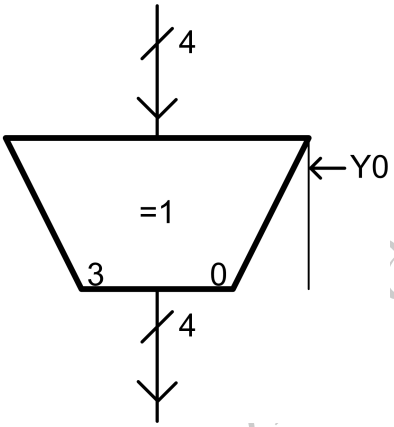

Рис. 1.6: 4-х разрядный инвертор

Если значение единственного управляющего сигнала 0, то разряды входа без изменений подаются на выход, иначе на выходе формируется поразрядная инверсия входа.

Мультиплексор

Условное графическое обозначение мультиплексора приведено на рисунке 1.7.

В зависимости от значения управляющего сиграла (на рисунке 1.7 это $Y0$) на выход подается либо нулевое, либо первое входное плечо.

Мультиплексоры с большим количеством плеч могут быть получены на основе 2-плечевых мультиплексоров каскадным подключением. Мультиплексор с произвольным количеством плеч рисуют упрощенно (см. рисунок 1.8). При этом самый верхний управляющий сигнал соответствует младшему разряду (LSB[3]) двоичного представления номера плеча, а самый нижний - старшему

[3]LSB — Least Significant Bit, Младший бит

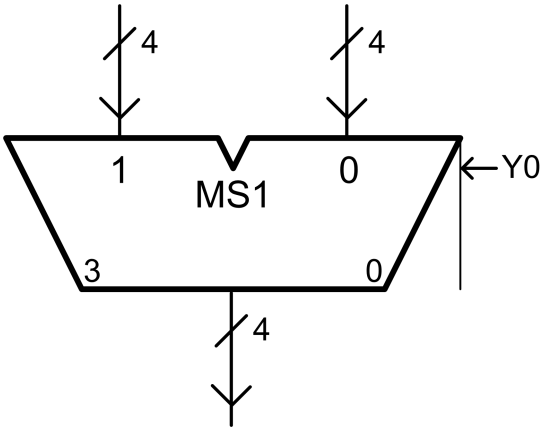

Рис. 1.7: Мультиплексор

(MSB[4]). Например, чтобы выбрать 2-е плечо мультиплексора на рисунке 1.8 нужно подать $Y1 = 1$, $Y0 = 0$.

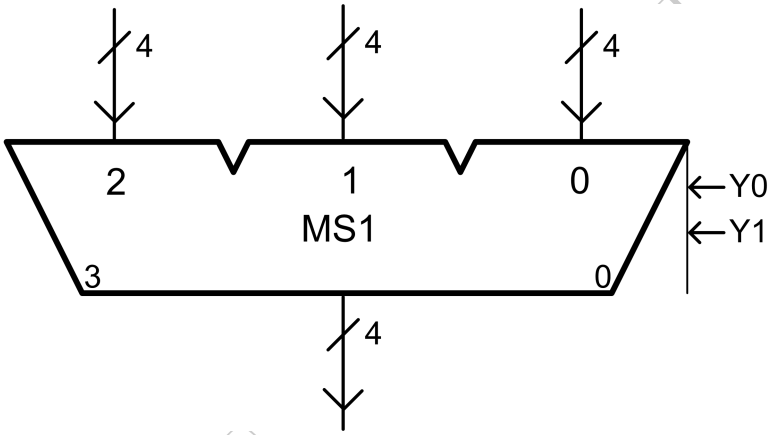

Рис. 1.8: Многоплечевой мультиплексор

Сумматор

Условное графическое обозначение сумматора приведено на рисунке 1.9.

Сумматор выполняет арифметическое сложение n-разрядных входов (на рисунке 1.9 $n = 4$) входов A и B а также одноразрядного входа переноса в младшие разряды c. На выход S выдаются младшие n разрядов результата, а на выход p — бит переноса из старшего $(n-1)$-го разряда суммы S:

$$S \equiv (A + B + c) \pmod{2^n};$$
$$p = (A + B + c) \div 2^n.$$

В случае, если нужно получить сумматор большей разрядности на основе сумматоров меньшей разрядности, то выход переноса из старших разрядов

[4]MSB — Most Significant Bit, Старший бит

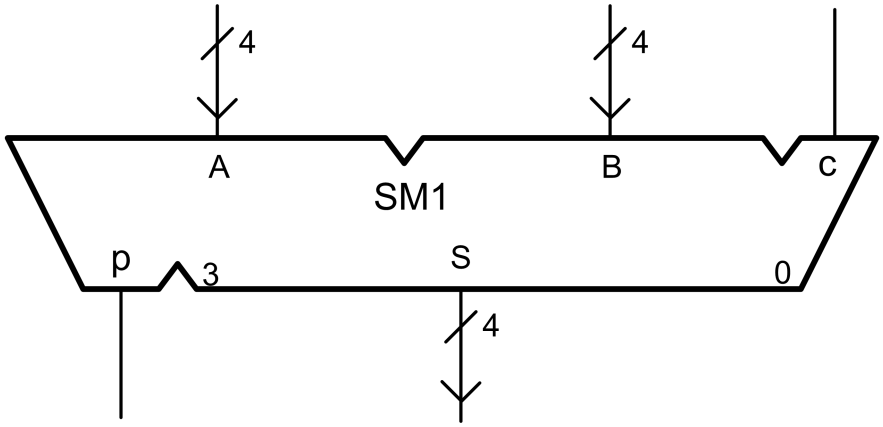

Рис. 1.9: Сумматор

младшего сумматора замыкают на вход переноса старшего сумматора (см. рисунок 1.10).

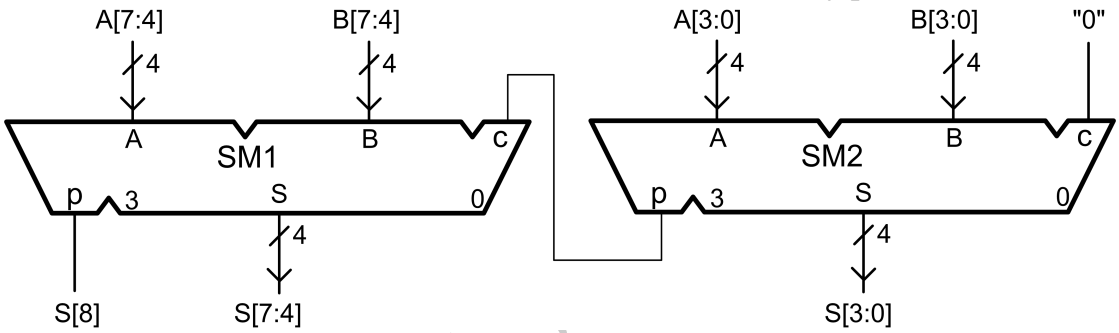

Рис. 1.10: 8-разрядный сумматор на основе двух 4-х разрядных

На схемах сумматор любой разрядности рисуют одним элементом, если неважно, как этот сумматор устроен.

Шинный формирователь

Шинным формирователем устройство называется потому, что обычно его выходы подключены к шине. По управляющему сигналу (на рисунке 1.11 сигнал $Y0$) шинный формирователь может отключать свои выходы от шины. Когда шинный формирователь подключен к шине, он выдает данные со входа на шину. Так как шина может быть общей для нескольких устройств, то использование шинных формирователей обязательно, чтобы гарантировать, что только одно устройство выдает на шину данные.

1.1.3 Элементы памяти

Элементы памяти могут сохранять информацию (состояние) между тактами. Они обычно имеют управляющий сигнал, который управляет запоминани-

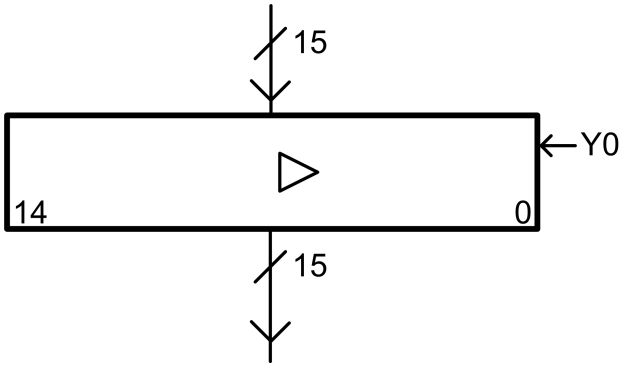

Рис. 1.11: 15-разрядный шинный формирователь

ем информации со входов. Т.е. в течение такта такой элемент либо «запоминает» информацию со входов, либо хранит (или преобразует) ранее запомненную информацию.

Регистры

Обычный регистр содержит автономную n-разрядную ячейку памяти и предназначен только для запоминания n бит информации со входов. Помимо основной функции он может, например, выполнить сброс всех бит своей ячейки памяти в 0 или 1. На выходы регистра всегда выдается содержимое его внутренней ячейки памяти.

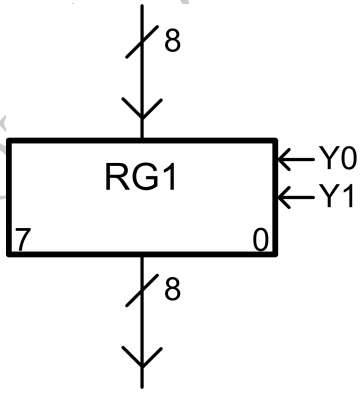

Рис. 1.12: 8-разрядный регистр

Например, на рисунке 1.12 изображен обычный регистр, расшифровка управляющих сигналов которого может быть следующей:

- $Y0$ — разрешение записи в $RG1$. Если данный сигнал подается в течение такта, то регистр «запоминает» информацию со входов и хранит до тех пор, пока $Y0$ вновь не будет подан в будущем.
- $Y1$ — сброс всех разрядов внутренней ячейки памяти $RG1$ в 0.

Управляющие сигналы регистра могут конфликтовать, например, состояние регистра не определено, если одновременно подать сигналы разрешения записи и сброса.

Сдвиговые регистры изображаются особым образом, в виде параллелограмма. Направление сдвига разрядов ячейки памяти обозначается стрелкой внутри регистра. Например, при сдвиге влево значение старшего разряда теряется, а в младший разряд заносится значение разряда со специального входа, который изображается боковой планкой с нужного края.

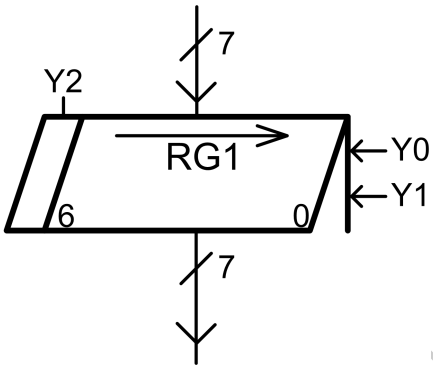

Рис. 1.13: 7-разрядный сдвиговый регистр

На рисунке 1.13 изображен сдвиговый регистр, расшифровка управляющих сигналов которого может быть следующей:

- $Y0$ — разрешение записи в $RG1$. Если данный сигнал подается в течение такта, то регистр «запоминает» информацию со входов и хранит до тех пор, пока $Y0$ вновь не будет подан в будущем.
- $Y1$ — сдвиг $RG1$ вправо.
- вход $Y2$ определяет значение 6-го разряда при сдвиге.

Триггеры

Триггер — это одноразрядное запоминающее устройство. Триггеров существует несколько видов: D, RS, JK, . . . В схемах будет использоваться D-триггер (см. рисунок 1.14), который можно считать одноразрядным регистром: если на вход C подается 1, то триггер «запоминает» бит информации со входа D, а если на вход C подается 0, то триггер выдает ранее запомненный бит.

1.2 Основы микропрограммного управления

Множительное устройство, ЦУУ и другие модули (см. рисунок 1.1) работают автономно и каждый выполняет собственный алгоритм функционирования.

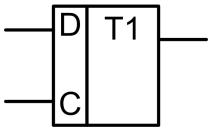

Рис. 1.14: D-триггер

Например, УЧ множительного устройства в бесконечном цикле выполняет алгоритм: принять задания от ЦУУ; выполнить умножение; выдать в ЦУУ результаты. ЦУУ, например, выполняет в бесконечном цикле алгоритм: считать очередную команду программы пользвателя из памяти; декодировать её; выдать необходимые задания исполнительным модулям; дождаться результатов, считать и сохранить их; выбрать следующую команду...

УЧ модуля (в том числе и множительного устройства) представляет собой автомат, по тактам выполняющий алгоритм работы модуля. Существует несколько способов создания таких автоматов. В данном разделе будут рассмотрены математические основы функционирования управляющих автоматов, а затем особый вид таких автоматов — *микропрограммные*.

В математической основе выделяют два типа управляющих автоматов: автоматы Мили и автоматы Мура. Общим для автоматов обоих типов является:

- Внутреннее состояние. Автомат в течение такта находится в одном из множества возможных внутренних состояний. По завершении такта, автомат переключается в новое состояние. Множество внутренних состояний
$$S = \{s_0, \ldots, s_{n-1}\}.$$
- Начальное состояние s_0. С этого внутреннего состояния автомат начинает свою работу.
- Набор значений (вектор) осведомительных сигналов
$$P = (p_{m-1}, \ldots, p_0).$$
Новое внутреннее состояние, в которое перейдет автомат, определяется его текущим внутренним состоянием и значениями осведомительных сигналов, которые поступали ему в течение такта.
- Набор управляющих сигналов
$$Y = (y_{k-1}, \ldots, y_0).$$
Автомат в течение такта формирует и выдает значения управляющих сигналов. В способе формирования управляющих сигналов заключается главное отличие между автоматами Мили и Мура.

Если в i-м такте автомат находится в состоянии $s^{(i)}$, и поступают значения осведомительных сигналов $P^{(i)}$, то состояние $s^{(i+1)}$, в которое он перейдет в следующем такте, определяется функцией переходов $\mathbf{S}$, а набор управляющих сигналов, которые выдаются в i-м такте $Y^{(i)}$ определяются функцией выходов $\mathbf{Y}$.

Для автомата Мили:

$$\begin{aligned} s^{(i+1)} &= \mathbf{S}(s^{(i)}, P^{(i)}); \\ Y^{(i)} &= \mathbf{Y}(s^{(i)}, P^{(i)}). \end{aligned} \tag{1.1}$$

Видно, что для автомата Мили векор управляющих сигналов $Y^{(i)}$ формируется на основе текущего состояния $s^{(i)}$ и вектора осведомительных сигналов $P^{(i)}$.

Для автомата Мура управляющие сигналы зависят только от текущего состояния:

$$\begin{aligned} s^{(i+1)} &= \mathbf{S}(s^{(i)}, P^{(i)}); \\ Y^{(i)} &= \mathbf{Y}(s^{(i)}). \end{aligned} \tag{1.2}$$

Работу автомата удобно изобразить на диаграмме переходов, которая представляет собой граф, вершинам которого соответствуют состояния, а дугам — функция переходов $\mathbf{S}$. Начальное состояние s_0 обычно выделяется двойным контуром. Над дугой подписываются значения осведомительных сигналов, определяющих переход в следующее состояние. Так как управляющие сигналы в автомате Мили зависят от состояния автомата и осведомительных сигналов, то управляющими сигналами взвешиваются дуги. В автомате Мура управляющие сигналы зависят только от состояния, поэтому управляющими сигналами взвешиваются вершины.

Далее приводятся примеры диаграмм переходов автоматов, из раздела 1.2.3, к этому разделу следует обратиться всем желающим понять назначение управляющих и осведомительных сигналов. Пока же — это неважно.

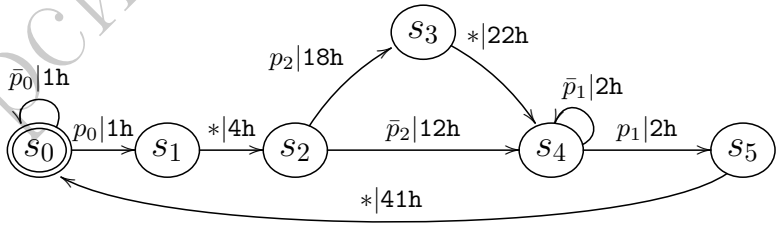

Рис. 1.15: Диаграмма переходов автомата Мили

На рисунке 1.15 приведена диаграмма автомата Мили. Далее приводится описание нескольких переходов.

- Из начального состояния s_0, автомат переходит в состояние s_1, при $p_0 = 1$ (на схеме это обозначено как p_0); при этом выдается управляющий сигнал

y_0 (на схеме вектор управляющих сигналов закодирован в 16-ичной СС: `1h`).

- Автомат остается в начальном состоянии если $\bar{p}_0$, при этом также выдается y_0.
- Из s_1 осуществляется переход только в s_2 — осведомительные сигналы не важны (переход помечен $*$), при этом выдается управляющий сигнал y_2 (вектор `4h`).
- Из s_5 автомат безусловно возвращается в s_0 и выдает при этом набор сигналов $\{y_6, y_0\}$, т.е. вектор `41h`.

Из состояния s_1 автомат переходит только в состояние s_2, при этом значения осведомительных сигналов не важны и обозначены на схеме $*$, выдается управляющий сигнал y_0. И т.д.

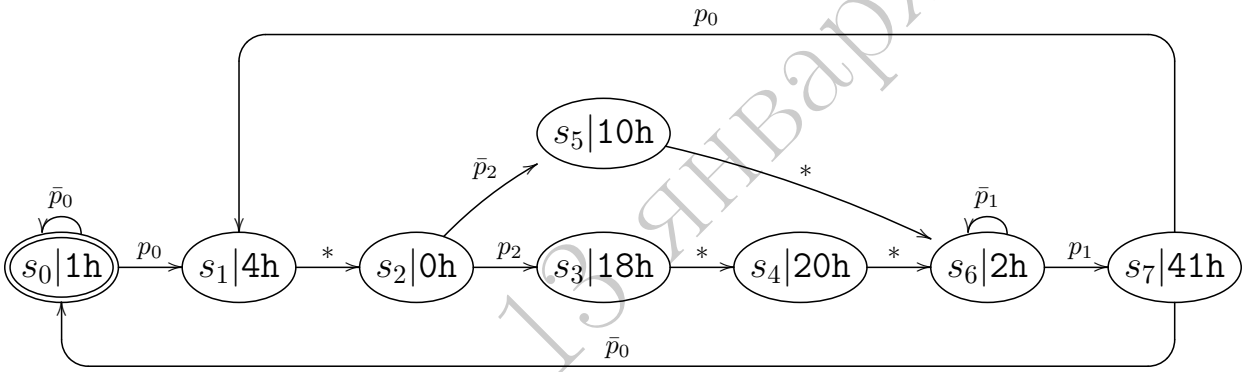

Рис. 1.16: Диаграмма переходов автомата Мура

На рисунке 1.16 приведена диаграмма автомата Мура. Далее следует описания нескольких переходов автомата.

- Находясь в начальном состоянии, автомат выдает управлающий сигнал y_0 (вектор `1h`). Если выполняется условие $\bar{p}_0$, т.е. $p_0 = 0$, то автомат остается в начальном состоянии, в противном случае — переходит в s_1.
- Из состояния s_3, в котором автомат выдает управляющие сигналы $\{y_4, y_3\}$, автомат безусловно переходит в s_4.
- Из состояния s_5 автомат может вернуться в начальное состояние или выполнить переход в s_0.

Микропрограммный автомат в своем составе имеет два элемента — микропроцессор и память *микрокоманд*. Такие автоматы довольно удобны в обращении, так как позволяют задать алгоритм своей работы простой записью

микропрограммы в память микрокоманд. Микрокоманда выполняется за один такт и сопровождается выдачей соответствующих управляющих сигналов.

1.2.1 Микропрограммные автоматы

Структура микропрограммного автомата Мили представлена на рисунке[5] 1.17, а Мура — на рисунке 1.18.

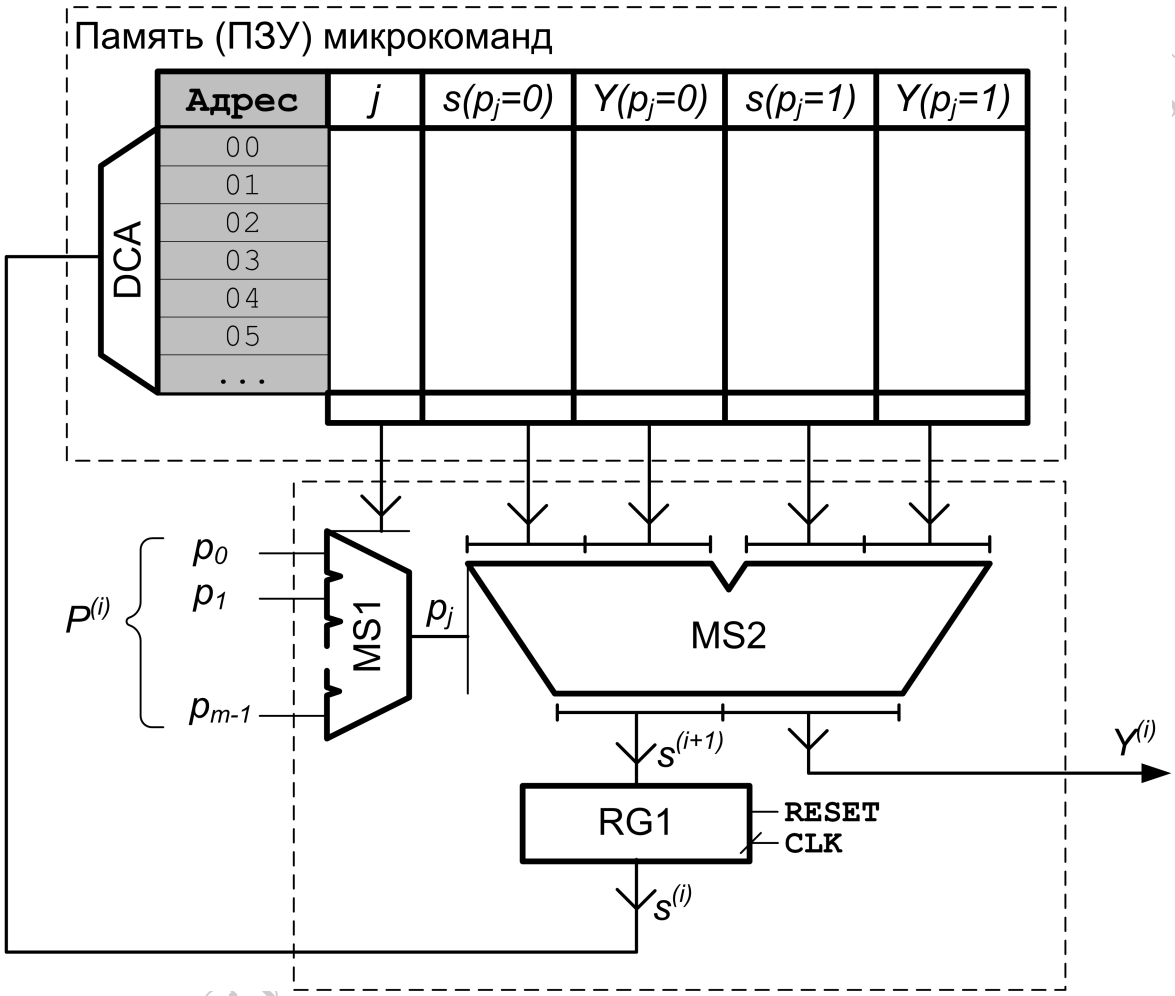

Рис. 1.17: Автомат Мили

В приведенных схемах микропрограммных автоматов выделены два блока — «Память (ПЗУ) микропрограмм» и элементарное управляющее устройство. Текущему состоянию автомата $S^{(i)}$ соответствует адрес в памяти микропрограмм, то есть текущее состояние определяет адрес микрокоманды. Адрес (состояние) в течение такта снимается с выхода `RG1`. По завершению такта, в регистре `RG1` будет запомнено новое состояние, которое будет выдаваться в течение следующего такта. `RG1` переключается в каждом такте под управлением тактового сигнала `CLC`.

[5] Мы намеренно отступили от некоторых правил оформления схем — например, мультиплексор MS1 в нашей схеме перевернут, а управление на MS2 подается слева

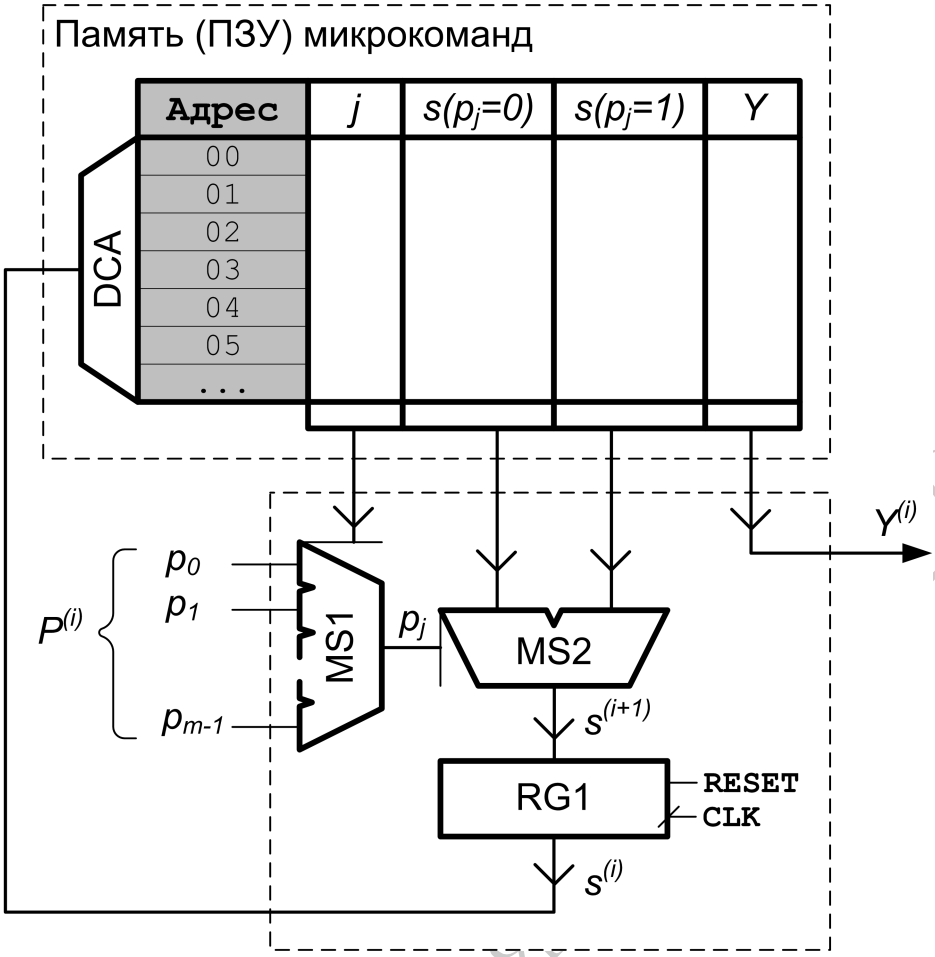

Рис. 1.18: Автомат Мура

Ячейка памяти имеет для обоих автоматов общие части:

- поле j — выбор осведомительного сигнала (т.е. на управление `MS2` будет поступать значение осведомителного сигнала p_j);
- поле $S(p_j = 0)$ — новое состояние автомата, которое будет занесено в `RG1`, если $p_j = 0$;
- поле $S(p_j = 1)$ — состояние, в которое перейдет автомат при $p_j = 1$;

Работа автомата начинается со сброса регистра `RG1` сигналом `RESET` в 0 — автомат переходит в состояние s_0.

Выход регистра `RG1` поступает на адресный вход памяти микрокоманд и через некоторое время значение ячейки памяти с указанным адресом поступает на выход модуля памяти. Значение поля j управляет мультиплексором `MS1`, на выходе которого появляется значение осведомительного сигнала p_j, которое в свою очередь поступает на вход управления `MS2`. На выходе `MS2` появляется одно из двух состояний автомата: либо значение поля $S(p_j = 0)$, либо — поля $S(p_j = 1)$. При переходе в следующий такт под управлением сигнала `CLC` регистр `RG1` запомнит новое состояние, сформировавшееся на выходе `MS2`.

В автомате Мили вектор управляющих сигналов снимается с выхода `MS2`, который осуществляет выбор из двух значений[6], хранимых в памяти: $Y(p_j = 0)$ и $Y(p_j = 1)$;

В автомате Мура вектор управляющих сигналов определяется только текущим состоянием, поэтому снимается непосредственно с модуля памяти (поле Y).

В i-м такте работы управляющего устройства происходит следующее:

1. Поступают осведомительные сигналы $P^{(i)}$. Эти осведомительные сигналы отражают изменения, произошедшие в предыдущем такте и в течение i-го такта не изменяются (так как снимаются обычно с выходов элементов памяти).

2. Управляющее устройство формирует и выдает управляющие сигналы i-го такта. Также управляющее устройство определяет состояние $s^{(i+1)}$ в которое оно перейдет по завершении i-го такта. Значения управляющих сигналов стабилизируются, они распространяются по схеме и на соответствующих входах элементов памяти устанавливаются корректные значения.

3. В конце такта (в зависимости от управляющих сигналов) в операционной части происходит защелкивание необходимых значений в элементах памяти. В частности, желательно сохранять в элементах памяти значения осведомительных сигналов.

Одному такту работы схемы автомата Мили на рисунке 1.17 можно сопоставить фрагменты граф-схемы алгоритма, приведенные на рисунке 1.19. Состояния автомата выделены серыми кружками-пометками. Автомат Мили в том же такте реагирует (управляющими сигналами) на осведомительные сигналы.

Граф-схема, соответствующая такту работы автомата Мура (рис. 1.18) приведена на рисунке 1.20. Автомат Мура реагирует на осведомительные сигналы только в следующем такте.

Видно, что автомат Мура тратит отдельный такт на анализ осведомительного сигнала, при этом в таком такте, конечно могут выдаваться и управляющие сигналы (в приведенном примере на Рис. 1.20 выдается `0h`). То есть, можно совместить вершину процесса и условную вершину в одном состоянии, если процесс на условие не влияет и тогда процесс и анализ условия *совмещаются* во времени. В противном случае, если имелось в виду, что *после* определенных действий, следует проанализировать осведомительный сигнал, то следует выделить два состояния (см. пример на рисунке 1.21).

[6] На практике автомат Мили «не прощает» ошибок проектирования: когда значение сигнала p_j меняется под воздействием управляющего сигнала в том же такте (такое бывает, когда значение p_j снимается с выхода элемента логики, а не памяти, и соответсвенно может измениться в течение такта). Такое

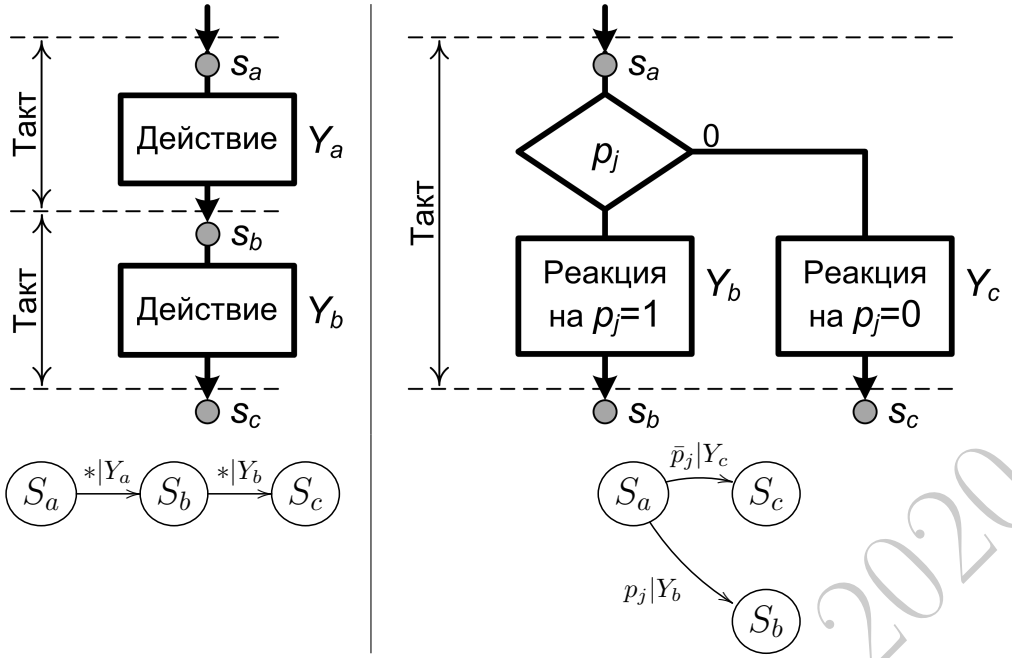

Рис. 1.19: Примеры фрагментов алгоритмов, реализуемых автоматом Мили за один такт

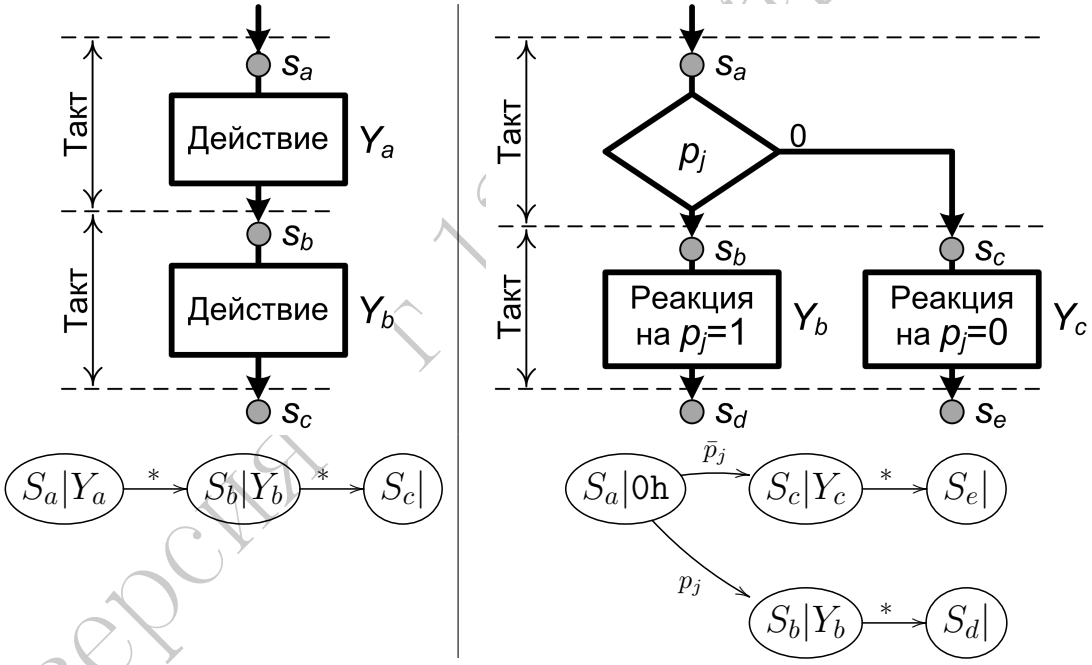

Рис. 1.20: Примеры фрагментов алгоритма, реализуемых автоматом Мура за один такт

На указанные фрагменты можно разбить любую произвольную схему алгоритма. Также при составлении алгоритма следует учитывать указанные особенности автомата, выполняющего этот алгоритм. Детальное обсуждение осо-

взаимовлияние может привести к «гонкам» осведомительных и управляющих сигналов и полной неопределенности в вопросе: «в какое же из двух возможных состояний перейдет автомат Мили?»

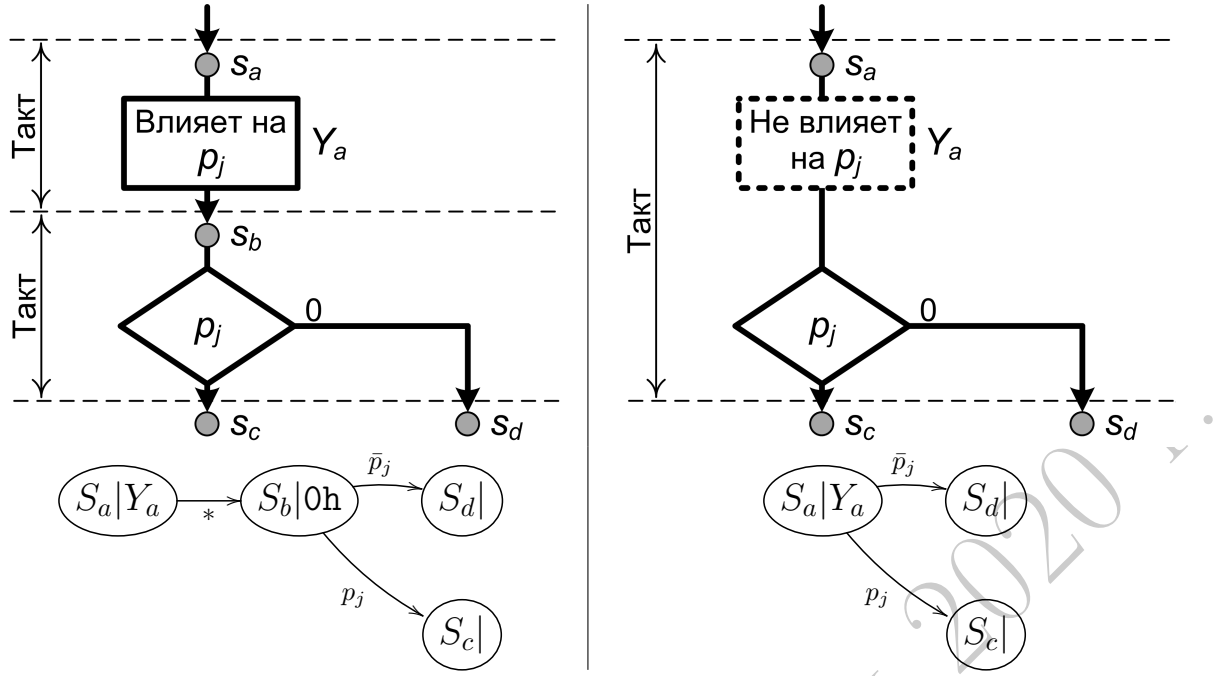

Рис. 1.21: Особенности выделения состояний автомата Мура

бенностей находится за рамками данного курса, поэтому далее рассматривается пример микропрограммирования, который позволить понять суть микропрограммного управления.

1.2.2 Соглашения о взаимодействии с ЦУУ

Особенности взамодействия ОЧ множительного устройства и ЦУУ были изложены в самом начале раздела 1. Введем следующие обозначения сигналов:

- p_0(`TASK`) — «Задание на шине» от ЦУУ;
- p_1(`BUS`) — «Шина твоя» от ЦУУ;
- y_0(`READY`) — «Свободен» в ЦУУ;
- y_1(`RESULT`) — «Результат готов» в ЦУУ;
- `CLK` — тактовые импульсы, синхронизирующие все устройства вычислительной системы в целом;
- `X(DATA)` — данные на шине данных, могут выдаваться как ЦУУ (исходные операнды), так и множительным устройством (результат).

Изменение сигналов принято изображать на временной диаграмме. Высокий уровень сигнала на диаграмме соответствует логической единице, низкий — нулю.

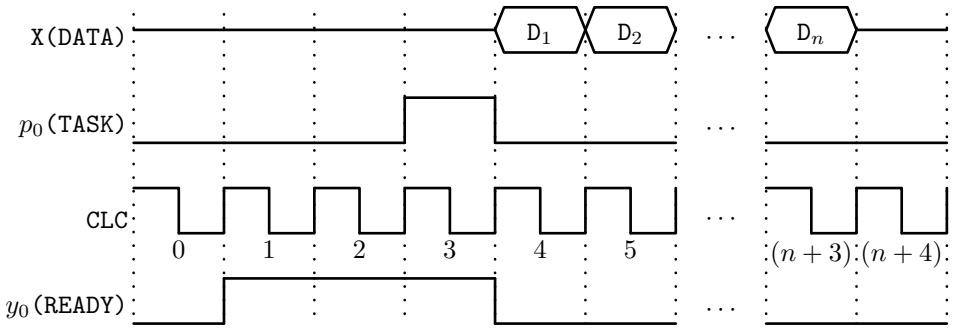

Рис. 1.22: Временная диаграмма получения задания

Состояние шины (жгута) изображается по особому: так как на диаграмме обычно нет места, чтобы изобразить все линии шины по отдельности, то состояние шины изображают как один сигнал (см. сигнал `X(DATA)`). Когда состояние шины не имеет значения (на рисунке 1.22 это такты 0 и 3), то рисуют линию посередине между высоким и низким уровнем. При этом по шине могут передаваться данные для других устройств.

Когда на шину поступают данные, которые важны, их изображают двумя линиями, проведенными на высоком и низком уровнях одновременно (такты 1, 2 на рисунке 1.22), а если нужно указать значение двоичного вектора на шине, то между линиями пишут соответствующее шестнадцатиричное значение.

На рисунке 1.22 изображена временная диаграмма получения задания от ЦУУ. Устройства следуют следующим соглашениям.

1. УЧ выдает в ЦУУ сигнал y_0`(READY)`. См. такт 1. Таким образом УЧ сообщает ЦУУ, что множительное устройство готово решать новую задачу. Сигнал будет удерживаться до тех пор, пока ЦУУ не даст задание.

2. Когда ЦУУ потребуется выполнить умножение, то оно, убедившись, что множительное устройство готово ($y_0 = 1$), выдает в УЧ сигнал p_0`(TASK)`. См. такт 3. В последующих тактах ЦУУ будет выдавать фрагменты задания $\mathtt{D}_1, \ldots, \mathtt{D}_n$. В общем случае, задание состоит из одного фрагмента. ЦУУ выдает сигнал p_0`(TASK)` только в течение одного такта.

3. УЧ *должна* в следующем такте, после получения сигнала p_0`(TASK)` от ЦУУ снять сигнал y_0`(READY)` и начать чтение с шины первого фрагмента задания. См. такт 4. Пока УЧ решает задачу, сигнал y_0`(READY)` не должен выдаваться.

На рисунке 1.23 изображена временная диаграмма выдачи результата в ЦУУ.

1. УЧ выдает в ЦУУ сигнал y_1`(RESULT)`. См. такт 1. УЧ будет ждать момента, когда ЦУУ освободит шину и будет удерживать этот сигнал, пока не получит разрешение на передачу. См. такты 1-3. Количество тактов, в

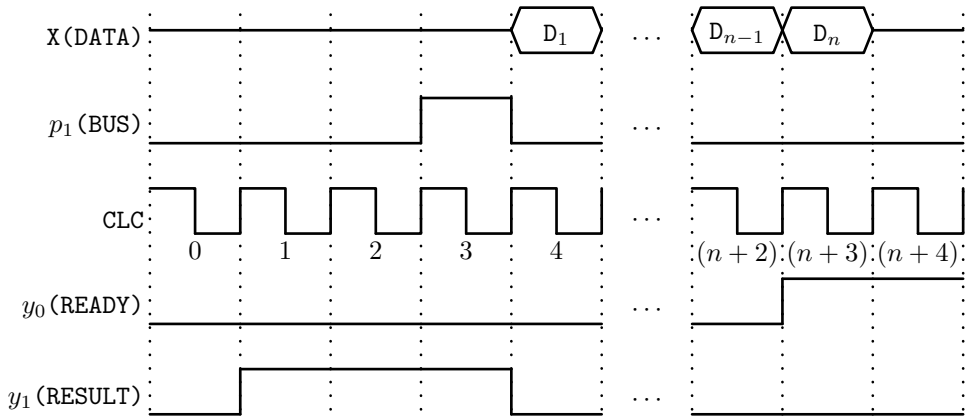

Рис. 1.23: Временная диаграмма выдачи результата

течение которых ЦУУ освобождает шину, варьируется, и непоправимой ошибкой будет выдача результата на «занятую» шину.

2. ЦУУ, убедившись, что результат готов, выдает в течение одного такта сигнал p_1(BUS). См. такт 3.

3. УЧ, приняв сигнал p_1(BUS) в следующем такте *должно* снять сигнал y_1(RESULT) и начать выдавать первый фрагмент результата. См. такт 4.

4. УЧ как можно раньше должно выдать в ЦУУ сигнал y_0(READY). Обычно, сигнал y_0(READY) выдается в том же такте, в котором выдается последний фрагмент результата. См. такт $(n + 3)$.

1.2.3 Пример микропрограммирования

В качестве примера рассматривается задача получения дополнительного кода из прямого. Несомненно, эту задачу можно решить проще.

Операционная часть устройства приведена на рисунке 1.24.

Назначение осведомительных и управляющих сигналов следующее:

- p_0(TASK) — «Задание на шине» от ЦУУ;
- p_1(BUS) — «Шина твоя» от ЦУУ;
- y_0(READY) — «Свободен» в ЦУУ;
- y_1(RESULT) — «Результат готов» в ЦУУ;
- p_2 — RG1[7], знак операнда;
- y_2 — запись в RG1
- y_3 — инверсия;

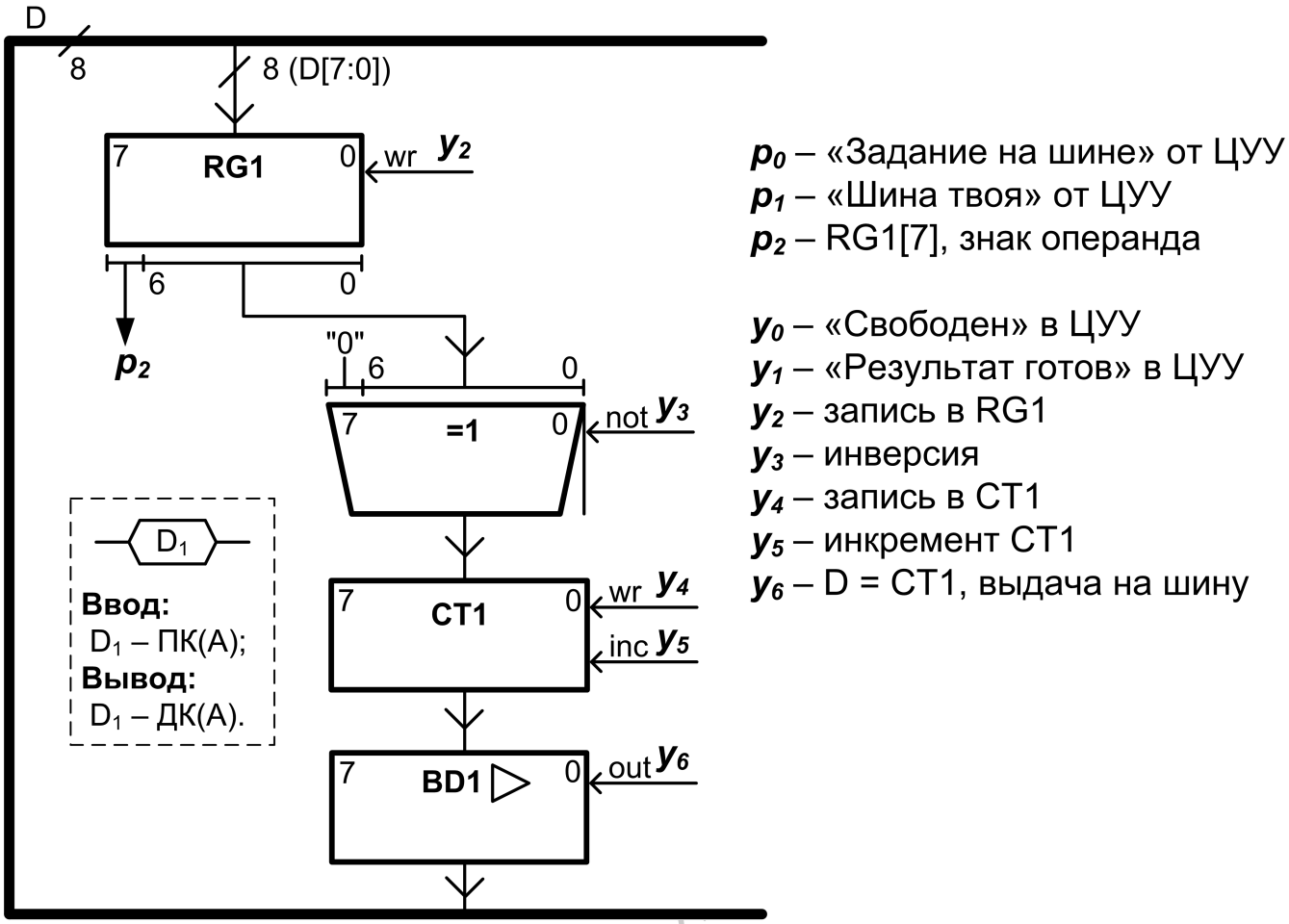

Рис. 1.24: Операционная часть преобразователя ПК $\mapsto$ ДК

- y_4 — запись в CT1;
- y_5 — инкремент CT1;
- y_6 — X=CT1, выдача результата на шину.

С помощью данной операционной части задачу можно решить так:

1. получить задание и записать операнд в RG1 (y_2);
2. если знак операнда $p_2 = 0$, то перейти к шагу 3, иначе к шагу 4;
3. записать модуль операнда в CT1 (y_4); перейти к шагу 6;
4. записать инвертированный модуль операнда в CT1 (y_3, y_4);
5. инкрементировать CT1 (y_5);
6. в CT1 получен результат; выдать его в ЦУУ.

В зависимости от управляющего автомата, в этот базовый алгоритм придется внести некоторые изменения. В следующих параграфах приводятся микропрограммные реализации данного алгоритма[7] с помощью автоматов Мили и Мура.

Ручное управление

При ручном управлении оператор (студент) сам определяет последовательность выполнения необходимых команд в зависимости от значений осведомительных сигналов. Для режима ручного управления необходимо записать набор отдельных команд, при этом каждая последующая команда выбирается вручную.

На рисунке 1.26 изображен алгоритм[8] работы преобразователя ПК ↦ ДК при ручном управлении.

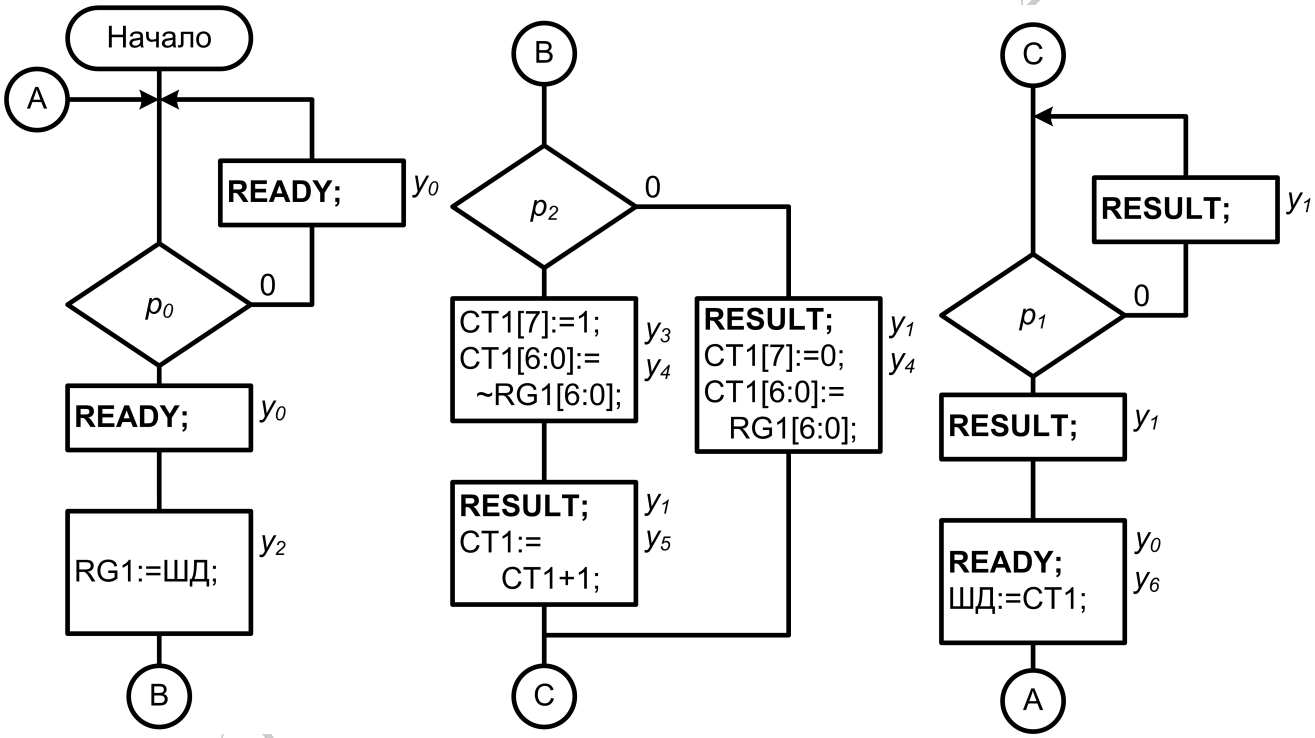

Рис. 1.25: Алгоритм работы преобразователя ПК ↦ ДК под управлением автомата Мили

[7] Следует отметить, что приведенные реализации, как для автомата Мили, так и Мура, можно оптимизировать. Оптимизация не была сделана авторами с целью упростить изложение

[8] Строго говоря, это метод, а не алгоритм — алгоритм обязан завершаться через конечное число шагов [4]

Автомат Мили

На рисунке 1.26 изображен алгоритм[9] работы преобразователя ПК ↦ ДК под управлением автомата Мили. Серыми кружками отмечаются состояния (s_0–s_5).

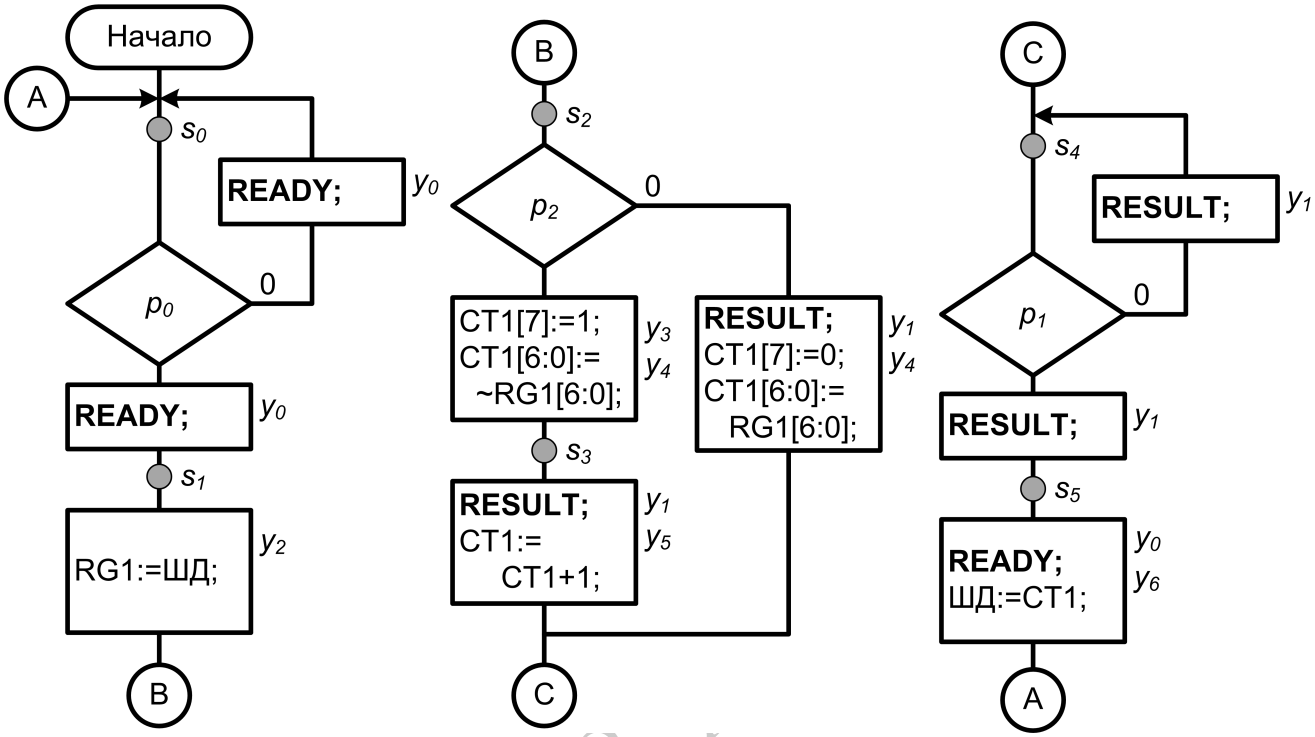

Рис. 1.26: Алгоритм работы преобразователя ПК ↦ ДК под управлением автомата Мили

Отметка состояния ставится над условной вершиной (см. рисунок 1.19). При отсутствии таковой, например, между двумя последовательными блоками процеса[10], вводится фиктивная вершина условия с одинаковым исходом для истины и лжи. В данном примере фиктивные условные вершины введены для состояний s_1, s_3, s_5.

Также следует отметить, что сигнал y_0(ACK) («Задание принял») выдается в следующем такте (при переходе из состояния s_1). Автомат мог среагировать на поступивший сигнал p_0 в текущем такте и выдать подтверждение приема задания, но это могло привести к гонкам[11] и нарушению соглашения приема задания, приведенного на рисунке 1.22.

Соответствующая алгоритму диаграмма переходов автомата уже была приведена на рисунке 1.15. Теперь несложно задать микропрограмму для автома-

[9] Строго говоря, это метод, а не алгоритм — алгоритм обязан завершаться через конечное число шагов [4]

[10] Блок процесса обозначается на блок-схеме прямоугольником

[11] Потому что упрвляющий автомат ЦУУ, при поступлении подтверждения приема должен снять сигнал выдачи задания и, если он это сделает в текущем такте, то начнутся гонки сигналов p_0 и y_0

та (рисунок 1.27). Это можно сделать как по отмеченной граф-схеме, так и по диаграмме переходов.

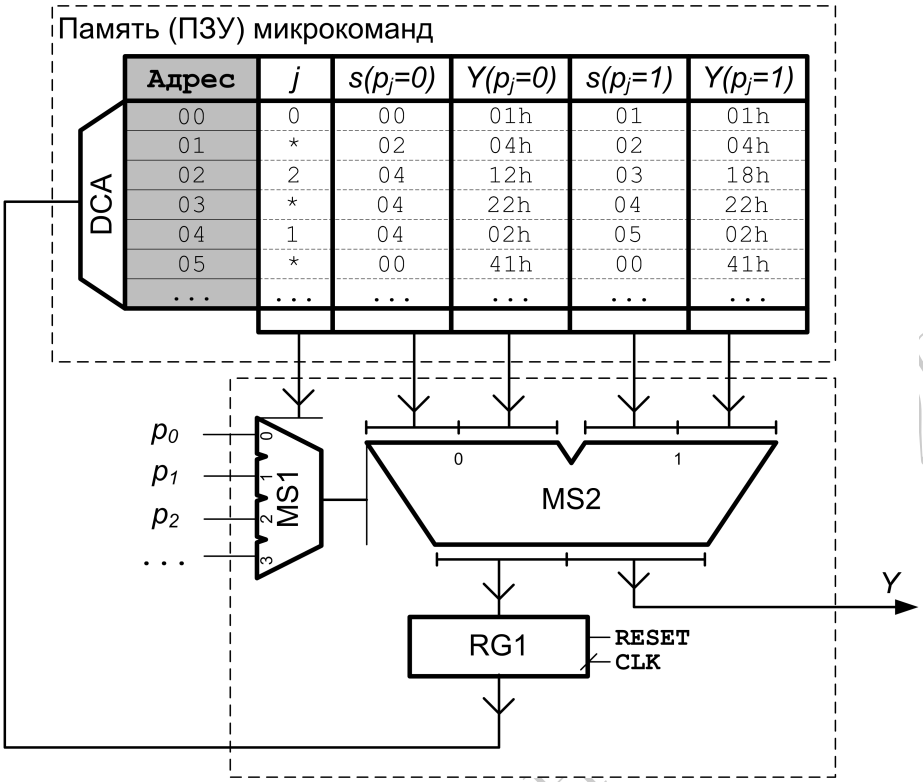

Рис. 1.27: Микропрограмма автомата Мили

Временная диаграмма работы преобразователя ПК ↦ ДК под управлением автомата Мили приводится на рисунке 1.28.

Автомат Мура

Алгоритм работы преобразователя ПК ↦ ДК, учитывающий особенности управляющего автомата Мура приведен на рисунке 1.29.

Диаграмма состояний автомата Мура уже была приведена на рисунке 1.16. Обычно у автомата Мура, эквивалентного автомату Мили, состояний больше, и времени (тактов) на работу он также тратит больше.

Прошивка памяти микропрограммного автомата Мура приводится на рисунке 1.30.

Временная диаграмма работы автомата Мура приведена на рисунке 1.31.

1.3 Работа с лабораторной установкой «Microcode»

Цель лабораторной установки «Microcode» — закрепление теоретических основ реализации арифметических операций.

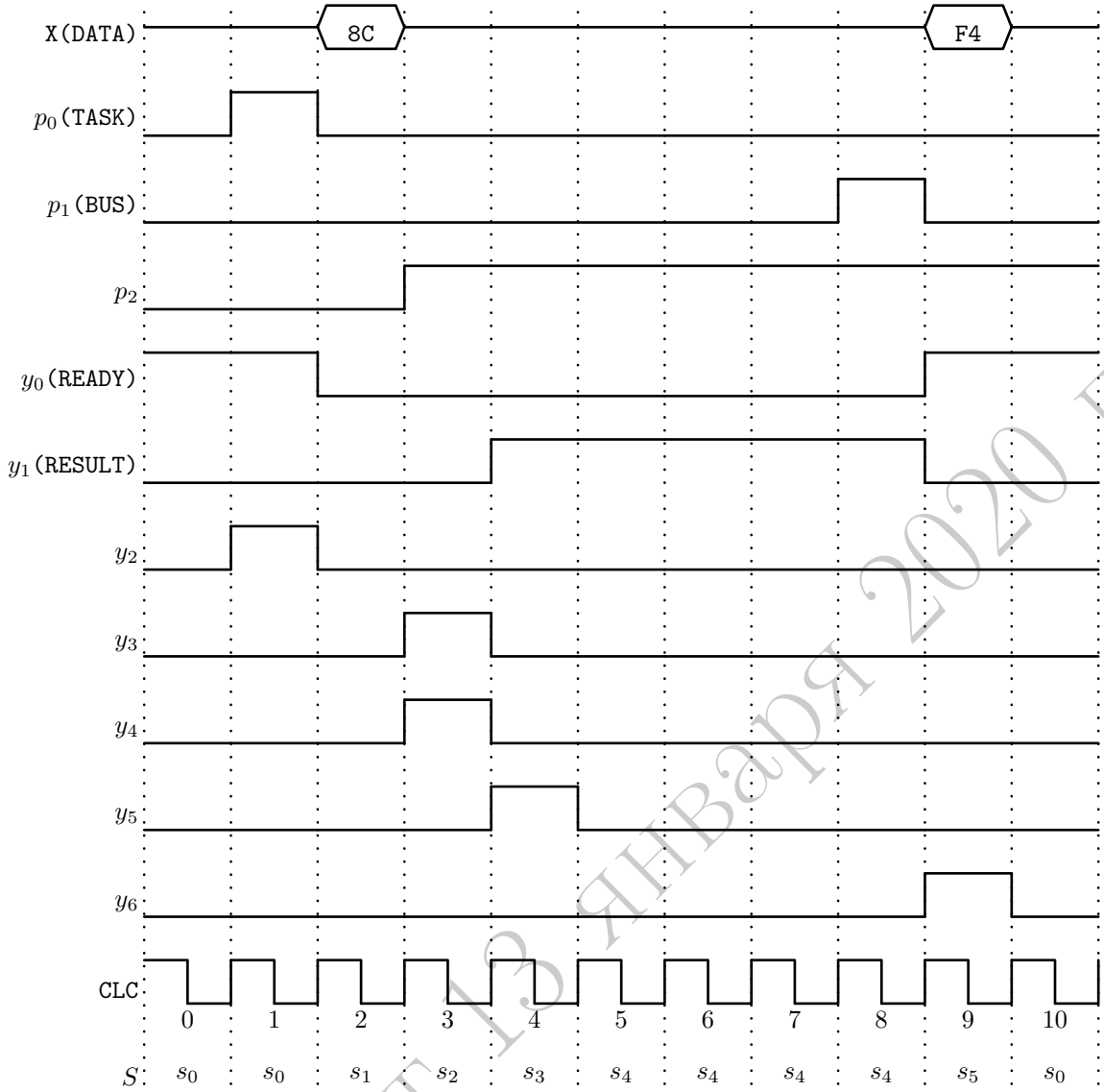

Рис. 1.28: Временная диаграмма работы преобразователя ПК $\mapsto$ ДК под управлением автомата Мили

Поставленная цель достигается за счет того, что:

- предлагается готовая операционная часть устройства (см. рисунок 1.1 и теорию в разделе 1), в структуре которой требуется разобраться и понять, как имеющимися средствами решить задачу (т.е. выполнить некоторую арифметическую операцию);
- моделируется работа центрального управляющего устройства, которое выдает задание и принимает результаты по определенным правилам;
- «Microcode» позволяет студенту задать собственные входные данные и выполнить пошаговое управление в режиме отладки;
- «Microcode» автоматически выполняет многократное тестирование пра-

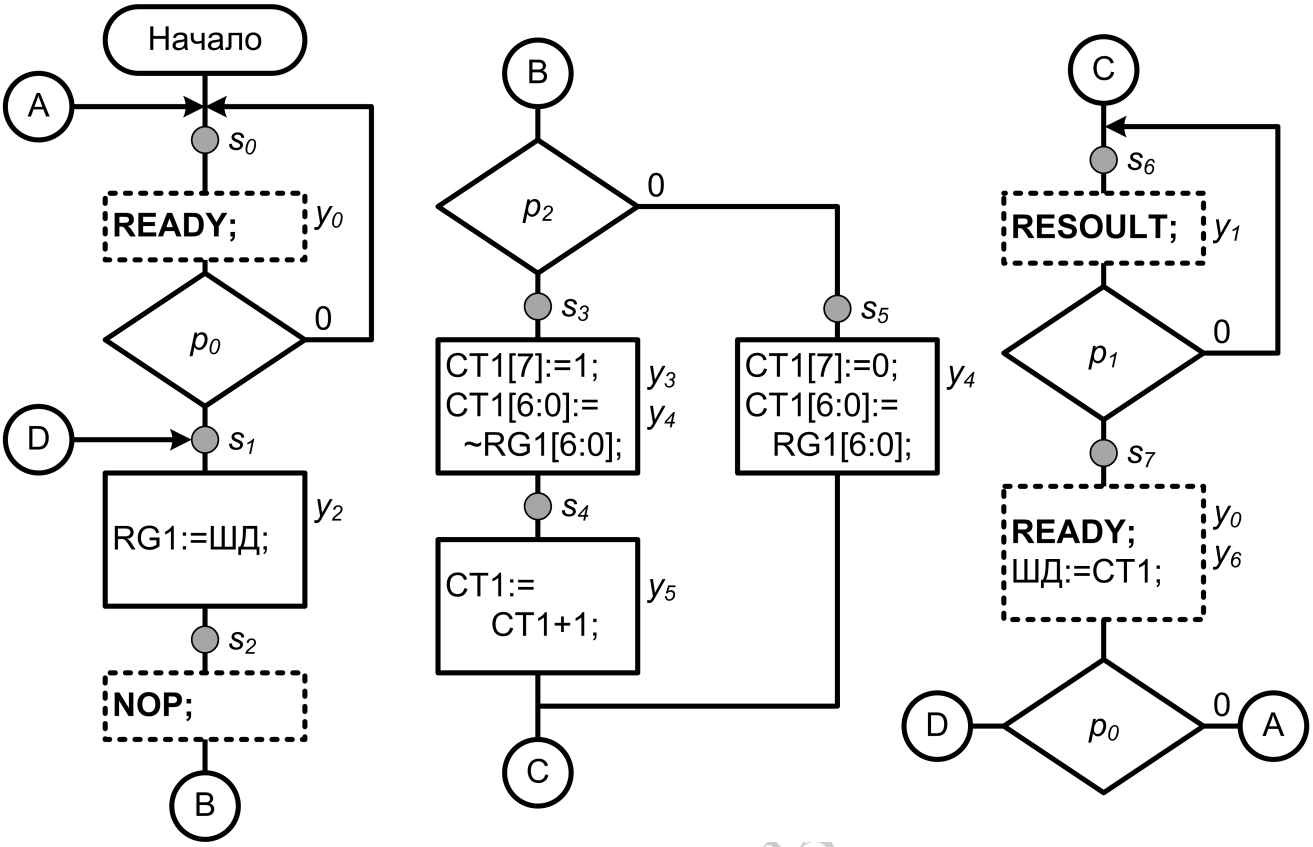

Рис. 1.29: Алгоритм работы преобразователя ПК $\mapsto$ ДК под управлением автомата Мура

вильности решения задачи по методу «черного ящика»;

- выполняемые «Microcode» тесты покрывают все ветки правильного алгоритма решения задачи (арифметической операции);
- тесты покрытия создаются «Microcode» на основе генератора случайных чисел;
- в «Microcode» заложена возможность генерации индивидуальных вариантов заданий, создаваемых за счет перестановки индексов управляющих сигналов y, что порождает $n!$ возможных уникальных вариантов кодирования, при среднем $n \geq 12$;
- в диалогах «Microcode» для ввода отладочных данных автоматически рассчитывается результат, а также выводятся десятичные представления числовых операндов, что дает возмножность глубже разобраться в форматах и кодах.

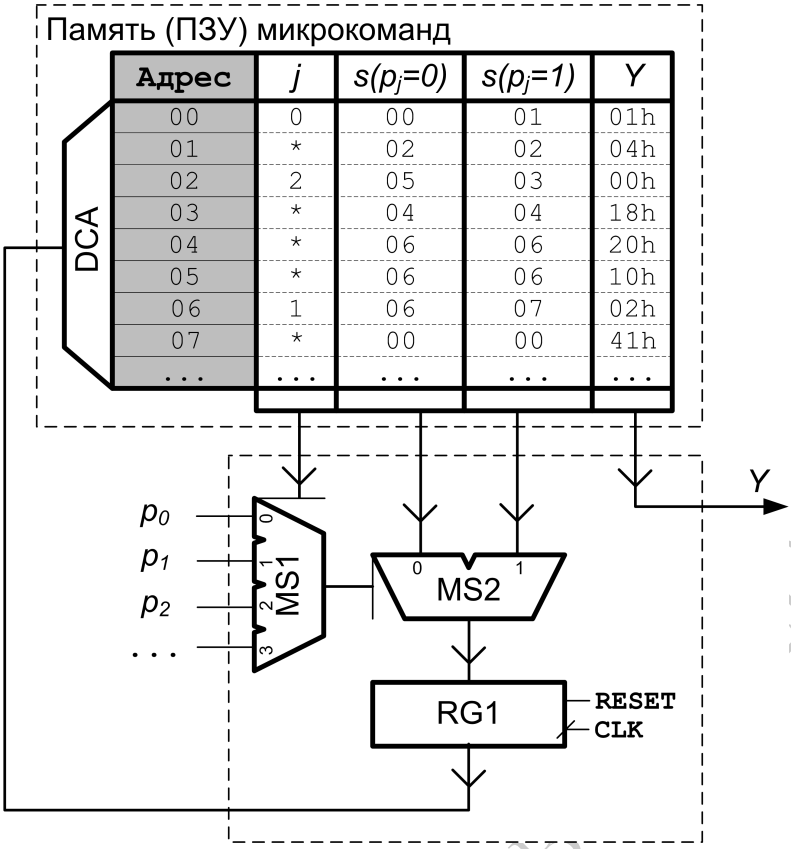

Рис. 1.30: Микропрограмма автомата Мура

1.3.1 Принципы работы «Microcode»

Лабораторная установка «Microcode» позволяет загружать групу заданий. Например, в рамках одной группы могут быть загружены задания на умножение чисел различными способами (см. рисунок 1.32).

Отдельное задание представляет собой вычислительное устройство, постороенное в соответствии с принципами, изложенными в разделе 1 и способное выполнять ту или иную сложную вычислительную операцию, например, умножение или деление.

Пользователю предлагается готовая схема операционной части, в устройстве и назначении которой он должен разобраться, опираясь на изученную теорию. В распоряжении пользователя имеется набор управляющих сигналов, с помощью которых он может изменять состояние операционной части. Задача пользователя — реализовать алгоритм работы сложной вычислительной операции.

У пользователя имеются следующие возможности реализовать алгоритм работы устройства:

- вручную выполнить выдачу необходимых управляющих сигналов, предварительно задав наборы управляющих сигналов в ПЗУ микрокоманд;

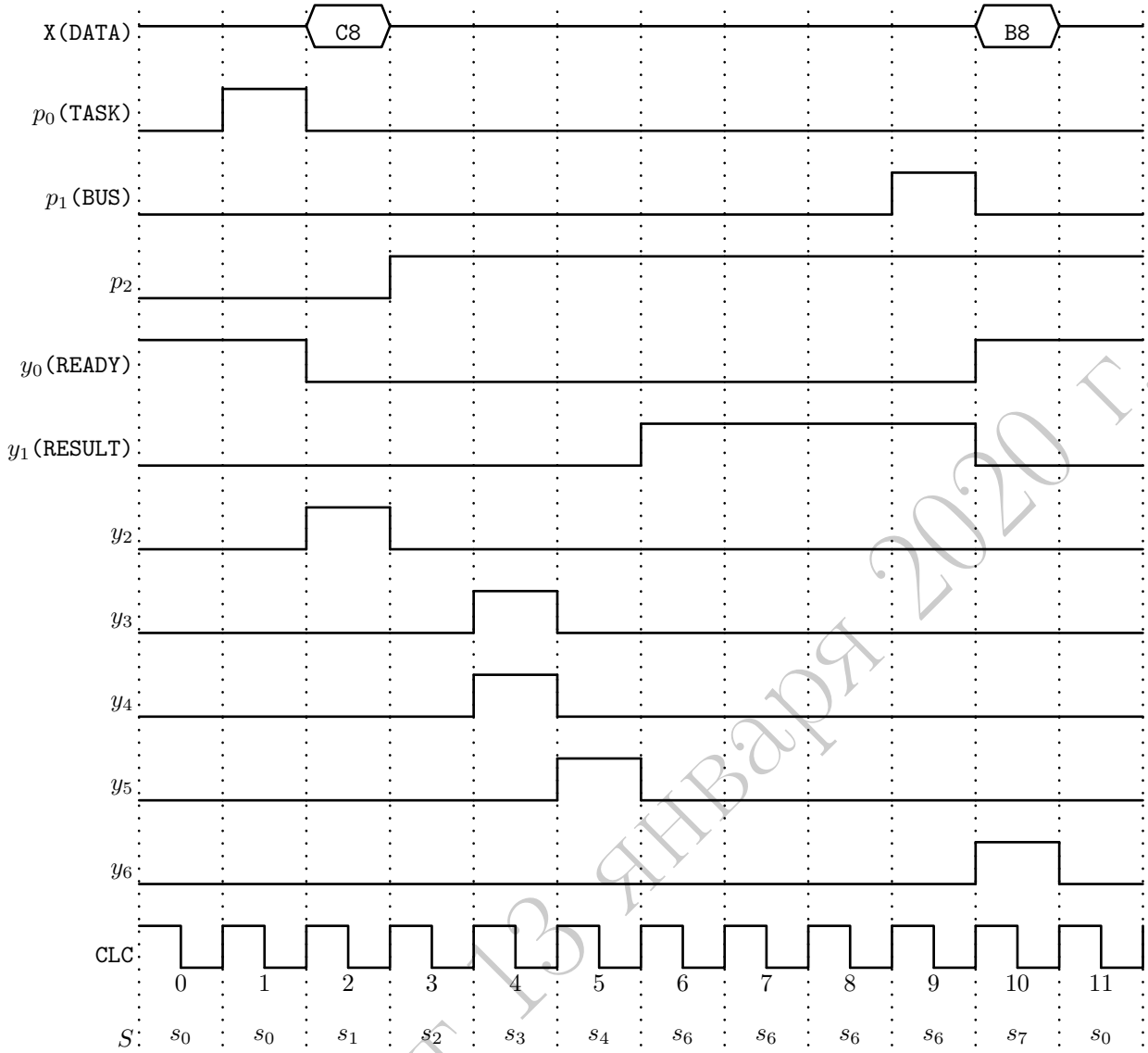

Рис. 1.31: Временная диаграмма работы преобразователя ПК $\mapsto$ ДК под управлением автомата Мура

- запрограммировать автомат Мили или Мура, заполнив таблицу ПЗУ микропрограмм.

Все значения в таблицы ПЗУ заносятся в шестнадцатеричной системе счисления. Управляющие сигналы кодируются двоичным вектором, который в ПЗУ вносится в шестнадцатеричном представлении, например, если требуется выдать сигналы y_5, y_3, y_2, y_0, то в соответствующей ячейке ПЗУ нужно записать значение `0x2D`.

По команде пользователя «Microcode» выполняет автоматическое тестирование составленного пользователем алгоритма. Чтобы составить корректный алгоритм, пользователь может использовать специальный режим отладки, в котором он может задать исходные данные и в пошаговом режиме отследить изменения, происходящие в операционной части.

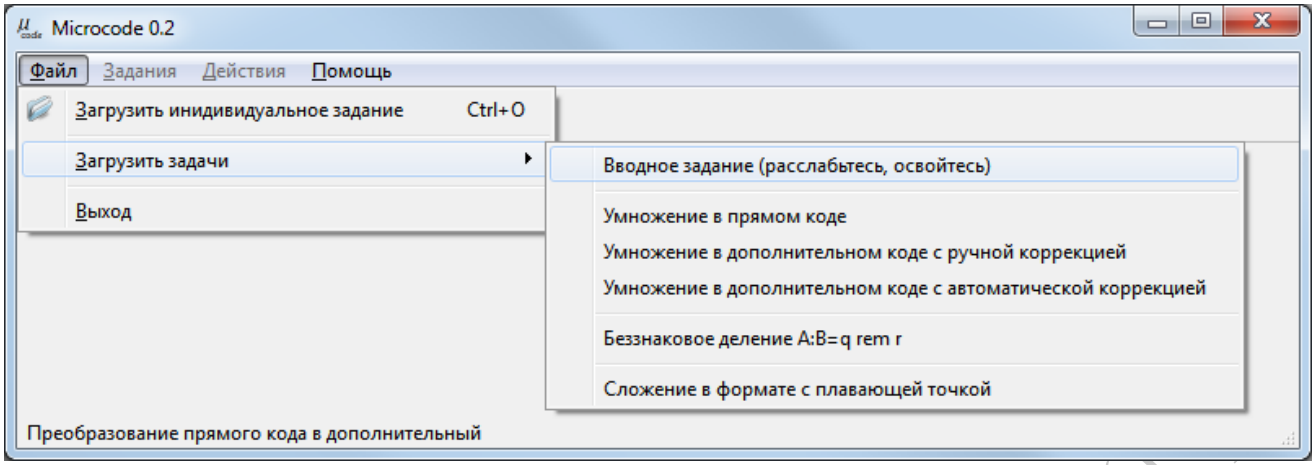

Рис. 1.32: Главное окно «Microcode» (русский язык интерфейса)

Прохождение автоматических тестов является допуском к защите разультатов работы.

1.3.2 Задания на лабораторные работы

В рамках лабораторной работы студент выполняет задание, которое выдает преподаватель.

В лабораторной установке «Microcode» реализованы следующие задания:

- перевод прямого кода в дополнительный;
- умножение в прямом коде;
- умножение в дополнительном коде с ручной коррекцией;
- умножение в дополнительном коде с автоматической коррекцией;
- беззнаковое деление $A \div B = (q \ \texttt{rem} \ r)$;
- сложение в формате с плавающей точкой.

1.3.3 Выполнение задания

После выбора группы заданий, становится активным пункт меню «Задания»(«Tasks»), позволяющий переключаться между заданиями в рамках группы (см. рисунок 1.33). Менять группу заданий в дальнейшем нельзя.

Пункт меню «Действия» позволяет выполнить те или иные действия в текущем задании:

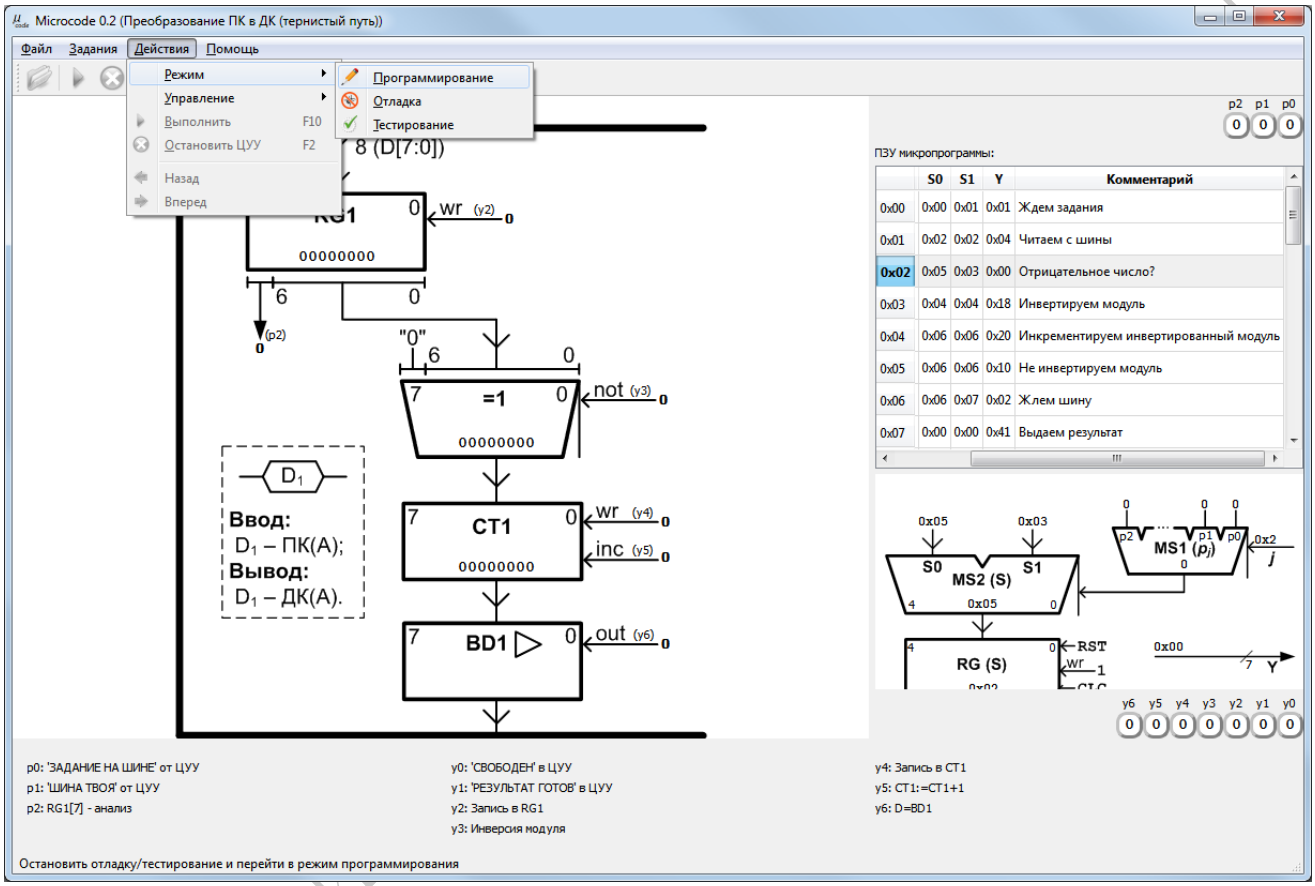

Рис. 1.33: «Microcode» после загрузки заданий

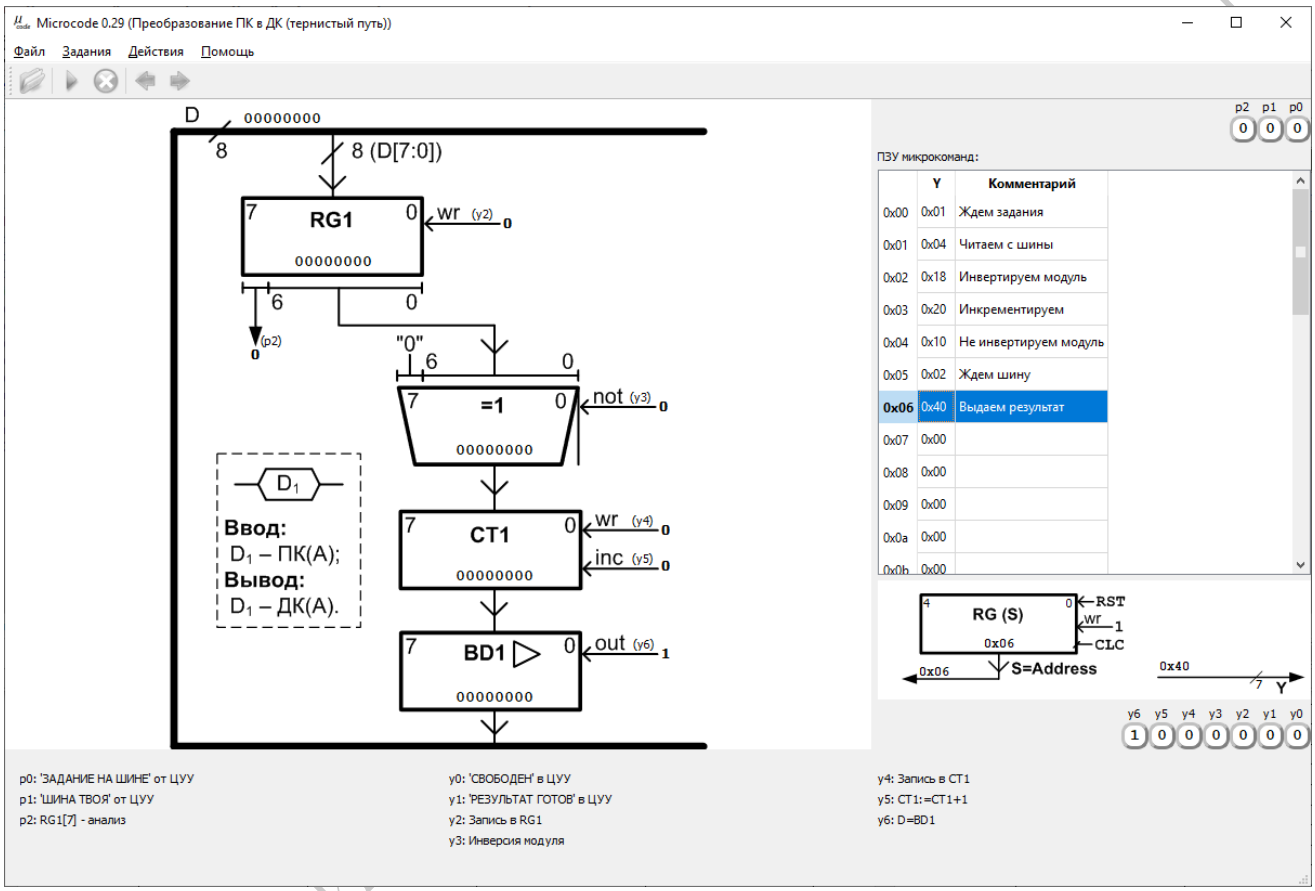

Рис. 1.34: «Microcode» после загрузки заданий

- выбрать один из режимов работы: «Программирование» (), «Отладка» () или «Тестирование» ();
- выбрать управление: «Ручное» (), «Автомат Мили» () или «Автомат Мура» ();
- выполнять/останавливать шаги отладки или автотесты;
- перемещаться в отладочном режиме по истории тактов.

Режим работы «Программирование» («Programming») предназначен для редактирования ПЗУ управляющей части. Как только редактирование ячейки ПЗУ завершается, соответствующие управляющие сигналы распространяются по схеме, но ошибки управления не выдаются, даже если они есть.

Режим «Отладка» («Debug») предназначен для пошагового выполнения придуманного студентом алгоритма управления. В момент переключения в этот режим студенту предлагается ввести исходные данные. После того, как данные будут введены, «Microcode» переключается в пошаговый (потактный) режим тестирования. В случае ручного управления («Управление/Ручное» («MCU/Manual»)) пользователь выбирает в таблице «ПЗУ микрокоманд» строку с нужной командой и нажимает кнопку «Выполнить» («Execute») или горячую клавишу «F10», что приводит к переключению в следующий такт. В случае автоматического управления («Автомат Мили» («Mili»), «Автомат Мура» («Moore»)) текущая ячейка ПЗУ микропрограммы (строка таблицы) подсвечивается. Если в процессе выполнения теста фиксируется ошибка, то тестирование прекращается, выводится соответствующее сообщение, и выполняется переход в режим программирования. Если алгоритм выполнен безошибочно и в ЦУУ корректно выдан правильный результат, то выводится информационное сообщение об успешном прохождении отладочного теста.

В режиме «Тестирование» («Autotest») выполняется тестирование на случайным образом генерируемых входных данных. Состояние операционной части при этом не отображается. При ручном управлении автотесты выполняются в пошаговом режиме, в противном случае — в автоматическом.

В режимах отладки и тестирования фиксируются следующие ошибки, приводящие к прекращению теста:

- выполняется чтение данных с «пустой» шины;
- выполняется выдача данных на «занятую» шину;
- нарушается протокол взаимодействия с ЦУУ, например, до выдачи результата, выдается сигнал «Свободен»;

- выдаются конфликтующие сигналы управления элементом памяти, например на регистр одновременно подаются управляющие сигналы сдвига и сброса.

Чтобы считать операнды, требуется выполнить следующие действия.

- Выдавать сигнал «Свободен» (y_0, Y = `0x0001`) в течение некоторого числа тактов (не более 6), пока осведомительный сигнал «Задание на шине» (p_0) не установится в 1.
- Сигнал «Задание на шине» выдается ЦУУ в течение одного такта, в этом такте также нужно выдавать сигнал «Свободен».
- ЦУУ, в течение одного или нескольких тактов, следующих после такта, в котором выдавался сигнал «Задание на шине» выдает на шину фрагменты задания. Необходимо считать эти фрагменты в регистры.

Чтобы выдать результат, требуется выполнить следующие действия.

- Выдавать сигнал «Результат готов» (y_1, Y = `0x0002`) в течение некоторого числа тактов (не более 6), пока осведомительный сигнал «Шина твоя» (p_0) не установится в 1.
- Сигнал «Шина твоя» выдается ЦУУ в течение одного такта, в этом такте также нужно выдавать сигнал «Результат готов».
- В течение одного или нескольких тактов, следующих после такта, в котором выдавался сигнал «Шина твоя» следует в определенном в задании порядке выдавать фрагменты результата на шину.

Результатом работы являются заполненные ПЗУ микрокоманд (ПЗУ микропрограмм). Допуском к защите результатов работы является успешное прохождение студентом отладочных или автоматических тестов.

1.3.4 Защита результатов работы

Результаты работы оформляются в виде отчета, в котором для каждого задания необходимо представить:

- Таблицы заполненных ПЗУ с построчными комментариями.
- Алгоритмы работы управляющего устройства, представленные в виде графической схемы. Напротив каждого блока процесса должны быть указаны выдаваемые управляющие сигналы.

- Таблицы, отражающие пошаговое (потактное) выполнение алгоритма на примере конкретных входных значений.
- Выводы в формате формулы изобретения[12].
- Предложения по оптимизации операционной части или микропрограммы.

Защищая результаты своей работы, студент должен

- *знать*:
  - теоретические основы реализуемой в задании вычислительной операции,
  - принципы работы всех элементов схемы операционной части вычислительного устройства,
  - протоколы обмена данными с центральным управляющим устройством: «получение задания» и «выдача результата»;
- *уметь*:
  - объяснить, как на предложенной схеме можно реализовать вычислительную операцию (например, в задании умножения в дополнительном коде с ручной коррекцией, указать какие сигналы, в какой последовательности подать, чтобы выполнить коррекцию псевдопроизведения множителем),
  - кодировать двоичные векторы управляющих сигналов,
  - реализовать алгоритм вычислительной операции, составляя необходимые наборы микрокоманд или реализуя микропрограмму,
  - оценить время выполнения микропрограммы в тактах в зависимости от аргументов: минимальное время, максимальное время, среднее время;
- *владеть навыками*:
  - перевода двоичных чисел в шестнадцатеричную систему счисления,
  - представления чисел в различных форматах,
  - составления микропрограмм для автоматов Мили и Мура (дополнительно),
  - работы в лабораторной установке «Microcode».

[12] Формула изобретения состоит из ограничительной (описательной) и отличительной частей и имеет типовую структуру: «Изобретение X, состоящее из (список составляющих элементов), отличается тем, что (список отличительных признаков и особенностей)». Например, устройство умножения первым способом в прямом коде, реализованное на основе…, отличается тем, что…

В заключение

Данный текст подготовлен в издательской системе LaTeX 2_ε (авторы использовали MiCTeX.). Эта издательская система является стандартом де факто в научных и технических кругах. Для иллюстраций были использованы свободные редакторы векторной графики, свободный пакет Xy-pic, а также программа METAPOST.

Заинтересовавшимся версткой в LaTeX можно рекомендовать следующие книги: [5, 6, 7]. Про предшественника LaTeX — программу TeX следует читать бестселлер от автора[13] [8]. Работа в METAPOST раскрыта в руководстве пользователя автором системы Джоном Хобби [9], а основы работы изложены в статье Троя Хендерсона [10].

s

[13] TeX на самом деле является ядром LaTeX

Литература

[1] Б.Г.Лысиков. Арифметические и логические основы цифровых автоматов / Б.Г.Лысиков. — 2 изд. — Мн.: Выш. школа, 1980.

[2] А.Я.Савельев. Прикладная теория цифровых автоматов / А.Я.Савельев. — М.: Высшая школа, 1987.

[3] ИПК ИЗДАТЕЛЬСТВО СТАНДАРТОВ. — ГОСТ 2.708-81: правила выполнения электрических схем цифровой вычислительной техники, 1982. — 01.

[4] Д.Э.Кнут. Искусство программирования, основные алгоритмы / Д.Э.Кнут. — 3 изд. — М.: Вильямс, 2010. — Т. 1.

[5] И.Котельников. LaTeX по-русски / И.Котельников, П.Чеботаев. — Новосибирск: Сибирский хронограф, 2009.

[6] Е.М.Балдин. Компьютерная типография LaTeX / Е.М.Балдин. — СПб.: БХВ-Петербург, 2008.

[7] С.М.Львовский. Набор и верстка в системе LaTeX / С.М.Львовский. — М.: МЦНМО, 2006.

[8] Д.Э.Кнут. Все про TeX / Д.Э.Кнут. — М.: Вильямс, 2003.

[9] Hobby, J. D. — METAPOST, a user manual, 1.53 edition, 2013.

[10] Henderson, T. A beginner's guide to metapost for creating high-quality graphics / Troy Henderson // TUGboat. — Vol. 28. — 2007.