АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Лабораторный практикум

1. ПРОГРАММНЫЕ МОДЕЛИ АРИФМЕТИКО-ЛОГИЧЕСКИХ УСТРОЙСТВ

Рассматриваемые ниже программные модели ***арифметико-логического устройства*** (АЛУ) значительно упрощены по сравнению с реальными арифметико-логическими устройствами процессоров. Реальное арифметико-логическое устройство связано рядом линий с другим устройством компьютера – устройством управления выборкой и выполнением команд. В число этих линий входят линии управления входными и, возможно, выходными регистрами АЛУ, а также линии внешнего управления управляющим автоматом, которые используются для настройки автомата на выполнение каждой конкретной операции. Поскольку на предлагаемых моделях предполагается выполнять лишь отдельные операции, внешние связи АЛУ в них не описаны.

Программные модели не описывают также и механизм тактирования АЛУ. При стандартном подходе к построению АЛУ каждый узел *операционного автомата*, имеющий память, а также регистры *управляющего автомата* имеют входы тактирования. На все входы тактирования подаются периодические прямоугольные импульсные сигналы от одного устройства, называемого *тактовым генератором*. АЛУ проектируются, например, таким образом, что выходные сигналы управляющего автомата изменяются скачком при положительных перепадах тактирующего сигнала, а состояния узлов с памятью операционного автомата – при отрицательных перепадах. В реальных арифметико-логических устройствах система тактирования значительно сложнее, поскольку разработчик стремится максимально увеличить быстродействие проектируемого устройства. Однако с точки зрения построения *микропрограммы* механизм тактирования существенного значения не имеет.

Ниже рассматриваются две программные модели АЛУ:

1) программная модель ALU-1;
2) программная модель ALU-R.

1.1. Программная модель ALU-1

Программная модель ALU-1 включает две составные части – *операционный автомат* 8-разрядного двоичного арифметико-логического устройства и *управляющий автомат с программируемой логикой*, который может работать как автономно, так и взаимодействуя с операционным автоматом.

1.1.1. Операционный автомат арифметико-логического устройства ALU-1

Рассматриваемая структура операционного автомата почти универсальна и позволяет реализовать большинство известных алгоритмов арифметических преобразований. Операционный автомат предназначен для реализации арифметических операций над двоичными числами, представленными в 8-разрядном формате с фиксированной запятой в беззнаковой и знаковой формах (диапазоны представления чисел – от 0 до 255 и от –127 до 127 соответственно), а также для реализации логических операций над 8-разрядными двоичными числами.

Операционный автомат позволяет с помощью микропрограмм реализовать следующие арифметические операции:

- сложение и вычитание в беззнаковой форме;
- сложение и вычитание в прямом, обратном или дополнительном двоичном коде;
- умножение в беззнаковой или знаковой форме;
- деление в беззнаковой или знаковой форме без восстановления остатка или с восстановлением остатка.

Кроме того, операционный автомат позволяет реализовать следующие поразрядные логические операции:

- отрицание («НЕ»);
- конъюнкция («И»);
- дизъюнкция («ИЛИ»);
- неравнозначность (исключающее «ИЛИ»).

Из приведенного списка следует, что возможно реализация и других, более сложных арифметических и логических операций.

Результатом выполнения всех арифметических и логических операций, за исключением умножения, является 8-разрядное число в формате, совпадающим с форматом исходных данных. Результат умножения может быть представлен как в 8-, так и в 16-разрядном формате.

На рис. 1.1 представлена структура операционного автомата в том виде, в котором она выводится на экран компьютера программой, поддерживающей процесс разработки микропрограмм управления выполнением операций для АЛУ.

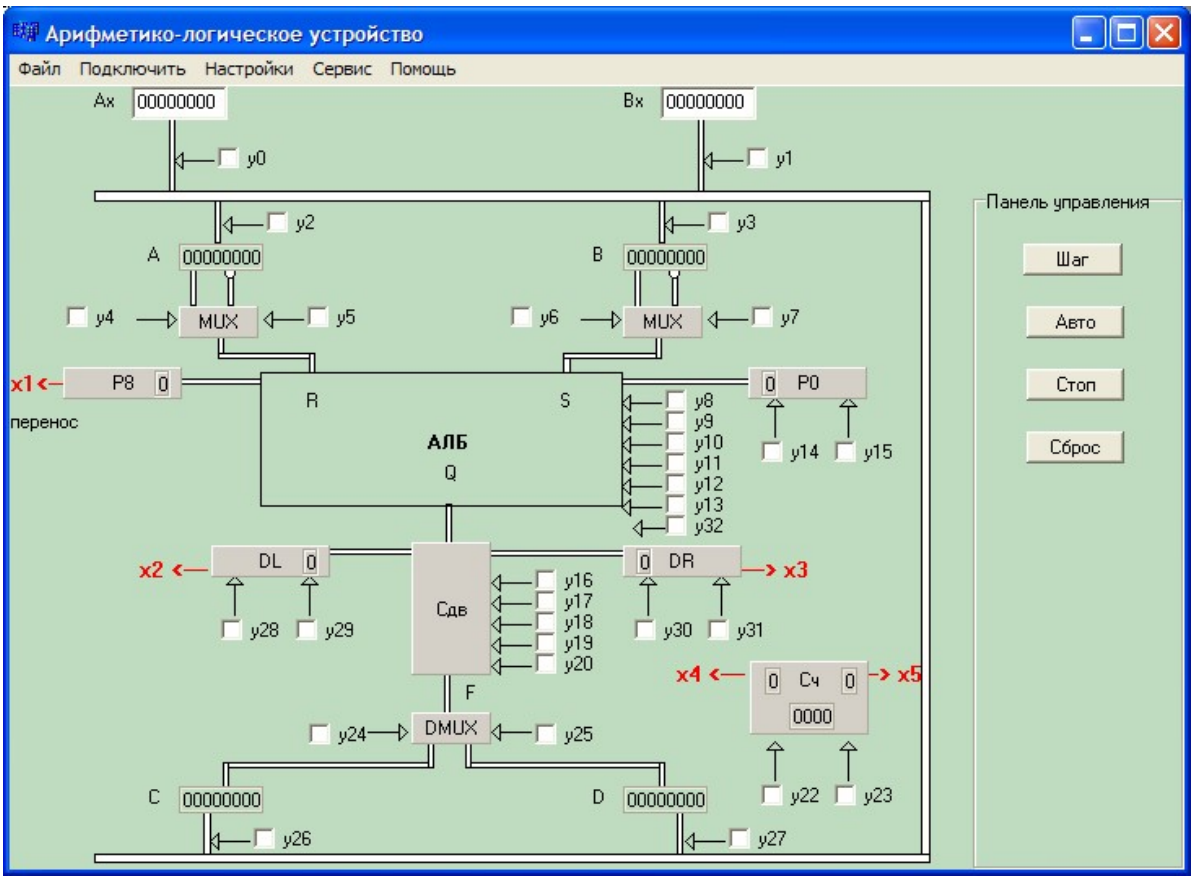


Рис. 1.1. Структура операционного автомата арифметико-логического устройства ALU-1

Операционный автомат содержит следующие элементы:

- входные 8-разрядные регистры `Ax` и `Bx`, допускающие загрузку исходных операндов из файла;
- 8-разрядные регистры `A` и `B` операндов;
- 8-разрядный арифметико-логический блок `АЛБ` с триггерами входного `P0` и выходного `P8` переносов (заемов);
- два 8-разрядных двухвходовых мультиплексора `MUX`, выбирающих для входов `R` и `S` `АЛБ` прямые или инверсные значения из регистров `A` и `B` соответственно;
- триггер `Z` (на рис. 1.1 явно не показан), значение которого формируется в соответствии с выражением

$$Z = P8 \oplus DL$$

и используется для формирования цифры частного в некоторых алгоритмах деления;

- блок сдвига, включающий 8-разрядный реверсивный комбинационный элемент Сдв, выполняющий сдвиг, и два триггера – DL и DR, формирующие или принимающие данные слева и справа соответственно;
- два 8-разрядных регистра результата C и D;
- демультиплексор DMUX, передающий значение 8-разрядного слова F с выхода блока сдвига в один из регистров результата (C или D);
- 4-разрядный двоичный счетчик Сч;
- 8-разрядную шину, на которую могут быть переданы слова из регистров Ax, Bx, C, D и с которой поступают в регистры A и/или B.

Регистры сами по себе никаких функциональных преобразований кодов не выполняют. Они являются лишь устройствами памяти, подключенными к общей шине. Входные регистры Ax, Bx и выходные регистры C, D могут находиться в одном из трех устойчивых состояний, которые позволяют подключать выходы регистра к шине или отключать их от шины. Если не принимать во внимание физические аспекты, то шину можно рассматривать как совокупность восьми переменных, каждая из которых может принимать одно из трех значений – «не активна», 0 или 1.

В рассматриваемой версии модели ALU-1 арифметико-логического устройства возможна лишь передача в рабочие регистры A, B содержимого регистров Ax, Bx, C, D. При этом за один такт можно выполнить передачу только из одного регистра-источника в регистр A, в регистр B или в оба регистра одновременно. Для этого выполняется одна из четырех микроопераций y0, y1, y26, y27, подключающая к шине выводы соответствующего регистра-источника, а также микрооперация y2 или y3 (или обе одновременно), передающая состояние шины на входы регистров-приемников (см. рис. 1.1).

Очевидно, что одновременное подключение к шине двух регистров-источников приводит к конфликту данных, поэтому оно является недопустимым. Одновременная же запись информации с шины в разные регистры-приемники, напротив, допустима. Однако такая необходимость редко встречается при реализации конкретных алгоритмов.

Арифметико-логический блок АЛБ содержит комбинаторную схему функционального преобразователя и два элемента памяти – триггеры P0 и P8. Состояние триггера P0 может быть задано произвольно с помощью микроопераций y14 и y15. В дальнейшем оно участвует в микрооперациях сложения (вычитания) в качестве значения переноса (заема) младшего разряда. Триггер P8 изменяется только микрокомандами арифметических и логических микроопераций АЛБ. При выполнении арифметических микроопераций он принимает значение переноса либо заема, возникающего из старшего разряда. При выполнении логических микроопераций триггер всегда устанавли-

вается в «0». Остальные микрооперации сохраняют ранее установленное значение P8.

Арифметико-логический блок реализует ряд бинарных операций над 8-разрядными двоичными словами R и S и битом P0. Результат операции записывается в 8-разрядное слово Q и бит P8. При этом значение P8 рассматривается как *логическое условие* x1, формируемое операционным автоматом. Кроме того, в блоке АЛБ формируется значение логического условия x6 = P8 ⊕ DL, для отображения которого необходимо выполнить команду меню **Настройки ► Z:=p8 (+) DL**.

Арифметико-логический блок реализует следующие бинарные микрооперации (y8, y9, y10, y11, y12, y13):

$$P8.Q = R + S + P0,\quad P8.Q = R - S - P0,$$
$$P8.Q = S - R - P0,\quad 0.Q = R \,\&\, S,$$
$$0.Q = R \vee S,\quad 0.Q = R \oplus S.$$

Кроме того, АЛБ реализует унарную микрооперацию P8.Q = P8.R (y32), в результате выполнения которой значение P8 не изменяется.

Блок сдвига содержит собственно элемент реверсивного сдвига Сдв на один разряд и два триггера данных – «слева» DL и «справа» DR. Каждый из этих триггеров может быть установлен с помощью микроопераций y28, y29, y30, y31 таким образом, чтобы он мог принять значения разрядов, теряемых при выполнении сдвига слова Q. Текущие значения триггеров DL и DR могут рассматриваться как значения логических условий x2 и x3 соответственно.

При сдвиге вправо младший разряд Q_0 сдвигаемого слова Q записывается в триггер DR, а в старший разряд F_7 выходного слова F помещается значение, предварительно установленное во триггере DL. При этом состояние самого триггера DL не изменяется. При сдвиге влево, наоборот, выполняются операции

$$DL = Q_7,\qquad F_0 = DR,\qquad DR = DR.$$

Элемент сдвига Сдв позволяет преобразовать входное слово Q в выходное слово F и без сдвига. При этом значения триггеров DL и DR не изменяются.

Таким образом, блок сдвига выполняет следующие микрооперации (y16, y17, y18, y19, y20):

- $F_{7:0} = Q_{7:0}$ – передача без сдвига;
- $F_{7:0}.DR = DL.Q_{7:0}$ – обыкновенный сдвиг вправо;
- $DL.F_{7:0} = Q_{7:0}.DR$ – обыкновенный сдвиг влево;

- $F_{7:0}$.DR = DR.$Q_{7:0}$ – специальный сдвиг вправо;
- $F_{7:0}$ = $Q_{6:0}$.HE(Z) – специальный сдвиг влево.

Специальный сдвиг вправо удобно использовать в микропрограмме умножения при формировании 16-разрядного произведения, а специальный сдвиг влево – в микропрограмме деления при формировании очередной цифры частного.

Четырехразрядный двоичный *счетчик* Сч реализует две микрооперации – y22 (начальная установка счетчика в состояние «0000») и y23 (инкремент). Он принимает последовательные значения от 0000 до 1000. Счетчик формирует два логических условия – x4 и x5. Логическое условие x4 истинно, если счетчик находится в состоянии 1000, а логическое условие x5 истинно, если состояние счетчика больше, чем 0011.

Назначение мультиплексоров MUX и демультиплексора DMUX очевидным образом следует из структурной схемы операционного автомата, представленной на рис. 1.1.

Полные списки микроопераций, реализуемых в операционном автомате ALU-1, и формируемых в нем логических условий приведены в табл. 1.1 и 1.2 соответственно.

Таблица 1.1

Микрооперации, реализуемые в операционном автомате ALU-1

Обозначение	Микрооперация	Описание микрооперации
y0	Shina = Ax	Подключение выходов регистра Ax к шине данных
y1	Shina = Bx	Подключение выходов регистра Bx к шине данных
y2	A = Shina	Запись в регистр A значения из регистра, подключенного к шине
y3	B = Shina	Запись в регистр B значения из регистра, подключенного к шине
y4	R = A	Подключение прямых выходов элементов регистра A к входу R АЛБ. При отсутствии микрокоманды подключения на вход R подаются нулевые значения
y5	R = HE(A)	Подключение инверсных выходов элементов регистра A к входу R АЛБ
y6	S = B	Подключение прямых выходов элементов регистра B к входу S АЛБ. При отсутствии микрокоманды подключения на вход S подаются нулевые значения
y7	S = HE(B)	Подключение инверсных выходов элементов регистра B к входу S АЛБ

продолжение табл. 1.1

Обозначение	Микрооперация	Описание микрооперации
y8	P8.Q = R + S + P0	Сложение чисел, находящихся на входах R, S, а также содержимого триггера P0 АЛБ. Содержимое триггера P0 добавляет 0 или 1 к младшему разряду суммы, а в триггере P8 формируется значение переноса
y9	P8.Q = R – S – P0	Вычитание из числа, находящегося на входе R, числа, находящегося на входе S, а также содержимого триггера P0 АЛБ. Содержимое триггера P0 вычитается из младшего разряда разности, а в триггере P8 формируется значение заема старшего разряда
y10	P8.Q = S – R – P0	Вычитание из числа, находящегося на входе S, числа, находящегося на входе R, а также содержимого триггера P0 АЛБ. Содержимое триггера P0 вычитается из младшего разряда разности, а в триггере P8 формируется значение заема старшего разряда
y11	0.Q = S & R	Поразрядная конъюнкция чисел, находящихся на входах R и S АЛБ. В триггер P8 помещается 0
y12	0.Q = S ∨ R	Поразрядная дизъюнкция чисел, находящихся на входах R и S АЛБ. В триггер P8 помещается 0
y13	0.Q = S ⊕ R	Поразрядная неравнозначность чисел, находящихся на входах R и S АЛБ. В триггер P8 помещается 0
y14	P0 = 0	Установка триггера P0 в состояние 0
y15	P0 = 1	Установка триггера P0 в состояние 1
y16	F = Q	Передача слова Q с выхода АЛБ на выход F схемы сдвига без изменений
y17	$F_{7:0}$.DR = DL.$Q_{7:0}$	Обыкновенный сдвиг слова Q с выхода АЛБ вправо. Младший разряд слова Q помещается в триггер DR. В старший разряд слова F помещается содержимое триггера DL. Содержимое триггера DL не изменяется
y18	DL.$F_{7:0}$ = $Q_{7:0}$.DR	Обыкновенный сдвиг слова Q с выхода АЛБ влево. Старший разряд слова Q помещается в триггер DL. В младший разряд слова F помещается содержимое триггера DR. Содержимое триггера DR не изменяется
y19	$F_{7:0}$.DR = DR.$Q_{7:0}$	Специальный сдвиг слова Q с выхода АЛБ вправо. Младший разряд слова Q помещается в триггер DR. В старший разряд слова F помещается предыдущее значение триггера DR
y20	$F_{7:0}$ = $Q_{6:0}$.HE(Z) Z = P8 ⊕ DL	Специальный сдвиг слова Q с выхода АЛБ влево. Старший разряд слова Q утрачивается. В младший разряд слова F помещается очередная цифра частного (в операциях деления). Содержимое триггера DL не изменяется
y21	Стоп	Останов. Сигнал об окончании выполнения операции

окончание табл. 1.1

y22	Сч = 0	Инициализация всех разрядов счетчика Сч нулевыми значениями. Установка триггеров признаков x4 и x5 в состояние 0
y23	Сч = Сч + 1	Увеличение счетчика на 1. При значении счетчика Сч = 1000 триггер признака x4 устанавливается в состояние 1. При значениях счетчика Сч, равных 0100, 0110, 0111 и 1000 триггер признака x5 устанавливается в состояние 1. Если текущее состояние счетчика Сч = 1000, то при увеличении на 1 он переходит в состояние 0000 и оба триггера признаков устанавливаются в состояние 0
y24	C = F	Помещение слова F на выходе схемы сдвига в регистр C
y25	D = F	Помещение слова F на выходе схемы сдвига в регистр D
y26	Shina = C	Подключение выходов регистра C к шине данных
y27	Shina = D	Подключение выходов регистра D к шине данных
y28	DL = 0	Установка триггера DL в состояние 0
y29	DL = 1	Установка триггера DL в состояние 1
y30	DR = 0	Установка триггера DR в состояние 0
y31	DR = 1	Установка триггера DR в состояние 1
y32	Q = R	Передача слова R с входа АЛБ на выход Q без изменений

Таблица 1.2

Логические условия, формируемые операционным автоматом ALU-1

Обозначение	**Логическое условие**	**Описание логического условия**
x1	P8	Признак переноса или заема
x2	DL	Состояние триггера DL
x3	DR	Состояние триггера DR
x4	Сч = 1000	Значение счетчика Сч равно 8
x5	Сч > 0011	Значение счетчика Сч больше 3
x6	Z = P8 ⊕ DL	Признак, по которому можно определить значение очередной цифры частного в операциях деления

Рассмотренная модель операционного автомата может работать в двух режимах – автономно или под управлением микропрограммы, являющейся составной частью модели управляющего автомата.

Автономный режим реализуется только пошагово и позволяет проверить правильность разработанных алгоритмов арифметических и логических преобразований на ряде конкретных примеров.

1.1.2. Управляющий автомат арифметико-логического устройства ALU-1 с программируемой логикой

Предлагаемая модель управляющего автомата ориентирована, прежде всего, на управление операционным автоматом, описанным в разд. 1.1.1. Однако с помощью этой модели можно реализовать и любую «абстрактную» микропрограмму с учетом приведенных далее ограничений.

Управляющий автомат с программируемой логикой реализован как автомат с естественной адресацией и смешанным способом кодирования поля микроопераций и использует единственный формат микрокоманды. Структурная схема моделируемого управляющего автомата аналогична схеме, приведенной на рис. ??. Отличия состоят лишь в количестве микроопераций и их полей, а также в количестве логических условий, поступающих из операционного автомата.

Количественные характеристики управляющего автомата и формат микрокоманды выбирались, прежде всего, исходя из потребностей управления операционным автоматом арифметико-логического устройства ALU-1. Анализ структуры операционного автомата и реализуемых в ней микропрограмм показывает, что можно ограничиться пятью микрооперациями в микрокоманде. Действительно, глядя на рис. 1.1, можно отметить, что для выполнения любой бинарной микрокоманды (например, для арифметического сложения двоичных чисел) необходимо одновременно выполнить следующие операции:

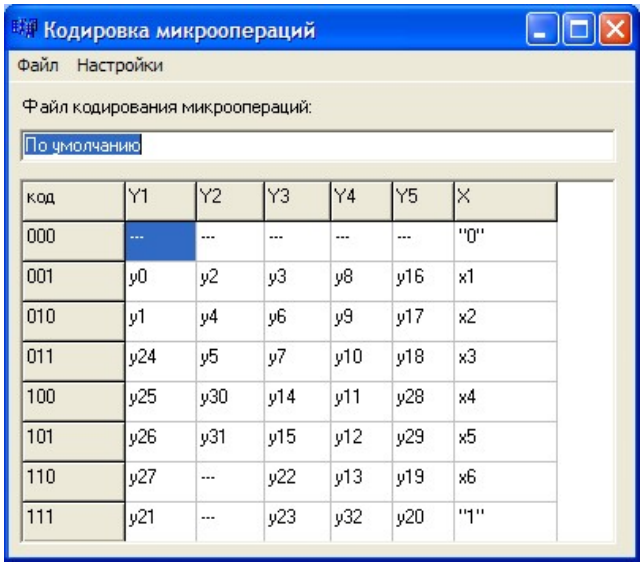
1) выбрать первый операнд `R`;
2) выбрать второй операнд `S`;
3) указать операцию, выполняемую арифметико-логическим блоком `АЛБ`;
4) указать операцию, выполняемую в блоке сдвига;
5) определить регистр, в который следует поместить результат.

Таким образом, все множество $Y = \{y_0, y_1, \ldots, y_{32}\}$ следует разбить на пять подмножеств (см. разд. ??) с учетом анализа типичных микропрограмм арифметических и логических операций. Такая работа была проделана при проектировании модели управляющего автомата, и ее результат предлагается в виде кодировки микроопераций `По умолчанию` (рис. 1.2).

Окно, изображенное на рис. 1.2, можно открыть, выполнив из меню окна **Арифметико-логическое устройство** команду **Подключить ► Управляющий автомат**. В результате выполнения откроется одноименное окно, из меню которого нужно выполнить команду **Кодировка микроопераций**.

Практика показывает, что большинство алгоритмов арифметических и логических преобразований можно реализовать в структуре операционного

автомата арифметико-логического устройства ALU-1 с предложенным разбиением и кодированием микроопераций. Однако пользователь может произвольно изменять разбиение микроопераций на подмножества и кодирование этих подмножеств (микрокоманд). Для этого нужно из меню окна **Кодировка микроопераций** выполнить команду **Настройки ► Разрешить правку**. Однако при этом нельзя увеличивать количество подмножеств, а число микроопераций в одном подмножестве не может быть больше семи. Данные ограничения связаны с жестко определенным форматом микрокоманды.



код	Y1	Y2	Y3	Y4	Y5	X
000	---	---	---	---	---	"0"
001	y0	y2	y3	y8	y16	x1
010	y1	y4	y6	y9	y17	x2
011	y24	y5	y7	y10	y18	x3
100	y25	y30	y14	y11	y28	x4
101	y26	y31	y15	y12	y29	x5
110	y27	---	y22	y13	y19	x6
111	y21	---	y23	y32	y20	"1"

Рис. 1.2. Кодирование микроопераций и логических условий по умолчанию

Формат микрокоманды (рис. 1.3) включает:

- пять трехразрядных полей Y_1, Y_2, Y_3, Y_4, Y_5 кодирования микроопераций;
- трехразрядное поле X номера логического условия (операционный автомат генерирует шесть логических условий, а еще два кода используются для кодирования констант 0 и 1);
- одноразрядное поле i инверсии логического условия;
- семиразрядное поле адреса перехода.

Очевидно, что максимальная длина микропрограммы в этом случае может составлять $2^7 = 128$ микрокоманд.

Y_1	Y_2	Y_3	Y_4	Y_5	X	i	Адрес перехода
3	3	3	3	3	3	1	7

Рис. 1.3. Формат микрокоманды

В разд. ?? приведен пример проектирования управляющего автомата с программируемой логикой по исходной микропрограмме. Процесс проектирования включает следующие этапы:

- определение формата микрокоманды (разбиение множества микроопераций на подмножества, кодирование микроопераций и логических условий);

- формирование последовательности управляющих слов в постоянном запоминающем устройстве микрокоманд, реализующих заданную микропрограмму.

Очевидно, что если для построения управляющего автомата с программируемой логикой использовать программную модель ALU-1, то первый этап выполнять не придется при условии, что предложенное «по умолчанию» разбиение позволит реализовать заданную микропрограмму.

Второй этап можно выполнить «вручную» или с помощью встроенного микроассемблера. Работа «вручную» весьма трудоемка и приводит к значительному числу ошибок. Однако на начальном этапе небольшую микропрограмму следует выполнить именно этим способом, чтобы получить полное представление о структуре микрокомандных слов и способах переходов по микропрограмме. Процесс «ручного» микропрограммирования подробно описан в примере из разд. ??.

Использование микроассемблера позволяет значительно увеличить скорость и надежность подготовки микропрограмм. Для работы с микроассемблером необходимо из меню окна **Управляющий автомат** выполнить команду **Компилятор**. В результате выполнения откроется окно **Компиляция кода** текстового редактора, и в этом окне и следует подготовить текст микропрограммы, состоящий из последовательности микрокоманд.

Формат микрокоманды (см. рис. 1.3) определяет структуру записи на языке микроассемблера. Определим элементы записи следующим образом:

<строка> ::= [<метка>][<микрокоманда>][#<комментарий>]

<микрокоманда> ::= [<последовательность микроопераций>] [; <логическое условие>, <инверсия логического условия>, <метка>]

[<последовательность микроопераций>] ::= <микрооперация> [, <микрооперация>, <микрооперация> …] (не более пяти)

<микрооперация> ∈ {y_i}, i ∈ {0, 1, 2, …, 32}

<логическое условие> ∈ {x_j, 0, 1}, j ∈ {1, 2, 3, …, 6}

<инверсия логического условия> ∈ {0, 1}

<метка> ::= <идентификатор>:

При разработке микропрограммы для последующей компиляции в среде модели ALU-1 арифметико-логического устройства необходимо соблюдать следующие правила:

- в одной строке должно находиться не более одной микрокоманды (возможно, помеченной);
- идентификаторы меток перехода, микроопераций и логических условий должны существовать;
- идентификаторы меток не должны дублироваться.

Для разделения полей в микропрограмме используются следующие знаки:

- «`,`» (запятая) – для разделения микроопераций в последовательности микроопераций;
- «`;`» (точка с запятой) – для отделения поля переадресации от поля микроопераций;
- «`:`» (двоеточие) – для отделения идентификатора метки от следующей за ней микрокоманды;
- «#» (хэш) – для отделения комментария от микрокоманды, причем комментарий распространяется от символа «хэш» до конца строки;
- «+n» – директива сдвига команды на `n` адресов (`n` задается десятичным целым числом).

Примерами микрокоманд могут являться следующие строки:

```
y26, y2 # передача слова из регистра C в регистр A
y4, y6, y8, y16, y25 # алгебраическое сложение слов
     # из регистров A и B и передача полученной суммы
     # в регистр D без сдвига
y28; 1, 0, L2 # сброс триггера DL и безусловный переход
     # на метку L2 (в качестве логического условия
     # используется константа 1)
y28; x1, 0, L2 # сброс триггера DL и переход на метку
     L2 при условии x1 = 1
y4, y32, y18; x2, 1, L3 # сдвиг слова из регистра A
     влево и переход на метку L3 при x2 = 0 (поскольку
     флаг инверсии логического условия равен 1)
L3:  y21 # микрокоманда завершения микропрограммы,
     помеченная идентификатором L3
```

Если микрокоманда содержит только последовательность микроопераций, то компилятор по умолчанию добавляет поля переадресации `; 000, 0, 0000000`, что соответствует тождественно ложному условию без инверсии, т. е. выполняется переход по адресу, который на единицу больше текущего.

При работе операционного автомата под управлением автомата с программируемой логикой необходимо предварительно загрузить в память модели управляющего автомата код микропрограммы. Проиллюстрируем эти действия на примере алгоритма сложения двоичных слов в прямом коде. В табл. ?? каждая строка определяет действие, выполняемое операционным автоматом *или* выбор адреса следующей микрокоманды. Формат микрокоманды управляющего автомата (см. рис. 1.3) содержит как поля микроопераций, так и поля переадресации. Поэтому в каждой микрокоманде можно задать действие операционного автомата *и* выбрать адрес следующей микрокоманды. Таким образом, *две* последовательные строки табл. ??, первая из которых определяет микрооперации, а вторая – переход, реализуются с помощью одной микрокоманды управляющего автомата. Если же после одной операционной микрокоманды следует другая, то на нее (на вторую) осуществляется безусловный переход.

Учитывая эти правила и приведенный выше синтаксис, можно записать текст микропрограммы для сложения двоичных слов в прямом коде на микроассемблере (рис. 1.4).

После успешной компиляции микропрограмма будет готова к исполнению, а память микропрограмм управляющего автомата будет иметь вид, представленный на рис. 1.5.

После этого можно перейти в окно операционного автомата, выполнить из его меню команду **Настройки ► Управление ► Управляющий автомат** и выполнять микропрограмму сложения с различными операндами в пошаговом или в автоматическом режиме.

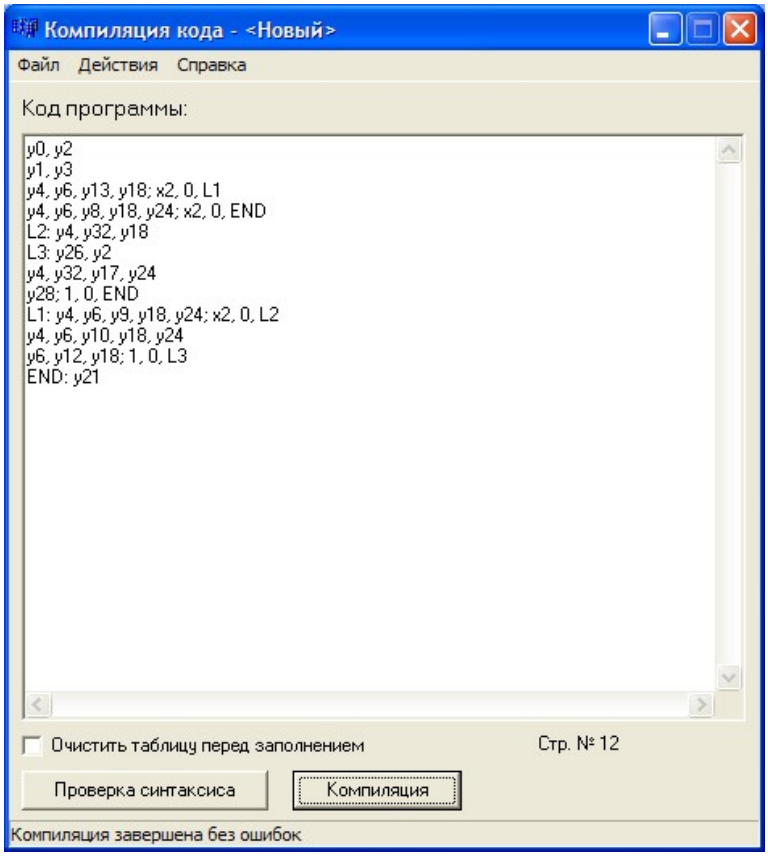


Рис. 1.4. Текст микропрограммы
сложения двоичных слов в прямом коде

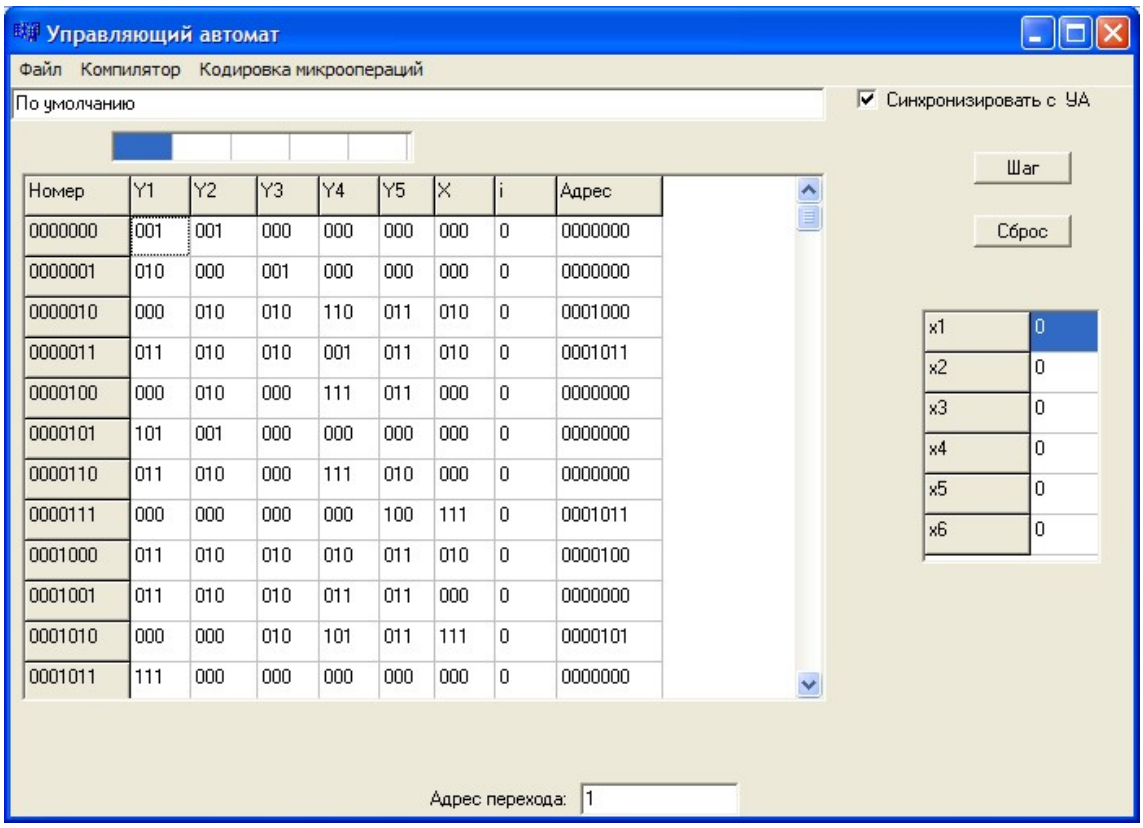


Рис. 1.5. Код микропрограммы сложения двоичных слов в прямом коде

Лабораторная работа № 1

РАЗРАБОТКА АЛГОРИТМА И МИКРОПРОГРАММЫ АРИФМЕТИЧЕСКОЙ ОПЕРАЦИИ

Задание 1. Разработать алгоритм сложения 8-разрядных двоичных целых чисел из табл. 1.1 и реализовать его в виде микропрограммы для операционных автоматов арифметико-логических устройств ALU-1 и ALU-R.

Внимание! Перед выполнением задания следует ознакомиться с теоретическим материалом, изложенным в разделах 3.3 – 3.6.

Таблица 1.1

Вариант	Коды представления				Примеры чисел	
	операнда *A*	операнда *B*	операции	результата	A_{10}	B_{10}
1	прямой	прямой	прямой	прямой	97	17
2	прямой	прямой	прямой	обратный	83	36
3	прямой	прямой	прямой	дополнительный	80	28
4	прямой	прямой	обратный	прямой	73	27
5	прямой	прямой	обратный	обратный	37	15
6	прямой	прямой	обратный	дополнительный	56	20
7	прямой	прямой	дополнительный	прямой	66	14
8	прямой	прямой	дополнительный	обратный	61	48
9	прямой	прямой	дополнительный	дополнительный	98	25
10	прямой	обратный	прямой	прямой	70	42
11	прямой	обратный	прямой	обратный	41	30
12	прямой	обратный	прямой	дополнительный	44	62
13	прямой	обратный	обратный	прямой	40	38
14	прямой	обратный	обратный	обратный	87	34
15	прямой	обратный	обратный	дополнительный	53	39
16	прямой	обратный	дополнительный	прямой	22	91
17	прямой	обратный	дополнительный	обратный	67	32
18	прямой	обратный	дополнительный	дополнительный	24	45
19	прямой	дополнительный	прямой	прямой	60	54
20	прямой	дополнительный	прямой	обратный	84	21
21	прямой	дополнительный	прямой	дополнительный	85	31
22	прямой	дополнительный	обратный	прямой	77	12
23	прямой	дополнительный	обратный	обратный	72	51
24	прямой	дополнительный	обратный	дополнительный	86	10
25	прямой	дополнительный	дополнительный	прямой	58	52
26	прямой	дополнительный	дополнительный	обратный	92	29
27	прямой	дополнительный	дополнительный	дополнительный	81	46
28	обратный	прямой	прямой	прямой	76	47
29	обратный	прямой	прямой	обратный	90	26
30	обратный	прямой	прямой	дополнительный	93	33
31	обратный	прямой	обратный	прямой	64	55
32	обратный	прямой	обратный	обратный	65	43
33	обратный	прямой	обратный	дополнительный	88	11
34	обратный	прямой	дополнительный	прямой	99	18
35	обратный	прямой	дополнительный	обратный	68	19
36	обратный	прямой	дополнительный	дополнительный	82	13

37	обратный	обратный	прямой	прямой	74	49
38	обратный	обратный	прямой	обратный	79	35
39	обратный	обратный	прямой	дополнительный	69	57
40	обратный	обратный	обратный	прямой	75	23
41	обратный	обратный	обратный	обратный	94	16
42	обратный	обратный	обратный	дополнительный	71	50
43	обратный	обратный	дополнительный	прямой	59	63
44	обратный	обратный	дополнительный	обратный	78	41
45	обратный	обратный	дополнительный	дополнительный	95	16
46	обратный	дополнительный	прямой	прямой	89	15
47	обратный	дополнительный	прямой	обратный	96	10
48	обратный	дополнительный	прямой	дополнительный	78	47
49	обратный	дополнительный	обратный	прямой	70	57
50	обратный	дополнительный	обратный	обратный	73	32
51	обратный	дополнительный	обратный	дополнительный	67	60
52	обратный	дополнительный	дополнительный	прямой	79	44
53	обратный	дополнительный	дополнительный	обратный	81	37
54	обратный	дополнительный	дополнительный	дополнительный	88	26
55	дополнительный	прямой	прямой	прямой	75	50
56	дополнительный	прямой	прямой	обратный	80	17
57	дополнительный	прямой	прямой	дополнительный	98	23
58	дополнительный	прямой	обратный	прямой	96	22
59	дополнительный	прямой	обратный	обратный	65	56
60	дополнительный	прямой	обратный	дополнительный	89	11
61	дополнительный	прямой	дополнительный	прямой	68	55
62	дополнительный	прямой	дополнительный	обратный	94	19
63	дополнительный	прямой	дополнительный	дополнительный	76	34
64	дополнительный	обратный	прямой	прямой	84	40
65	дополнительный	обратный	прямой	обратный	77	25
66	дополнительный	обратный	прямой	дополнительный	82	30
67	дополнительный	обратный	обратный	прямой	74	28
68	дополнительный	обратный	обратный	обратный	85	13
69	дополнительный	обратный	обратный	дополнительный	59	52
70	дополнительный	обратный	дополнительный	прямой	83	27
71	дополнительный	обратный	дополнительный	обратный	58	61
72	дополнительный	обратный	дополнительный	дополнительный	53	62
73	дополнительный	дополнительный	прямой	прямой	91	29
74	дополнительный	дополнительный	прямой	обратный	99	20
75	дополнительный	дополнительный	прямой	дополнительный	97	14
76	дополнительный	дополнительный	обратный	прямой	46	54
77	дополнительный	дополнительный	обратный	обратный	12	33
78	дополнительный	дополнительный	обратный	дополнительный	95	31
79	дополнительный	дополнительный	дополнительный	прямой	43	72
80	дополнительный	дополнительный	дополнительный	обратный	69	51
81	дополнительный	дополнительный	дополнительный	дополнительный	86	36