МПСвЭПиТК

Программирование в виде релейно-контактных схем

Основной алгоритм

1. Сделайте список всех входных/выходных устройств и точек входов/выходов, которые им присвоены, и приготовьте таблицу, в которой показаны эти присвоения.

2. Если Вы применяете биты LR для связи двух ПК, приготовьте таблицу, в которой будет показано применение данных битов.

3. Определите, какие слова доступны в качестве рабочих, и приготовьте таблицу, в которой показано их распределение.

4. Также приготовьте таблицы номеров ТС и номеров переходов, чтобы распределить их в соответствии с планом использования. Не забывайте, что функцию номера ТС можно задать только один раз в программе; Номера переходов 01..99 можно использовать один раз каждый. (Номер ТС описан в п. 5-15, номера переходов в данной главе далее).

5. Нарисуйте релейно-контактную схему.

6. Введите программу в ЦУ. При использовании программатора это повлечет преобразование программы в мнемоническую форму.

7. Проверьте программу на синтаксические ошибки и исправьте их.

8. Выполните программу для проверки на ошибки исполнения и исправьте их.

9. После установки всей системы и готовности ее к работе выполните программу и в случае необходимости произведите точную настройку.

Распределение входов/выходов

Наименование системы		Разработал	Проверил	Утвердил
Модель ПК	Лист N			

IR____	Блок №	Модель	IR____	Блок №	Модель
00			00		
01			01		
02			02		
03			03		
04			04		
05			05		
06			06		
07			07		
08			08		
09			09		
10			10		
11			11		
12			12		
13			13		
14			14		
15			15		

Терминология команд

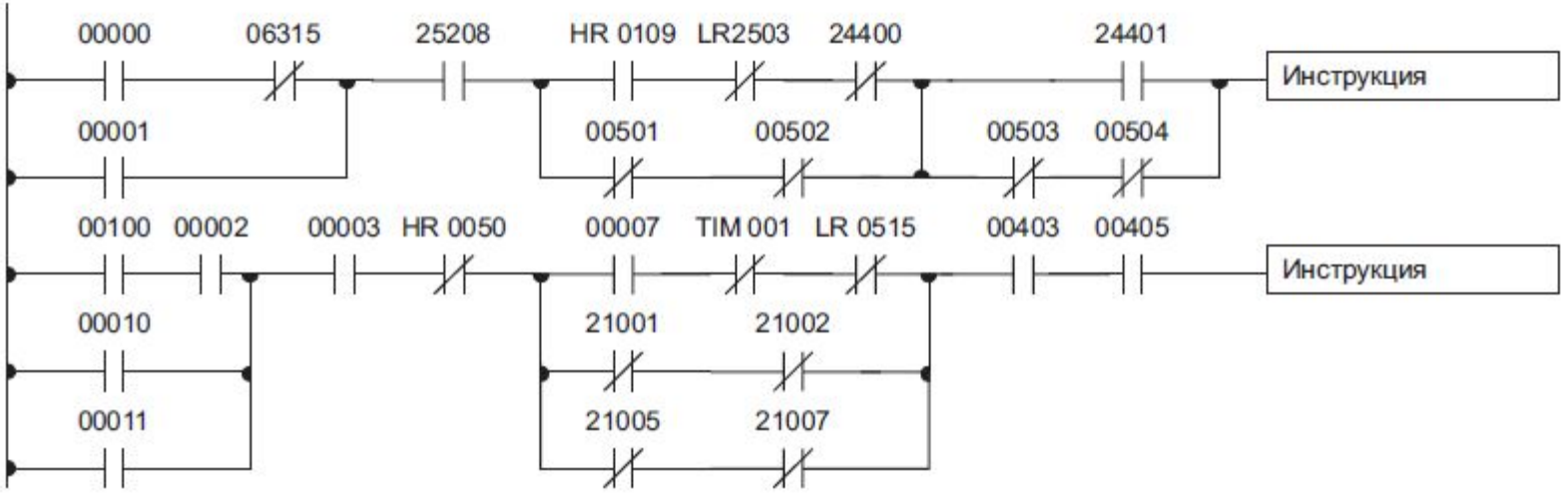
Имеется два типа команд при программировании в виде релейно-контактных схем:

- команды РКС - команды, которые соответствуют условиям на релейно-контактной схеме и используются в форме команд только при преобразовании программы в мнемокод;
- команды, которые располагаются с правой стороны релейно-контактной схеме и выполняются согласно условиям на командных линиях, ведущих к ним.

Большинство команд имеет минимум один или больше связанных с ними операндов. Операнды указывают или содержат данные, над которыми должна исполняться команда. Операнды иногда введены, как численные значения, но обычно представляют собой адреса слов или битов, которые содержат данные, которые будут использоваться. Например, команда MOVE, имеющая в качестве операнда источника IR 000, переместит содержимое IR 000 в заданное место. Данное место тоже задается как операнд. Бит, адрес которого задан как операнд, называется битовый операнд; слово, адрес которого задан как операнд, называется словный операнд; Если текущее значение введено ка константа, ей предшествует # для указания того, что это не адрес.

Основные релейно-контактные схемы

Релейно-контактная схема состоит из одной вертикальной линии с левой стороны с линиями, отходящими направо. Вертикальная линия слева называется шиной, ответвление - командной линией или ступенькой. На командной линии располагаются условия, ведущие к командам на правой стороне. Логические комбинации этих условий определяют, когда и как выполняются “правосторонние” команды. На рисунке приведен пример релейно-контактной схемы.



Как показано на схеме, командные линии могут разветвляться и снова соединяться. Две рядом расположенные вертикальные линии называются условием. Условия без диагональной черты называются нормально открытыми условиями и соответствуют командам LOAD, AND или OR. Условия с диагональной чертой называются нормально закрытыми условиями и соответствует командам LOAD NOT, AND NOT или OR NOT. Число над каждым условием указывает битовый операнд для команды. Это состояние бита определяет условие исполнения следующих команд. Как исполняется каждая команда в соответствии с условиями, описано далее. Перед рассмотрением этого, однако, следует объяснить некоторые базовые термины.

Нормально открытые и нормально закрытые условия

Каждое условие на релейно-контактной схеме может находится в состоянии либо 1, либо 0 в зависимости от состояния битового операнда, связанного с ним. Нормально открытое условие = 1, если битовый операнд = 1 и = 0, если битовый операнд =0. Нормально закрытое условие = 1, если битовый операнд = 0 и = 0, если битовый операнд =1. Короче говоря, Вы используете нормально открытое условие, если желаете выполнения команд, когда бит = 1, и нормально закрытое условие, если желаете выполнения команд, когда бит = 0.

Условия исполнения

В программировании методом релейно-контактных схем логическая комбинация условий 1 и 0 перед командой определяет общее условие, при котором команда выполняется. Данное общее условие (либо 0, либо 1) называется условием исполнения команды. Все команды, за исключением команд LOAD, имеют условия исполнения.

Битовые операнды

Операндами для любой команды релейно-контактной схемы могут быть любые биты в областях IR, SR, HR, AR, LR или TC. Это значит, что условия на релейно-контактной схеме могут определяться битами входов/выходов, флагами, рабочими битами, таймерами, счетчиками и т. д. Команды LOAD и OUTPUT могут также использовать биты области TR, но только при специальных применениях.

Логические блоки

Условие исполнения команды определяется отношением между условиями внутри командных строк, которые их соединяют. Любая группа условий, работающих совместно для выработки логического результата, называется логическим блоком. Хотя релейно-контактную схему можно написать без анализа отдельных логических блоков, понимание работы логических блоков необходимо для эффективного программирования и существенно в случае, когда программы должны вводиться в виде мнемокода.

Блок команд

Блок команд состоит из всех команд, которые соединяются между собой, пересекая РКС. Таким образом, один блок команд состоит из всех команд от того места, где Вы можете провести горизонтальную линию через релейно-контактную схему, не пересекая ни одной вертикальной линии, до следующего места, где Вы можете провести другую такую линию.

Мнемокод

Релейно-контактную схему нельзя непосредственно ввести в ПК с программатора, требуется SSS. Для ввода с программатора необходимо преобразовать релейно-контактную схему в мнемокод. Мнемокод отображает ту же информацию, что и РКС, но в форме, которую можно ввести прямо в ПК. Фактически Вы можете программировать непосредственно в мнемокодах, но начинающим, или при сложных программах, это не рекомендуется. Итак, независимо от того, ли какое программирующее устройство используется, программа загружается в ПК в мнемокоде. Поэтом важно понимать и мнемокод.

Из-за важности программатора как периферийного устройства и важности мнемокода в понимании программы, наряду с РКС мы вводим и описываем мнемокоды. Помните, что при вводе с SSS знание мнемокода не требуется (хотя Вы, если предпочитаете, можете использовать его с SSS).

Структура памяти программ

Программа вводится по адресам памяти программ. Адреса в памяти программ несколько отличаются от адресов в других типов памяти, поскольку каждый адрес необязательно содержит одинаковый объем данных. Точнее, каждый адрес содержит одну команду и все определители и операнды (подробно описанные далее), требуемые для команды. Поскольку некоторые команды не требуют операндов, в то время как другие требуют до трех операндов, адреса памяти программ могут быть длиной 1..4 слова.

Адреса памяти программ начинаются с 00000 и продолжаются до тех пор, пока не будет занят весь объем памяти программ. Первое слово каждого адреса задает команду. Любые определители, используемые командой, также содержатся в первом слове. Кроме того, если команда требует только один битовый операнд (без определителя), битовый операнд также программируется в той же строке, что и команда. Остальные слова, требуемые командой, содержат операнды, которые определяют используемые данные. При преобразовании в мнемокод все команды, кроме РКС, записываются в такой же форме, одно слово в строке, как они появляются в символах РКС. Пример мнемокодов приведен в таблице. Используемые команды описаны далее в инструкции.

Адрес	Инструкция	Операнд
00000	LD	HR 0001
00001	AND	00001
00002	OR	00002
00003	LD NOT	00100
00004	AND	00101
00005	AND LD	00102
00006	MOV (21)	
		000
		DM 0000
00007	CMP (20)	
		DM 0000
		HR 00
00008	LD	25505
00009	OUT	10000
00010	MOV (21)	
		DM 0000
		DM 0500
00011	DIFU (13)	00502
00012	AND	00005
00013	OUT	10003

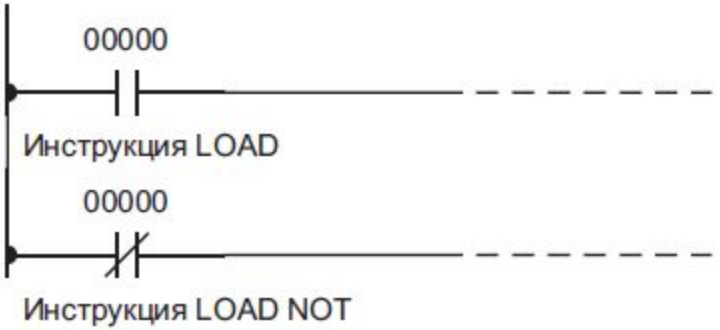
Столбцы адрес и команда таблицы мнемокодов заполнен только для слов команд. Для всех других строк два левые столбца остаются незаполненными. Если команда не требует определителя или битового операнда, столбец операнд остается пустым для первой строки. Рекомендуем зачеркивать все незанятые ячейки столбцов (для всех слов команд, не требующих данных), чтобы быстро просмотреть столбец для проверки, не опущен ли какой-либо адрес.

Команды, расположенные на релейно-контактной схеме (РКС)

Команды РКС - такие команды, которые соответствуют условиям на РКС. Команды РКС, либо независимо, либо в комбинации с блоковыми командами, описанными далее, образуют условия исполнения, в зависимости от которых исполняются все другие команды.

LOAD и LOAD NOT

Первым условием, которым начинается любой логический блок РКС, являются команды LOAD и LOAD NOT. Каждая команда требует одной строки в мнемокоде. В следующем примере “Команда” может быть любой “правосторонней” командой, , которые описаны далее в данной инструкции.



Адрес	Инструкция	Операнд
00000	LD	00000
00001	Инструкция	
00002	LD NOT	00000
00003	Инструкция	

Когда данное условие является единственным в командной строке, условие исполнения “правосторонней” команды = 1 при данном условии = 1. Для команды LOAD (нормально открытого условия) условие исполнения = 1 при IR 00000 = 1. Для команды LOAD NOT (нормально закрытого условия) условие исполнения = 1 при IR 00000 = 0.

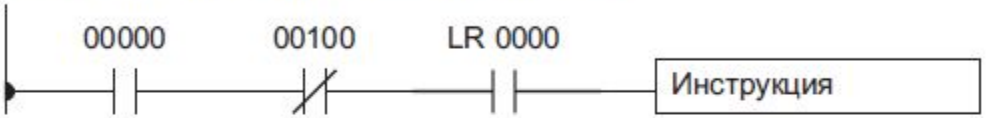
Вышеприведенную РКС нельзя преобразовать в мнемокод, используя только команды AND и OR. Если пытаться произвести ЛОГИЧЕСКОЕ И между IR 00002 и результатом ИЛИ между IR 00000 и IR 00001, теряется ИЛИ НЕ между IR 00002 и IR 00003 и ИЛИ НЕ прекращается, будучи ИЛИ НЕ между IR 00003 и результатом И между IR 00002 и первым ИЛИ. Нам же необходим метод произвести ИЛИ (НЕТ) независимо и далее объединить результаты.

Для выполнения данной задачи мы можем пользоваться командой LOAD или LOAD NOT в середине командной строки. При выполнении LOAD или LOAD NOT таким образом, текущее значение результатов операций сохраняется в специальных буферах и логический процесс начинается снова. Для объединения результатов текущих условий исполнения с ранее “неиспользованными” условиями исполнения используется команда AND LOAD или OR LOAD. Здесь LOAD подразумевает загрузку последнего неиспользованного условия исполнения. Неиспользованное условие исполнения производится использованием команд LOAD или LOAD NOT для любого, кроме первого, условия в командной линии.

Анализируя приведенную РКС с точки зрения мнемокоманд, условием перед IR 00000 является команда LOAD, а условием после нее - ИЛИ между состоянием IR 00000 и IR 00001. Условием на IR 00002 является другая команда LOAD, а условием после нее является команда ИЛИ НЕТ, т.е. ИЛИ между состоянием IR 00002 и инверсией IR 00003. Для получения условия исполнения “правосторонней” команды нужно проделать операцию И над этими двумя блоками. Это делает команда AND LOAD. Мнемокод для

AND и AND NOT

Когда два или более условий расположены последовательно в одной командной линии, первое условие соответствует команде LOAD или LOAD NOT. Остальные условия командам AND и AND NOT. В следующем примере показаны три условия, расположенные последовательно: LOAD, AND и AND NOT . Каждое из этих условий требует одной строки в мнемокоде.



Адрес	Инструкция	Операнд
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 0000
00003	Инструкция	

Условие исполнения команды будет = 1 только когда все три условия = 1, т.е. IR 0000 = 1, IR 00100 = 0, LR 0000 = 1.

Последовательно расположенные команды AND можно рассматривать индивидуально, каждая команда выполняет ЛОГИЧЕСКОЕ И над своим условием исполнения (т.е. результат всех условий до данной точки) и состоянием битового операнда самой команды AND. Если оба этих условия = 1, то условие исполнения для следующей команды будет = 1. Если хотя бы одно из этих условий = 0, то условие исполнение следующей команды будет = 0. Условие исполнения первой команды AND при последовательном соединении является первым условием командной линии.

Каждая команда AND NOT в последовательном соединении выполняет ЛОГИЧЕСКОЕ И над своим условием исполнения (т.е. результатом всех условий до данной точки) и инверсией своего битового операнда.

OR и OR NOT

Когда два или более условий расположены на разных командных линиях, идущих параллельно, и затем соединяющихся, первое условие соответствует команде LOAD или LOAD NOT. Остальные условия соответствуют командам OR и OR NOT. В примере показаны три условия, расположенные по порядку сверху: LOAD NOT, OR NOT и OR. Каждое из этих команд также требует одну строку в мнемокоде.

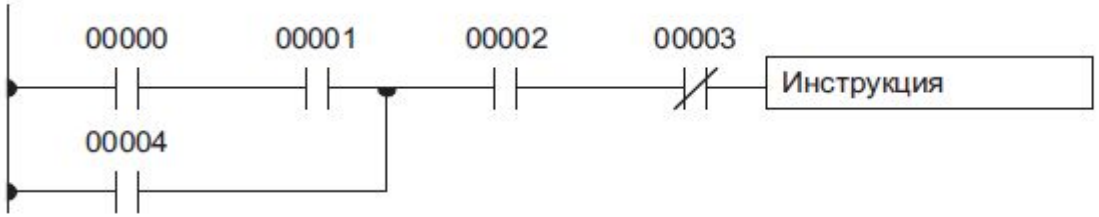
Адрес	Инструкция	Операнд
00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Инструкция	

Условие исполнения для “правосторонней” команды будет = 1, когда хотя бы одно из условий = 1, т.е. когда IR 0000 = 0, IR 00100 = 0, или LR 0000 = 1.

Команды OR и OR NOT можно рассматривать индивидуально, каждая команда выполняет ЛОГИЧЕСКОЕ ИЛИ над своим условием исполнения и состоянием битового операнда самой команды OR. Если хотя бы одно из этих условий = 1, то результат исполнения следующей команды будет = 1.

Сочетание команд AND и OR

Когда команды AND и OR объединяются в сложные схемы, их иногда можно рассматривать индивидуально, каждая команда выполняет логическую операцию с предыдущим условием и состоянием битового операнда. Далее приведен пример такой ситуации. Изучите данный пример, пока не поймете, что мнемокод следует той же логике, что и РКС.



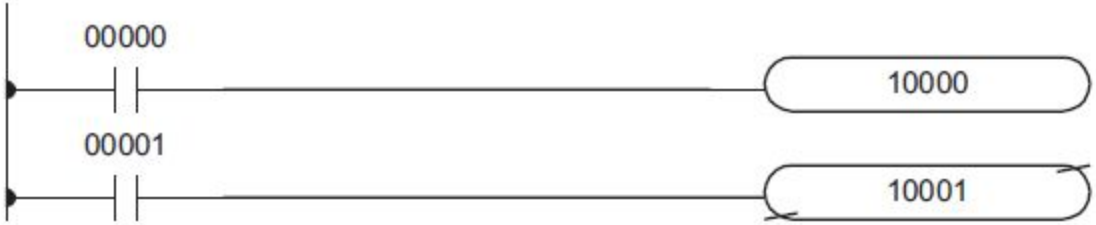
Адрес	Инструкция	Операнд
00000	LD	00000
00001	AND	00001
00002	OR	00200
00003	AND	00002
00004	AND NOT	00003
00005	Инструкция	

В данном примере выполняется ЛОГИЧЕСКОЕ И между состоянием IR 0000 и IR 00001 для определения условия исполнения для ИЛИ с состоянием IR 0200. Результат данной операции определит условие исполнения для И с состоянием IR 00002, которое, в свою очередь, определит условие исполнения для И с инверсией (AND NOT) с состоянием IR 00003.

Однако в более сложных РКС необходимо рассматривать логические блоки перед определением условий исполнения для конечной команды, и именно здесь используются команды AND LOAD и OR LOAD. Перед рассмотрением более сложных схем, мы рассмотрим команды, которые требуются для завершения простейшей программы ”ввод/вывод”.

Ввод и Вывод инверсии (OUTPUT и OUTPUT NOT)

Простейший способ выдать результат комбинаций условий исполнения - прямая выдача командами OUTPUT и OUTPUT NOT. Данные команды служат для управления состоянием заданного битового операнда в соответствии со своим условием исполнения. Командой OUTPUT битовому операнду будет присваиваться значение 1, пока условие исполнения = 1 и 0, пока условие исполнения = 0. Командой OUTPUT NOT битовому операнду будет присваиваться значение 1, пока условие исполнения = 0 и 0, пока условие исполнения = 1. Это происходит в соответствии с схемой. В мнемокоде каждая из этих команд требует одной строки.

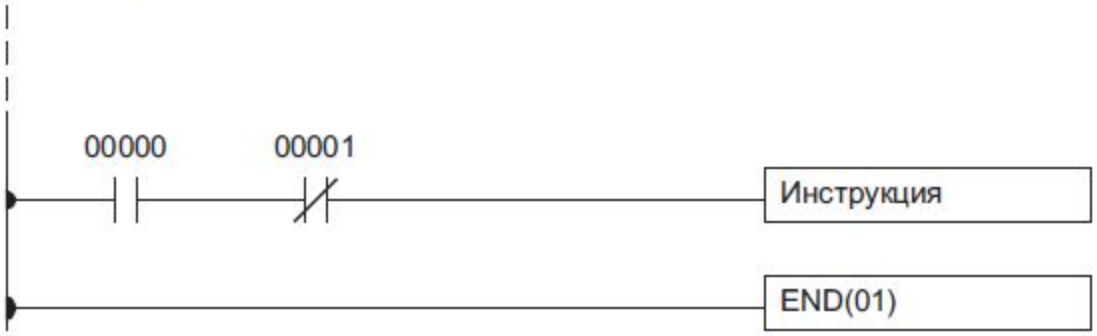


Адрес	Инструкция	Операнд
00000	LD	00000
00001	OUT	10000
00002	LD	00001
00003	OUT NOT	10001

В данных примерах IR 10000 будет в положении 1, пока IR 00000 = 1, а IR 10001 будет = 0, пока IR 00001 = 1. В данном примере IR 00000 и IR 00001 - входные биты, а IR 10000 и IR 100001 - выходные биты, приписанные к блокам ПК. Т.е. сигналы, приходящие с входов, приписанных к IR 00000 и IR 00001, управляют выходами, приписанными к IR

Команда END

Последняя команда, необходимая для завершения простейшей программы -команда END. Когда ЦУ сканирует программу, оно выполняет все команды до первой команды END, возвращается в начало программы и выполняет ее снова. Хотя команду END можно поместить в любую точку программы, что иногда делается при отладке, команды после первой встреченной команды END не будут исполняться, пока не будет удалена END. Число, следующее за командой END в мнемокоде - функциональный код, который используется для ввода большинства команд ПК. Об этом будут объяснения далее. Команда END не требует операндов и на командной линии END нельзя ставить никакую команду.



Адрес	Инструкция	Операнд
00500	LD	00000
00501	AND NOT	00001
00502	Инструкция	
00503	END(01)	-

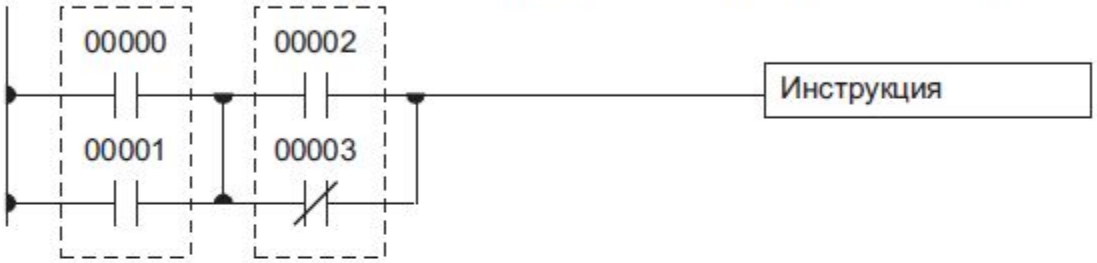
Если в программе нет команды END, программа вообще не будет выполняться.

Команды логического блока

Команды логического блока не соответствуют конкретным условиям на РКС, а описывают отношения между блоками. Команда AND LOAD производит операцию ЛОГИЧЕСКОЕ И над условиями исполнения, произведенными двумя логическими блоками. Команда OR LOAD производит операцию ЛОГИЧЕСКОЕ ИЛИ над условиями исполнения, произведенными двумя логическими блоками.

AND LOAD

Хотя данная РКС проста по форме, она требует команды AND LOAD.



Адрес	Инструкция	Операнд
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	-

Два логических блока отмечены пунктиром. Изучение данного примера показывает, что условие исполнение команды будет = 1,

- когда какое-либо из условий левого блока = 1 (т.е. либо IR 00000, либо IR 00001 = 1) и
- когда какое-либо из условий правого блока = 0 (т.е. либо IR 00002 = 1, либо IR 00003 = 0).

OR LOAD

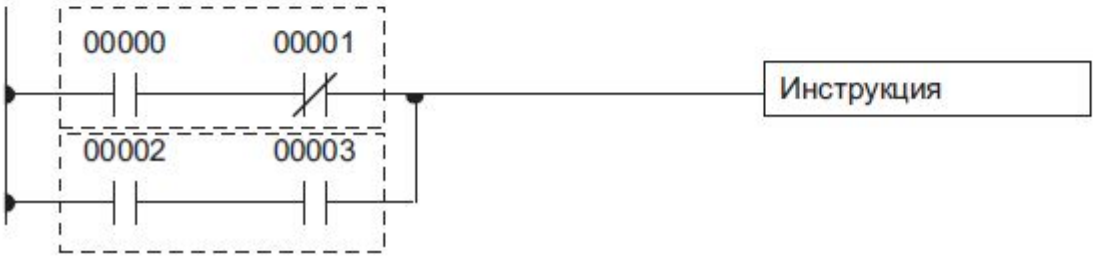
Следующая схема требует команды OR LOAD между верхним и нижним логическими блоками. Условие исполнения для “правосторонней” команды будет = 1,

- либо когда IR 00000 = 1 и IR 00001 = 0
- либо когда IR 00002 = 1 и IR 00003 = 1.

Работа мнемокода операции OR LOAD аналогично команде AND LOAD, за исключением того, что над текущим условием исполнения и последним неиспользованным условием исполнением производится операция ИЛИ.

Разумеется, некоторые схемы могут потребовать обеих команд AND LOAD и OR LOAD.

Адрес	Инструкция	Операнд
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	



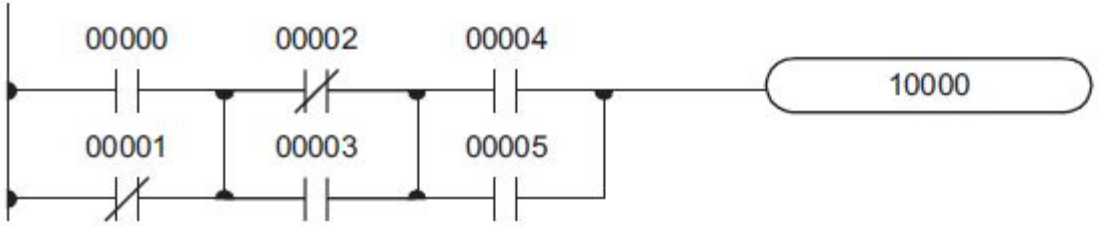
Последовательное включение команд логического блока

Для кодирования РКС с последовательным включением блоковых команд РКС должна быть разбита на логические блоки. Каждый блок кодируется с использованием команды LOAD для кодирования первого условия, затем AND LOAD или OR LOAD для логического объединения блоков. И для AND LOAD, и для OR LOAD есть 2 метода.

Закодировать блоковую команду после первых двух блоков и далее после каждого нового блока.

Кодирование всех блоков, подлежащих объединению, начиная каждый блок с LOAD или LOAD NOT, а затем блоковые команды для их объединения. В этом случае команды для последней пары блоков должны быть заданы первыми, и далее для каждого предшествующего блока, возвращаясь к первому блоку. Хотя оба этих метода производят одинаковый результат, второй метод может применяться только в случае, когда число блоков не более 8.

Следующая схема требует AND LOAD для преобразования в мнемокод, поскольку последовательно расположены три параллельных условия. Показаны два метода кодирования программы.



Адрес	Инструкция	Операнд
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	-
00005	LD	00004
00006	OR	00005
00007	AND LD	-
00008	OUT	10000

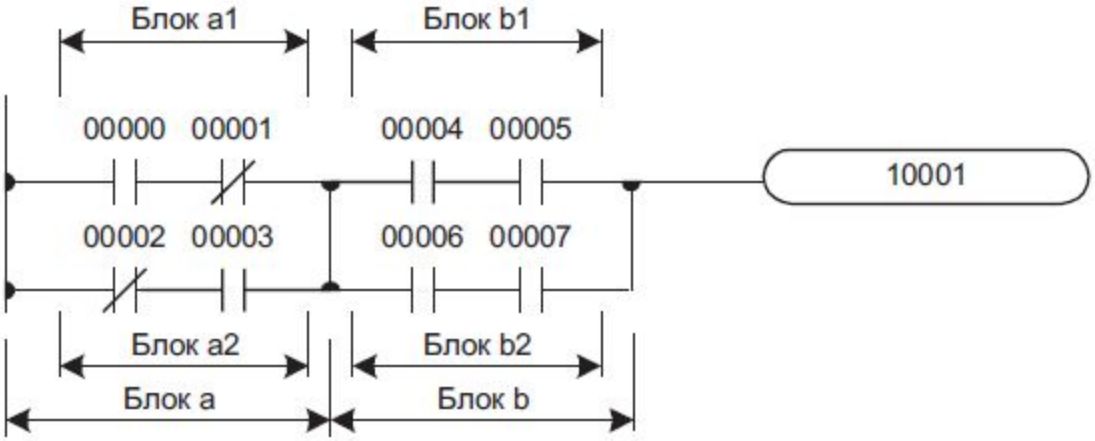
Адрес	Инструкция	Операнд
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	LD	00004
00005	OR	00005
00006	AND LD	-
00007	AND LD	-
00008	OUT	10000

Сложные схемы

При определении, какие блоковые команды будут необходимы для кодирования схем, иногда необходимо разбить схему на большие блоки и затем продолжать дробить большие блоки до тех пор, пока не будут сформированы блоки, которые можно программировать без блоковых команд. Затем данные блоки кодируются, сначала объединяются маленькие блоки, затем объединяются большие. Для объединения блоков используются команды AND LOAD или OR LOAD, т.е. AND LOAD или OR LOAD всегда объединяют два последних существующих условия исполнения, независимо от того, является ли оно результатом от единственного условия, логического блока или предыдущих блоковых команд.

При работе со сложными схемами блоки начинают кодировать с левого верхнего края, двигаясь сначала вниз, затем пересечение вправо. Это значит, что, если имеется выбор, OR LOAD нужно кодировать перед AND LOAD.

Следующую схему перед тем, как ее можно будет кодировать, нужно разбить на 2 блока и каждый из этих блоков разбить на 2 блока. Как показано, блоки a и b требуют AND LOAD. Перед применением AND LOAD нужно применить OR LOAD для объединения верхних и нижних блоков с каждой стороны, т.е. для объединения a1 и a2; b1 иb2 .



Адрес	Инструкция	Операнд
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	- ***(блок a1 и a2)***
00005	LD	00004
00006	AND	00005
00007	LD	00006
00008	AND	00007
00009	OR LD	- ***(блок b1 и b2)***
00010	AND LD	- ***(блок a и b)***
00011	OUT	10003