Московский государственный
университет
им. М. В. Ломоносова

Научно-исследовательский
вычислительный центр МГУ

Ленинградский государственный
университет
им. А. А. Жданова

Вычислительный центр ЛГУ

П10
2764
Вып. 15
1978

10
2764

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ВОПРОСЫ КИБЕРНЕТИКИ

ВЫПУСК 15

Под редакцией *Н. П. Брусенцова* и *А. М. Шаумана*

ИЗДАТЕЛЬСТВО МОСКОВСКОГО УНИВЕРСИТЕТА
1978

УДК 371.3:577.4:681.3

Печатается по постановлению
Редакционно-издательского совета
Московского университета

Рецензенты:

проф. *В. И. Дмитриев*, доц. *И. Л. Братчиков*

ГОСУДАРСТВЕННАЯ ПУБЛИЧНАЯ БИБЛИОТЕКА Ленинград ОЭ 1978 ОКТ 754

В $\frac{30502-035}{077(02)-78}$ 79—78 © Издательство Московского университета, 1978 г.

Н. П. БРУСЕНЦОВ, Х. РАМИЛЬ АЛЬВАРЕС

СТРУКТУРИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА МАЛОЙ ЦИФРОВОЙ МАШИНЕ

Структурированное программирование [1] расценивается обычно как методика разработки больших, сложных программ, реализуемая на базе языка высокого уровня, например Фортрана или Кобола. При этом преимущества структурированных программ (регулярность логической структуры, легкость понимания, отладки, обслуживания и внесения изменений) достигаются ценой некоторого снижения их эффективности: структурированная программа, как правило, длиннее обычной и выполняется дольше. Не видно, однако, никаких доводов против реализации структурированного программирования на уровне машинного языка, чтобы использовать его на всех последующих уровнях. Есть основания надеяться, что эффективность структурированных программ при этом повысится.

В 1975 г. авторы произвели в указанном направлении успешный эксперимент, введя в набор команд машины «Сетунь-70» [2] новые команды, функционально равноценные операторам структурированного программирования. Настоящая статья посвящена изложению предпосылок и результатов этой работы.

Разработка машины «Сетунь-70» основывалась на опыте освоения и использования в вузах и в ряде промышленных организаций малой цифровой машины «Сетунь», выпускавшейся в 1962—1965 гг. Этот опыт убедительно подтвердил целесообразность тех принципов построения малой цифровой машины, которые определили архитектуру «Сетуни» [3]:

короткое машинное слово (9 тритов, что эквивалентно 14,3 битам),

компактный набор простых команд (24 команды),

страничная (зонная) структура памяти, выполнение памяти в виде двух ступеней (основная — на магнитном барабане, оперативная — в виде быстродействующего устройства небольшой емкости), обмен между которыми производится страницами,

наличие резерва скорости выполнения операций, позволяющего эффективно использовать интерпретирующие программы

для продолжения и развития в нужном направлении и до желаемого уровня вычислительных и управляющих функций, которые аппаратурой осуществляются в весьма элементарном виде.

Реализация данных принципов, составивших впоследствии основу архитектуры миникомпьютеров, не только позволила значительно удешевить машину и резко повысить величину отношения производительность/цена, но также обеспечила появление у нее таких важных качеств, как программная модифицируемость и приспосабливаемость к применениям (настраиваемость на применение). Не предназначавшаяся для системных применений «Сетунь» была успешно использована в ряде весьма разнохарактерных автоматизированных систем: статистической обработки данных, представленных графиками, структурного анализа кристаллов, прогноза погоды, проектирования строительных конструкций, программированного обучения и др. Как вычислительная машина, она в зависимости от используемой интерпретирующей программы могла работать с фиксированной и с плавающей запятой, с 6, 8 и 12 десятичными знаками, а также с комплексными числами. Существенно, что интерпретирующие программы не были «зашиты» в постоянном запоминающем устройстве (ПЗУ), а вводились по выбору с перфоленты на магнитный барабан. Благодаря этому, с одной стороны, машина была более простой и дешевой, а с другой стороны, легко настраивалась на желаемый вычислительный режим и для использования ее в той или иной системе не требовалось «перешивать» ПЗУ.

Принципы, определившие направление и характер разработки «Сетуни-70» (1966—1968 гг.), в общем были теми же, на которых базировалась разработка «Сетуни». Вместе с тем архитектура машины претерпела значительные изменения. Стремление повысить компактность представления программ и создать благоприятные условия для оптимальной компиляции привело к использованию в качестве машинного языка польской инверсной записи (ПОЛИЗ). Кроме того, были произведены многие другие усовершенствования, направленные на более эффективное использование оборудования, обеспечение возможности наращивать память и периферию, усиление приспосабливаемости к применениям и т. п. В частности, размер прямо адресуемой единицы памяти был уменьшен с 9 до 6 тритов, минимальная емкость главной памяти увеличена с 3 до 27 страниц (в том числе 18 страниц ПЗУ для системных программ) с возможностью дальнейшего расширения при неизменных форматах команд и незначительных изменениях аппаратуры; обеспечена возможность обрабатывать совместно данные длиной 6, 12 и 18 тритов и длиной результата до 36 тритов; введена приоритетная система прерываний, привилегированные режимы и аппарат программируемых макроопераций, призванный повысить эффективность использования интерпретирующих программ; реорганизована и значительно расширена периферийная часть машины, в состав которой включены три канала ввода-вывода, способные обслужить до 24 устройств, и три канала страничного обмена для устройств периферийной памяти.

Несмотря на то что та программная надстройка, на которую рассчитывались все эти средства, не была осуществлена, машину удалось эффективно использовать, причем в деле, специфика которого в процессе разработки учтена не была, а именно в автоматизированной системе обучения [4]. Правда, при этом почти не используются вычислительные ресурсы машины и свойственная ей простота программирования арифметических выражений в ПОЛИЗ, но в остальном архитектура оказалась хорошо отвечающей требованиям программированного управления учебой и построенная на основе ее обучающая система может быть охарактеризована как достаточно гибкая и экономичная. Опыт создания этой системы дает право утверждать, что машина окажется весьма подходящей и для многих других автоматизированных систем управления.

Тем не менее разработка программного оборудования обучающей системы в условиях, когда машина, рассчитанная на программную реализацию многих выполняемых обычно аппаратурой функций, вообще не оснащена программами, естественно была трудоемкой. Известные трудности составления и отладки логически сложных управляющих программ значительно увеличивались вследствие программирования на языке машинных команд и из-за страничной структуры памяти.

Стремясь уменьшить трудоемкость и повысить эффективность программистского труда, мы стали внедрять в практику нашей работы структурированное программирование. Однако управляющие команды машины плохо соответствовали операторам управления, используемым в структурированных программах, и поэтому машинная эффективность этих программ получалась низкой, что на малой машине с ее скудными ресурсами совершенно неприемлемо. Решили перестроить архитектуру машины применительно к требованиям структурированного программирования, чтобы структурированные программы могли стать машинно-эффективными.

Архитектура «Сетуни-70» оказалась исключительно благоприятной для указанной перестройки. То, что команды этой машины представляются сочетаниями адресных и операционных слогов и имеют варьируемую длину, позволило легко построить в машинном языке конструкции, прямо соответствующие основным операторам структурированного программирования (ветвление, повторение, выполнение подпрограммы).

Наличие же в машине стекового механизма для автоматического упрятывания и восстановления (операция «возврат») значений программного счетчика обеспечило возможность простой и эффективной реализации новых команд.

Перестройка значительно осложнялась тем, что ее надо было осуществить, не нарушив нормальной эксплуатации машины и избежав сколько-нибудь существенных переделок в имеющихся программах. Последнее условие казалось особенно трудным, потому что свободных кодов операций не было и новые операции можно было ввести, лишь исключив некоторые из имеющихся. К счастью, оказалось, что имеются операции либо вообще не использованные в программах, либо использованные исключительно редко. Анализ показал, что среди этих операций имеют место действительно бесполезные, утратившие значение, например, в результате того, что приемы программирования, для реализации которых они были предусмотрены, на практике оказались ненужными вследствие изобретения других, более удачных приемов. Так, практически не получили применения операции прибавления единицы к содержимому регистра E и вычитания единицы из этого регистра ($E+1$ и $E-1$). Незначительным оказалось применение операции, устанавливающей в E нуль ($E=0$). Не нашла употребления операция засылки в арифметический стек части текущего значения программного счетчика, определяющей адрес выполняемого слога программы в пределах страницы ($T=C$), предназначавшаяся для запоминания адреса возврата при обращении к подпрограмме. Именно эти четыре операции были изъяты из числа выполняемых машиной и заменены новыми операциями.

В число новых операций вошли операции ветвления, повторения (цикла), выполнения подпрограммы и обобщенного перехода. Последняя к структурированному программированию отношения не имеет и в дальнейшем не рассматривается.

Операция *ветвления* является троичным аналогом оператора *IF THEN ELSE*. Ветвление производится в зависимости от знака содержимого вершины T арифметического стека выбором одной из трех альтернатив:

if $T<0$ **then** $P1$ **else if** $T=0$ **then** $P2$ **else** $P3$.

Здесь $P1$, $P2$, $P3$ — идентификаторы подпрограмм. Попарное отождествление $P1$, $P2$, $P3$ позволяет получить любое из двухальтернативных ветвлений. Например,

при $P3 \equiv P2$ имеем **if** $T<0$ **then** $P1$ **else** $P2$,
при $P2 \equiv P1$ имеем **if** $T \leqslant 0$ **then** $P1$ **else** $P3$

и т. д.

Команда *ветвления* представляет собой последовательность четырех слогов: $K0$ $K1$ $K2$ $K3$. Слог $K0$ — операционный, код операции ветвления. Остальные три слога — адресные, начальные адреса подпрограмм соответственно $P1$, $P2$, $P3$. Выполняя команду ветвления, машина запоминает в системном стеке адрес возврата, равный $PC+4$, где PC — текущее значение программного счетчика (адрес слога $K0$), и засылает в программный счетчик один из адресных слогов команды — $K1$, $K2$, $K3$ в зависимости от знака содержимого вершины T арифметического стека. Таким образом происходит переход на одну из подпрограмм $P1$, $P2$, $P3$. Каждая из этих подпрограмм оканчивается слогом «возврат», вызывающим обратную пересылку адреса возврата из системного стека в программный счетчик, т. е. переход на продолжение программы, на команду, непосредственно следующую за командой ветвления.

Операция *повторения* реализована в виде **while** $T \neq 0$ **do** P, где P — индентификатор подпрограммы.

Команда *повторения* состоит из двух слогов — KO KA, операционного и адресного, указывающего начальный адрес подпрограммы P.

Если условие $T \neq 0$ выполнено, то текущее значение PC, являющееся адресом первого слога команды повторения, отсылается в системный стек, а его место в программном счетчике занимает второй слог этой команды — KA, т. е. происходит переход на подпрограмму P. Подпрограмма оканчивается операционным слогом «возврат», который вызывает обратный переход на команду повторения, таким образом замыкая цикл.

Если же при очередном выполнении слога KO команды повторения условие $T \neq 0$ окажется невыполненным, то машина пропускает слог KA этой команды и переходит к выполнению следующего за ним слога программы.

Команда *выполнения подпрограммы* также имеет вид KO KA, но интерпретируется иначе. Выполняя ее, машина не проверяет условия и в качестве адреса возврата запоминает в системном стеке не PC, а $PC+2$. Таким образом, переход на подпрограмму производится безусловно, а возврат осуществляется на команду, непосредственно следующую за командой выполнения подпрограммы. Другими словами, команда выполнения подпрограммы в процессе выполнения просто замещается указываемой ею подпрограммой.

Длина машинного слога — 6 тритов — достаточна для обеспечения прямой адресации в пределах 9 страниц по 81 слогу, т. е. во всей оперативной (чтение и запись) памяти машины. Доступ к подпрограммам, расположенным в ПЗУ, обеспечивается аппаратом макроопераций.

Практика разработки и эксплуатации программ в условиях модернизированной таким образом системы команд полностью подтвердила отмечаемые обычно преимущества струк-

турированного программирования: программы на самом деле отличаются легко прослеживаемой логикой, их проще проверять, отлаживать, модифицировать. Существенно, что все эти преимущества получены без снижения машинной эффективности программ, сопутствующей обычно применению структурированного программирования на уровне машинно-независимых языков. Но, кроме того, специфическая форма машинной реализации операторов структурированного программирования на основе стекового механизма обращения к подпрограмме обусловила возможность дальнейшего улучшения структуры программы и явилась избавлением от неудобств, которые при обычном программировании имели место из-за страничной организации памяти машины.

Как видно из данного выше описания, команды ветвления и цикла отличаются от соответствующих операторов структурированного программирования, в частности, тем, что тела ветвей и цикла в них заменены ссылками на подпрограммы (именами или адресами начала подпрограмм). С этим отличием связаны по меньшей мере два следующих преимущества. Во-первых, вынесение фрагментов программы, составляющих тела ветвей и циклов, в подпрограммы, на которые ссылается главная программа, значительно улучшает обозримость и понятность последней. Во-вторых, программа, представлявшая собой единую последовательность команд, отладка и выполнение которой с учетом имеющихся в ней ветвлений и циклов в условиях страничной памяти сопряжены с известными трудностями, теперь распадается на сравнительно небольшие части, которые можно автономно отлаживать и произвольно размещать в памяти. Ведущая часть (главная программа) состоит в основном из команд ветвления, цикла и безусловного выполнения подпрограммы, содержащих ссылки на другие части (подпрограммы первого уровня), которые в свою очередь могут содержать команды, ссылающиеся на подпрограммы второго уровня и т. д. Не исключены также ссылки в подпрограммах более высоких уровней на подпрограммы низших уровней.

ЛИТЕРАТУРА

1. Дал У., Дейкстра Э., Хоор К. Структур(ирован)ное программирование. М., «Мир», 1975.

2. Брусенцов Н. П., Жоголев Е. А. Структура и алгоритм функционирования малой вычислительной машины. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 8. Изд-во ЛГУ, 1971, с. 34—51.

3. Брусенцов Н. П., Жоголев Е. А. и др. Малая автоматическая цифровая машина «Сетунь». — «Вестн. Моск. ун-та, сер. матем.» 1962, № 4; 3—12.

4. Брусенцов Н. П., Маслов С. П., Рамиль Альварес Х. Автоматизированная система обучения «Наставник». — В кн.: Вычислительная техника и вопросы кибернетики, вып. 13. Изд-во МГУ, 1976, с. 3—17.

Ю. Ю. ГАЛИМОВ

ОСНОВНЫЕ ЧЕРТЫ СОВРЕМЕННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ *APL*

Задача описания *APL* довольно сложна. Стандарта языка пока нет, и поэтому строго следует говорить об *APL*\360 или *APL.SV* фирмы *IBM*, *APL*/700 фирмы *Burroughs* и т. д. Однако все эти реализации объединяет, во-первых, довольно жесткое общее ядро и, во-вторых, «дух *APL*» — особенность языка, о которой часто говорят. В какой-то степени присутствие этого «духа *APL*» объясняет тот факт, что человек, знакомый с языком, обычно становится либо его «фанатиком», либо «ниспровергателем». Преобладают именно крайние точки зрения, причем в свое отношение к языку человек обычно вкладывает что-то личное — ситуация, очень характерная для *APL* и требующая объяснения.

Вероятно, наличием явного и характерного собственного «духа» *APL* обязан очень тщательной разработке, которая проводилась автором языка К. Айверсоном в течение длительного времени, с середины 50-х до конца 60-х годов. Будучи профессором в Гарварде, К. Айверсон столкнулся с трудностями при описании процессов и систем обработки информации — трудностями, связанными с отсутствием удобной системы обозначений. Такая система обозначений (такой язык) должна быть строгой, точной, мнемоничной. Кроме того, — и это очень важно — язык сознательно рассматривался Айверсоном в первую очередь как инструмент, и освоение этого инструмента должно быть простым и легким (значительно легче, чем разработка алгоритмов, для описания которых он предназначен). Он должен не усложнять работу, но, наоборот, «провоцировать мысль». Эти принципы легли в основу языка *APL*, который разрабатывался К. Айверсоном в процессе преподавательской деятельности в Гарварде, во время которой Айверсон имел возможность постоянно проверять достоинства и недостатки языка на практике. Итогом этой работы была публикация в 1962 г. книги [1]. Описанный там язык не предназначался для реализации на ЭВМ, под программированием в [1] понимается вообще описание процессов и систем обработки информации. После перехода в IBM Айверсон продолжил работу над языком. В 1964 г. было опубли-

обсуждение которых, однако, не является сейчас нашей целью.

Литературы по *APL* на русском языке практически нет. Лучшим учебником по *APL*, на наш взгляд, является книга [7], в которой очень последовательно и полно описаны язык и система *APL*/360 и *APL. SV*, включая и недавние уточнения и расширения этих систем. В качестве справочников по системам *APL*/360 и *APL*/700 (но не для знакомства с *APL*) можно использовать [8] и [9]. Идеология разработки *APL* очень подробно описана его создателями в статье [10].

```
      3×4
12
      A←5
      A—2
3
      A÷
SYNTAX ERROR
A÷
 ∧
```

Рис. 1

```
      ∇ A COMPARE B
[1]→(L, E, G) [2+×A—B]
[2] L: 'LESS'
[3]→0
[4] E: 'EQUAL'
[5]→0
[6] G: 'GREATER'
      ∇
      3 COMPARE 4
LESS
      4 COMPARE 4
EQUAL
      5 COMPARE 4
GREATER
```

Рис. 2

ЛИТЕРАТУРА

1. Iverson K. E. A Programming Language. John Wiley and Sons, New York, 1962.
2. Falkoff A. D., Iverson K. E., Sussenguth E. H. A Formal Description of System 360. — «IBM Systems J.», 1964, 3, N 3, 198—261.
3. Falkoff A. D., Iverson K. E. The APL/360 Terminal System.— «Symposium on Interactive Systems for Experimental Applied Mathematics». Academic Press, New York, 1968, p. 22—37.
4. Harris L. R. A Logical Control Structure for APL. — «APL Congress 73». North-Holland Pub. Co., London—New York, 1973, p. 203—210.
5. Kelley R. A. APLGOL, an Experimental Structured Programming Language. — «IBM J. of Research and Developments», 1973, 17, N 1 69—73.
6. «APL Language» Order N GC26—3847—0. IBM Corp. Palo Alto, Calif., 1975.
7. Gilman L., Rose A. «APL — An Interactive Approach», 2 ed. John Wiley and Sons, New York, 1974.
8. Pakin S. «APL/360 Reference Manual», 2 ed. Science Research Ass. Chicago, 1972.
9. «APL/700 User Reference Manual» Burroughs Corp., 1973.
10. Falcoff A. D., Iverson K. E. The Design of APL. — «IBM J. of Research and Developments», 1973, 17, N 3, 324—334.

Х. РАМИЛЬ АЛЬВАРЕС

АЛГОРИТМЫ ПЕРЕВОДОВ ДЛЯ СИСТЕМ С ПОЛОЖИТЕЛЬНЫМИ И ОТРИЦАТЕЛЬНЫМИ БАЗИСНЫМИ ЧИСЛАМИ

Алгоритмы переводов чисел из одной позиционной системы счисления в другую разбираются во многих работах, например [1—4]. В основном рассматриваются системы с неотрицательными базисными числами. В связи с использованием симметричного троичного кода в ЦВМ [5] и возможностью его использования в системах передачи данных представляют интерес алгоритмы переводов для систем с положительными и отрицательными базисными числами.

При создании программного обеспечения для машин «Сетунь» [6] были разработаны алгоритмы переводов из десятичной системы в симметричную троичную систему и обратно [7, 8]. В этих алгоритмах использовалась операция умножения.

В данной работе рассматриваются алгоритмы переводов, неиспользующие операций умножения, для некоторых систем счисления с положительными и отрицательными базисными числами.

1. Позиционные системы по основанию p

В позиционных системах по основанию p *натуральное представление* числа N есть массив

$$(a_n \ldots a_0),$$

при этом $N = \sum_{i=0}^{n} a_i p^i$.

Совокупность $l (l \geqslant p)$ значений $\alpha_j (j=1,\ldots,l)$, принимаемых элементами a_i, называется *базисными числами*.

Системы с $l=p$ будем называть *полными*, а системы с $l>p$ будем называть *избыточными*.

Полная система с базисными числами $\alpha_j (j=1,\ldots,p)$, удовлетворяющими условию

$$0 \leqslant \alpha_j \leqslant p-1,$$

называется *p-ичной системой*.

Полная система с положительными и отрицательными базисными числами $\alpha_j (j=1,...,p)$, удовлетворяющими условию

$$-k \leqslant \alpha_j \leqslant p-k-1,$$

где $0 \leqslant k \leqslant p-2$, называется *$(p, k)$-ичной системой.* Частным случаем (p, k)-ичных систем являются симметричные системы, в которых число положительных базисных чисел равно числу отрицательных, т. е. $p=2k+1$. Такие системы будем называть *p-ичными.*

Избыточные системы с базисными числами $\alpha_j (j=1,...,l)$, удовлетворяющими условию

$$-k-n \leqslant \alpha_j \leqslant p-k-1+m,$$

где $n \geqslant 0$, $m \geqslant 0$, $k \geqslant 0$ и $l=p+n+m$, будем называть *(p, k, n, m)-ичными системами.* Заметим, что натуральное представление числа N в (p, k)-ичной системе

$$(a_n \ldots a_0)_{(p,\,k)}$$

является также натуральным представлением и в (p, k, n, m)-ичной системе.

Избыточную систему (p, k, n, m)-ичную с натуральным представлением

$$(a_n \ldots a_0),$$

удовлетворяющим условиям

$$-k \leqslant a_i \leqslant p-k-1, \quad \text{для } i \neq 0,$$

$$k-n \leqslant a_0 \leqslant p-k-1+m$$

будем называть *$[p, k, n, m]$-ичной.*

Заметим, что натуральное представление числа N в (p, k)-ичной системе является также натуральным представлением числа в $[p, k, n, m]$-ичной системе, а $[p, k, n, m]$-ичное натуральное представление числа есть одновременно (p, k, n, m)-ичное натуральное представление.

2. Вычеты и неполные частные

Для целых чисел A и $p (p \geqslant 2)$ введем

$\{A\}_p$ — *наименьший неотрицательный вычет по основанию p,* удовлетворяющий условию

$$0 \leqslant \{A\}_p \leqslant p-1;$$

$[A]_p$ — *неполное частное по основанию p,* определяемое из соотношения

$$A = p \cdot [A]_p + \{A\}_p.$$

Для целых чисел A, $p (p \geqslant 2)$ и $k (0 \leqslant k \leqslant p-2)$ введем

$\{A\}_{(p,k)}$ — *вычет по основанию (p, k),* удовлетворяющий условию

$$-k \leqslant \{A\}_{(p,\,k)} \leqslant p-k-1;$$

$[A]_{(p,k)}$ — *неполное частное по основанию (p, k),* определяемое из соотношения

$$A = p \cdot [A]_{(p,\,k)} + \{A\}_{(p,\,k)}.$$

Заметим, что для $k=0$ $\{A\}_{(p,k)}$ есть $\{A\}_p$ и $[A]_{(p,k)}$ есть $[A]_p$.

3. Переводы $p \to p \pm 1$

В работе [4] рассмотрены вопросы перевода чисел из p-ичной системы в $(p+1)$-ичную (перевод $p \longrightarrow p+1$) и в $(p-1)$-ичную системы (перевод $p \longrightarrow p-1$).

Правила переводов основывались на алгоритмах вычисления вычетов и неполных частных по основаниям $p+1$ и $p-1$, а также на алгоритмах умножения на $p+1$ и $p-1$.

Алгоритмы вычисления вычетов и неполных частных по основаниям $p+1$ и $p-1$ в общем виде записываются следующим образом:

$$[(a_n \ldots a_0)_p]_{p\pm1} = (b_n \ldots b_0)_p,$$

$$\{(a_n \ldots a_0)_p\}_{p\pm1} = c_0,$$

где $b_i (i=n, n-1,..., 0)$ и c_0 определяются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i \mp c_{i+1}]_{p\pm1}, \\ c_i &= \{a_i \mp c_{i+1}\}_{p\pm1}, \\ c_{n+1} &= 0. \end{aligned} \tag{1}$$

Алгоритмы умножения целых чисел в p-ичной системе на $p+1$ и на $p-1$ в общем виде записываются следующим образом:

$$(p \pm 1) \cdot (a_n \ldots a_0)_p = (b_{n+2} \ldots b_0)_p,$$

где b_i $(i = 0, 1, \ldots, n+2)$ определяются по формулам

$$\begin{aligned} b_i &= \{c_i \pm a_i\}_p, \\ c_{i+1} &= a_i + [c_i \pm a_i]_p, \\ c_0 &= a_{n+1} = a_{n+2} = 0. \end{aligned} \tag{2}$$

4. Переводы $p \longrightarrow p+t$

Правила переводов $p \to p+t$ (для t положительного или отрицательного) основываются на алгоритмах вычисления вычетов и неполных частных по основанию $p+t$, а также на алгоритмах умножения на $p+t$.

Алгоритмы вычисления вычетов и неполных частных по основанию $p+t$ могут быть записаны в следующем виде:

$$[(a_n \ldots a_0)_p]_{p+t} = (b_n \ldots b_0)_p,$$

$$\{(a_n \ldots a_0)_p\}_{p+t} = c_0,$$

где b_i $(i = n, n-1, \ldots, 0)$ и c_0 определяются по формулам

$$b_i = c_{i+1} + [a_i - t \cdot c_{i+1}]_{p+t},$$

$$c_i = \{a_i - t \cdot c_{i+1}\}_{p+t}, \qquad (3)$$

$$c_{n+1} = 0.$$

Заметим, что формулы (3) для случаев $t=1$ и $t=-1$ совпадают с формулами (1).

Покажем, что $(b_n \ldots b_0)$ есть натуральное представление в p-ичной системе.

Формулу для b_i можно записать в виде

$$b_i = [a_i + p \cdot c_{i+1}]_{p+t}.$$

Имеем

$$0 \leqslant a_i \leqslant p-1,$$

$$0 \leqslant c_{i+1} \leqslant p+t-1,$$

$$0 \leqslant a_i + p \cdot c_{i+1} \leqslant p-1+p \cdot (p+t-1),$$

$$0 \leqslant a_i + p \cdot c_{i+1} \leqslant (p-1) \cdot (p+t) + (p+t-1),$$

откуда

$$0 \leqslant [a_i + p \cdot c_{i+1}]_{p+t} \leqslant p-1.$$

Приведем пример применения алгоритма для перевода числа 4095 из десятичной системы в восьмеричную ($p=10$, $t=-2$)

```
      4 09  5
  ----------------
  c 0 4 01 |7|
  b   0 51  1
  ----------------
  c   0 53 |7|
  b     06  3
  ----------------
  c     06 |7|
  b       0  7
  ----------------
  c       0 |7|
  b           0
  ----------------
```

Таким образом, имеем $(4\,0\,9\,5)_{10} = (7\,7\,7\,7)_8$.

Алгоритмы умножения в p-ичной системе целого числа на $p+t$ $(|t| \leqslant p)$ можно записать в следующем виде:

$$(p+t) \cdot (a_n \ldots a_0)_p = (b_{n+3} \ldots b_0)_p,$$

где $b_i (i=0,1,\ldots,n+3)$ определяются по формулам

$$b_i = \{c_i + t \cdot a_i\}_p,$$

$$c_{i+1} = a_i + [c_i + t \cdot a_i]_p, \qquad (4)$$

$$c_0 = a_{n+1} = a_{n+2} = a_{n+3} = 0.$$

Заметим, что при $t=1$ и $t=-1$ получаем формулы (2).

Рассмотрим пример использования алгоритма умножения на $p+t$ для перевода числа 7777 из восьмеричной системы в десятеричную ($p=10$, $t=-2$), при этом c_0 на каждом шаге равно очередной цифре переводимого числа:

```
      7777
  ----------
  c 07
  b 07
  ----------
  c 067
  b 063
  ----------
  c 0537
  b 0511
  ----------
  c 04017
  b 04095
  ----------
```

Отметим, что формулы (4) можно рассматривать как сложение в p-ичной системе чисел $(a_n \ldots a_0 0)_p$ и $t \cdot (a_n \ldots a_0)$. На этом основывается алгоритм перевода целых чисел $p \longrightarrow p+t$, приведенный в работе [9]. Рассмотрим пример с использованием этого алгоритма, рекомендованного в работе [2] для ручного перевода:

```
    7'777
  -  14
  ---------
    63'7
  -  126
  ---------
    511'7
  -  1022
  ---------
    4095
```

Таким образом, имеем $(7777)_8 = (4095)_{10}$.

5. Переводы $p \longrightarrow (p \pm 1,\ k)$

Рассматриваются алгоритмы переводов целых чисел из p-ичной системы в $(p \pm 1,\ k)$-ичные системы, которые основы-

ваются на правилах вычисления вычетов и неполных частных по основаниям $(p\pm1, k)$ в некоторых системах счисления.

5.1. Вычисление вычетов и неполных частных по основанию $(p+1, k)$ в $(p, 0, k, 0)$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{(p,\,0,\,k,\,0)}]_{(p+1,\,k)} = (b_n \ldots b_0)_{(p,\,0,\,k,\,0)},$$
$$\{(a_n \ldots a_0)_{(p,\,0,\,k,\,0)}\}_{(p+1,\,k)} = c_0,$$

где c_0 и b_i $(i=n, \ldots, 0)$ определяются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i - c_{i+1}]_{(p+1,\,k)}, \\ c_i &= \{a_i - c_{i+1}\}_{(p+1,\,k)}, \\ c_{n+1} &= 0. \end{aligned} \tag{5}$$

Заметим, что при $k=0$ формулы (5) совпадают с формулами (1) для случая $p+1$.

Следует показать, что $(b_n \ldots b_0)$ есть натуральное представление в $(p, 0, k, 0)$-ичной системе.

Формулу для b_i можно записать в виде

$$b_i = [a_i + p\cdot c_{i+1}]_{(p+1,\,k)}.$$

Имеем

$$-k\cdot(p+1) \leqslant a_i + p\cdot c_{i+1} \leqslant p-1+p\cdot(p-k),$$
$$-k\cdot(p+1) \leqslant a_i + p\cdot c_{i+1} \leqslant (p+1)\cdot(p-k)+k-1,$$
$$-k \leqslant [a_i + p\cdot c_{i+1}]_{(p+1,\,k)} \leqslant p-k+[k-1]_{(p+1,\,k)}.$$

Правый член неравенства для $k\leqslant1$ равен $p-1$, а для $k\geqslant2$ имеем

$$p-k+[k-1]_{(p+1,\,k)} \leqslant p-k+1 \leqslant p-1.$$

Таким образом,

$$-k \leqslant b_i \leqslant p-1,$$

т. е. $(b_n \ldots b_0)$ есть натуральное представление в $(p, 0, k, 0)$-ичной системе.

Приведем пример использования алгоритма для перевода числа 7777 из восьмеричной системы в симметричную девятеричную систему (перевод $8\to\overline{9}$), $p=8$, $k=4$[1]:

		7	7	7	7
c	0	$\overline{2}$	0	$\overline{2}$	$\boxed{0}$
b		1	$\overline{1}$	1	$\overline{1}$
c	0	1	$\overline{2}$	3	$\boxed{\overline{4}}$
b		0	1	$\overline{2}$	3
c	0	1	$\overline{3}$	$\boxed{\overline{3}}$	
b		0	1	$\overline{2}$	
c	0	1	$\boxed{\overline{3}}$		
b		0	1		
c	0	$\boxed{1}$			
b		0			

Таким образом, имеем $(7777)_8 = (1\overline{3}\overline{3}\overline{4}0)_{\overline{9}}$.

5.2. Вычисление вычетов и неполных частных по основанию $(p-1, k)$ в $(p, 0, k+1, 0)$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{(p,\,0,\,k+1,\,0)}]_{(p-1,\,k)} = (b_n \ldots b_0)_{(p,\,0,\,k+1,\,0)},$$
$$\{(a_n \ldots a_0)_{(p,\,0,\,k+1,\,0)}\}_{(p-1,\,k)} = c_0,$$

где c_0 и b_i $(i=n, n-1, \ldots, 0)$ определяются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i + c_{i+1}]_{(p-1,\,k)}, \\ c_i &= \{a_i + c_{i+1}\}_{(p-1,\,k)}, \\ c_{n+1} &= 0. \end{aligned} \tag{6}$$

Заметим, что при $k=0$ формулы (6) совпадают с формулами (1) для случая $p-1$.

Следует показать, что $(b_n \ldots b_0)$ есть натуральное представление в $(p, 0, k+1, 0)$-ичной системе.

Имеем

$$-(k+1)-k \leqslant a_i + c_{i+1} \leqslant (p-1)+(p-k-2).$$

Отметим, что $-2k-1 = -k+(-(p-1)+(p-k-2))$,

поэтому $$[-2k-1]_{(p-1,\,k)} \geqslant -1.$$

Отсюда

$$-1 \leqslant [a_i + c_{i+1}]_{(p-1,\,k)} \leqslant 1$$

и

$$-k-1 \leqslant b_i \leqslant p-k-1.$$

Таким образом,

$$-k-1 \leqslant b_i \leqslant p-1.$$

Приведем пример использования алгоритма для перевода числа 7591 из десятичной системы в симметричную девятеричную систему (перевод $10\to\overline{9}$), $p=10$, $k=4$:

[1] В дальнейшем отрицательные базисные числа будем обозначать цифрами с надчеркиванием, например базисное число —2 будем обозначать $\overline{2}$.

$$
\begin{array}{lrrrr|l}
 & & 7 & 5 & 9 & 1 \\
\hline
c & 0 & \bar{2} & 3 & 3 & \underline{|4|} \\
b & & 1 & \bar{2} & 4 & 3 \\
\hline
c & 0 & 1 & \bar{1} & 3 & \underline{|\bar{3}|} \\
b & & 0 & 1 & \bar{1} & 4 \\
\hline
c & & 0 & 1 & 0 & \underline{|4|} \\
b & & & 0 & 1 & 0 \\
\hline
c & & & 0 & 1 & \underline{|1|} \\
b & & & & 0 & 1 \\
\hline
c & & & & 0 & \underline{|1|} \\
b & & & & & 0 \\
\hline
\end{array}
$$

Таким образом, имеем $(7591)_{10} = (11\bar{4}34)_{\bar{9}}$.

5.3. Вычисление вычетов и неполных частных по основаниям $(p \pm 1, k)$ в $(p, 0, p-1, 0)$-ичной системе

Обобщая формулы (5) и (6), можно утверждать, что

$$[(a_n \ldots a_0)_{(p,\,0,\,p-1,\,0)}]_{(p\pm1,\,k)} = (b_n \ldots b_0)_{(p,\,0,\,p-1,\,0)},$$
$$\{(a_n \ldots a_0)_{(p,\,0,\,p-1,\,0)}\}_{(p\pm1,\,k)} = c_0,$$

где c_0 и b_i $(i = n, n-1, \ldots, 0)$ определяются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i \mp c_{i+1}]_{(p\pm1,\,k)}, \\ c_i &= \{a_i \mp c_{i+1}\}_{(p\pm1,\,k)}, \\ c_{n+1} &= 0. \end{aligned} \tag{7}$$

Применение формул (7) дает возможность производить переводы $p \to (p \pm 1, k)$ и для отрицательных чисел, если в переводимом числе все цифры рассматривать как цифры со знаком числа.

Приведем пример использования алгоритма деления на $p+1$ для перевода числа -7591 из десятичной системы в симметричную девятеричную систему (перевод $10 \longrightarrow \bar{9}$), $p = 10$, $k = 4$:

$$
\begin{array}{lrrrr|l}
 & & \bar{7} & \bar{5} & \bar{9} & \bar{1} \\
\hline
c & 0 & 2 & \bar{3} & \bar{3} & \underline{|\bar{4}|} \\
b & & \bar{1} & 2 & \bar{4} & \bar{3} \\
\hline
c & 0 & \bar{1} & 1 & \bar{3} & \underline{|\bar{3}|} \\
b & & 0 & \bar{1} & 1 & \bar{4} \\
\hline
c & & 0 & \bar{1} & 0 & \underline{|\bar{4}|} \\
b & & & 0 & \bar{1} & 0 \\
\hline
c & & & 0 & \bar{1} & \underline{|\bar{1}|} \\
b & & & & 0 & \bar{1} \\
\hline
c & & & & 0 & \underline{|\bar{1}|} \\
b & & & & & 0 \\
\hline
\end{array}
$$

Таким образом, имеем $(-7591)_{10} = (\bar{1}\bar{1}\bar{4}3\bar{4})_{\bar{9}}$.

5.4. Вычисление вычетов и неполных частных по основаниям $(p \pm 1, k)$ в $[p, 0, 0, 1]$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{[p,\,0,\,0,\,1]}]_{(p\pm1,\,k)} = (b_n \ldots b_0)_{[p,\,0,\,0,\,1]},$$
$$\{(a_n \ldots a_0)_{[p,\,0,\,0,\,1]}\}_{(p\pm1,\,k)} = c_0,$$

где c_0 и b_i $(i = n, n-1, \ldots, 0)$ определяются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i \mp c_{i+1}]_{p\pm1}, \\ c_i &= \{a_i \mp c_{i+1}\}_{p+1} \quad \text{для } i \neq 0, \\ c_{n+1} &= 0, \\ b_0 &= c_1 + [a_0 \mp c_1]_{(p\pm1,\,k)}, \\ c_0 &= \{a_0 \mp c_1\}_{(p\pm1,\,k)}. \end{aligned} \tag{8}$$

Заметрим, что при $k = 0$ формулы (8) совпадают с формулами (1).

Следует показать, что $(b_n \ldots b_0)$ есть натуральное представление в $[p, 0, 0, 1]$-ичной системе, при этом, так как формулы (8) для $i \neq 0$ совпадают с (1), необходимо показать выполнение условий для b_0:

$$b_0 = c_1 + [a_0 \mp c_1]_{(p\pm1,\,k)} = [a_0 + p \cdot c_1]_{(p\pm1,\,k)}.$$

Рассмотрим случай $(p+1, k)$:

$$0 \leqslant a_0 + p \cdot c_1 \leqslant p + p \cdot p.$$

Отсюда

$$0 \leqslant [a_0 + p \cdot c_1]_{(p+1,\,k)} \leqslant p,$$

т. е. условие для b_0 выполняется.

Рассмотрим случай $(p-1, k)$:

$$0 \leqslant a_0 + p \cdot c_1 \leqslant p + p \cdot (p-2)$$

или

$$0 \leqslant a_0 + p \cdot c_1 \leqslant p \cdot (p-1).$$

2 Зак. 392

Отсюда

$$0 \leqslant [a_0 + p \cdot c_1]_{(p-1,\ k)} \leqslant p,$$

т. е. условие для b_0 выполняется.

Таким образом, показано, что $(b_n \ldots b_0)$ есть натуральное представление в $[p, 0, 0, 1]$-ичной системе.

Приведем пример использования алгоритма для перевода числа 7777 из восьмеричной в симметричную девятеричную систему (перевод $8 \to \overline{9}$), $p=8$, $k=4$:

		7	7	7	7
c	0	7	0	7	$\underline{0}$
b		0	7	0	7
c		0	7	2	$\underline{\overline{4}}$
b			0	6	3
c			0	6	$\underline{\overline{3}}$
b				0	6
c				0	$\underline{\overline{3}}$
b					1
c				0	$\underline{1}$
b					0

Таким образом, $(7777)_8 = (1\overline{3}\overline{3}\overline{4}0)_{\overline{9}}$.

6. Переводы $(p, k) \to p \pm 1$

Рассматриваются алгоритмы переводов неотрицательных целых чисел из (p, k)-ичной системы в $(p \pm 1)$-ичные системы, которые основываются на правилах вычисления вычетов и неполных частных по основаниям $(p \pm 1)$ в некоторых системах счисления.

6.1. Вычисления вычетов и неполных частных по основанию $p-1$ в $(p, k, 0, k+1)$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}]_{p-1} = (b_n \ldots b_0)_{(p,\ k,\ 0,\ k+1)},$$

$$\{(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}\}_{p-1} = c_0,$$

где c_0 и b_i $(i = n, n-1, \ldots, 0)$ определяются по формулам

$$\begin{aligned} b_i &= c_i + [a_i - c_{i+1}]_{p-1}, \\ c_i &= \{a_i - c_{i+1}\}_{p-1}, \\ c_{n+1} &= 0. \end{aligned} \tag{9}$$

Заметим, что формулы (9) при $k=0$ совпадают с формулами (1) для случая $p-1$.

Покажем, что $(b_n \ldots b_0)$ есть натуральное представление в $(p, k, 0, k+1)$-ичной системе.

Имеем

$$-k \leqslant a_i \leqslant p,$$

$$0 \leqslant c_{i+1} \leqslant p-2.$$

Формулу для b_i можно записать в виде

$$b_i = [a_i + p \cdot c_{i+1}]_{p-1},$$

$$-k \leqslant a_i + p \cdot c_{i+1} \leqslant p + p(p-2).$$

Отсюда имеем

$$-1 \leqslant [a_i + p \cdot c_{i+1}]_{p-1} \leqslant p.$$

Таким образом,

$$-k \leqslant b_i \leqslant p.$$

Приведем пример использования алгоритма для перевода числа $1\overline{1}\overline{1}4$ из симметричной девятеричной системы в восьмеричную (перевод $\overline{9} \to 8$), $p=9$, $k=4$:

		1	$\overline{1}$	$\overline{1}$	4
c	0	1	0	7	$\underline{3}$
b		0	1	$\overline{1}$	8
c		0	1	0	$\underline{0}$
b			0	1	1
c			0	1	$\underline{2}$
b				0	1
c				0	$\underline{1}$
b					0

Таким образом, имеем $(1\overline{1}\overline{1}4)_{\overline{9}} = (1203)_8$.

6.2. Вычисление вычетов и неполных частных по основанию $p+1$ в $(p, k, 0, k+1)$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}]_{p+1} = (b_n \ldots b_0)_{(p,\ k,\ 0,\ k+1)},$$

$$\{(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}\}_{p+1} = c_0,$$

где b_i $(i = n, n-1 \ldots, 0)$ и c_0 вычисляются по формулам

$$\begin{aligned} b_i &= c_{i+1} + [a_i - c_{i+1}]_{p+1}, \\ c_i &= \{a_i - c_{i+1}\}_{p+1}, \\ c_{n+1} &= 0. \end{aligned} \tag{10}$$

Заметим, что при $k=0$ получаем формулы (1) для случая $p+1$.

Покажем, что $(b_n \ldots b_0)$ есть натуральное представление в $(p, k, 0, k+1)$-ичной системе.

Имеем

$$-k \leqslant a_i \leqslant p,$$
$$0 \leqslant c_{i+1} \leqslant p.$$

Формулу для b_i можно записать в виде

$$b_i = [a_i + p \cdot c_{i+1}]_{p+1},$$
$$-k \leqslant a_i + p \cdot c_{i+1} \leqslant p + p \cdot p.$$

Отсюда следует

$$-k \leqslant b_i \leqslant p.$$

Приведем пример использования алгоритма для перевода числа $1\bar{1}\bar{1}4$ из симметричной девятеричной системы в десятичную (перевод $\bar{9} \to 10$), $p=9$, $k=4$:

		1	$\bar{1}$	$\bar{1}$	4
c	0	1	8	1	$\lvert\underline{3}\rvert$
b		0	0	7	1
c			0	7	$\lvert\underline{4}\rvert$
b				0	6
c				0	$\lvert\underline{6}\rvert$
b					0

Таким образом, имеем $(1\bar{1}\bar{1}4)_{\bar{9}} = (643)_{10}$.

6.3. Вычисление вычетов и неполных частных по основаниям $p \pm 1$ в $(p, k, 0, k+1)$-ичной системе

Обобщая формулы (9) и (10), можно утверждать, что

$$[(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}]_{p\pm1} = (b_n \ldots b_0)_{(p,\ k,\ 0,\ k+1)},$$
$$\{(a_n \ldots a_0)_{(p,\ k,\ 0,\ k+1)}\}_{p\pm1} = c_0,$$

где b_i $(i=n, n-1, \ldots, 0)$ и c_0 определяются по формулам

$$b_i = c_{i+1} + [a_i \mp c_{i+1}]_{p\pm1},$$
$$c_i = \{a_i \mp c_{i+1}\}_{p\pm1},$$
$$c_{n+1} = 0. \qquad (11)$$

6.4. Вычисление вычетов и неполных частных по основанию $p-1$ в $(p, к, 1, 1)$-ичной системе

Покажем, что для $k \geqslant 1$

$$[(a_n \ldots a_0)_{(p,\ k,\ 1,\ 1)}]_{p-1} = (b_n \ldots b_0)_{(p,\ k,\ 1,\ 1)},$$
$$\{(a_n \ldots a_0)_{(p,\ k,\ 1,\ 1)}\}_{p-1} = c_0,$$

где b_i $(i=n, n-1, \ldots, 0)$ и c_0 определяются по формулам

$$b_i = c_{i+1} + [a_i + c_{i+1}]_{(p-1,\ k-1)},$$
$$c_i = \{a_i + c_{i+1}\}_{(p-1,\ k-1)} \text{ для } i \neq 0,$$
$$c_{n+1} = 0, \qquad (12)$$
$$b_0 = c_1 + [a_0 + c_1]_{p-1},$$
$$c_0 = \{a_0 + c_1\}_{p-1}.$$

Следует показать, что $(b_n \ldots b_0)$ есть натуральное представление в $(p, к, 1, 1)$-ичной системе.

Имеем

$$-k-1 \leqslant a_i \leqslant p-k,$$
$$-k+1 \leqslant c_{i+1} \leqslant p-k-1,$$
$$-2k \leqslant a_i + c_{i+1} \leqslant (p-k) + (p-k-1).$$

Для $i=0$

$$-2 \leqslant [a_0 + c_1]_{p-1} \leqslant 1,$$

т. е.

$$-k-1 \leqslant b_0 \leqslant p-k.$$

Для $i \neq 0$

$$[(-k+1) + (-p+1) + (p-2-k)]_{(p-1,\ k-1)} \leqslant$$
$$\leqslant [a_i + c_{i+1}]_{(p-1,\ k-1)} \leqslant [(p-k-1) + (p-k)]_{(p-1,\ k-1)}.$$

Отсюда

$$-1 \leqslant [a_i + c_{i+1}]_{(p-1,\ k-1)} \leqslant 1,$$

т. е.

$$-k \leqslant b_i \leqslant p-k \quad (i \neq 0).$$

Приведем пример использования этого алгоритма для перевода числа $1\bar{1}\bar{1}4$ из симметричной девятеричной системы в восьмеричную (перевод $\bar{9} \to 8$), $p=9$, $k=4$:

		1	$\bar{1}$	$\bar{1}$	4
c	0	1	0	1	$\lvert\underline{3}\rvert$
b		0	1	0	1
c		0	1	1	$\lvert\underline{0}\rvert$
b			0	1	1
c			0	1	$\lvert\underline{2}\rvert$
b				0	1
c				0	$\lvert\underline{1}\rvert$
b					0

Таким образом, имеем $(1\bar{1}\bar{1}4)_{\bar{9}} = (1203)_8$.

6.5. Вычисление вычетов и неполных частных по основанию $p+1$ в $(p, k, 1, 1)$-ичной системе

Покажем, что

$$[(a_n \ldots a_0)_{(p,\ k,\ 1,\ 1)}]_{p+1} = (b_n \ldots b_0)_{(p,\ k,\ 1,\ 1)},$$
$$\{(a_n \ldots a_0)_{(p,\ k,\ 1,\ 1)}\}_{p+1} = c_0,$$

где b_i $(i = n,\ n+1,\ \ldots,\ 0)$ и c_0 определяются по формулам

$$b_i = c_{i+1} + [a_i - c_{i+1}]_{(p+1,\ k)},$$
$$c_i = \{a_i - c_{i+1}\}_{(p+1,\ k)} \text{ для } i \neq 0,$$
$$c_{n+1} = 0,$$
$$b_0 = c_1 + [a_0 - c_1]_{p+1}, \qquad (13)$$
$$c_0 = \{a_0 - c_1\}_{p+1}.$$

Заметим, что при $k=0$ формулы (13) совпадают с формулами (1) для случая $p+1$.

Покажем, что $(b_n \ldots b_0)$ есть натуральное представление в $(p,\ k,\ 1,\ 1)$-ичной системе.

Имеем

$$-k \leqslant a_i \leqslant p - k,$$
$$-k \leqslant c_{i+1} \leqslant p - k,$$
$$-k \cdot (p+1) \leqslant a_i + p \cdot c_{i+1} \leqslant (p+1)(p-k).$$

Отсюда следует, что для всех i имеем

$$-k \leqslant b_i \leqslant p - k.$$

Приведем пример использования алгоритма для перевода числа $1\bar{1}\bar{1}4$ из симметричной девятеричной системы в десятичную (перевод $\overline{9} \to 10$), $p=9$, $k=4$:

		1	$\bar{1}$	$\bar{1}$	4
c	0	1	$\bar{2}$	1	$\boxed{3}$
b		0	1	$\bar{2}$	1
c		0	1	$\bar{3}$	$\boxed{4}$
b			0	1	$\bar{3}$
c			0	1	$\boxed{6}$
b				0	0

Таким образом, имеем $(1\bar{1}\bar{1}4)_{\overline{9}} = (643)_{10}$.

7. Заключение

Приведенные алгоритмы переводов из систем с положительными и отрицательными базисными числами в ближайшие системы с неотрицательными базисными числами и обратные переводы сводятся к операциям над разрядами (сложению, вычитанию, вычислению вычетов и неполных частных). Эти алгоритмы позволяют разрешить вопросы переводов из симметричной троичной системы в десятичную и двоичную системы и обратные переводы.

В работе также даны алгоритмы перевода для систем с неотрицательными базисными числами, сводящиеся к операциям над разрядами.

Автор выражает благодарность Н. П. Брусенцову, помощь и советы которого способствовали появлению данной работы.

ЛИТЕРАТУРА

1. Карцев М. А. Арифметика цифровых машин. М., «Наука», 1969.
2. Knuth D. E. The art of computer programming, v. 2. Reading, Mass., Addison—Wesley Publ. Comp., 1969.
3. Lanning W. C. General algorithms for direct radix conversion.— «Computer Design», 1975, 14, N 6, 61—73.
4. Рамиль Альварес Х. Простые алгоритмы переводов $p \to p-1$ и $p \to p+1$. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 13. Изд-во МГУ, 1977.
5. Брусенцов Н. П. Об использовании троичного кода и трехзначной логики в цифровых машинах. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 7. Изд-во МГУ, 1970.
6. Брусенцов Н. П., Маслов С. П., Розин В. П., Тишулина А. М. Малая цифровая вычислительная машина «Сетунь». Изд-во МГУ, 1965.
7. Жоголев Е. А. Особенности программирования и математического обслуживания для машины «Сетунь». — В кн: Вычислительные методы и программирование, вып. 5. Изд-во МГУ, 1966.
8. Бондаренко Н. В. Система подпрограмм для ввода и вывода алфавитно-цифровой информации. — В кн.: Вычислительные методы и программирование, вып. 5. Изд-во МГУ, 1966.
9. Soden W. A Metod of radix conversion. — «Mathematical tables and other aids to computation», 1953, 7, N 44, 273—274.

представимый каналом связи C_m с точкой сечения $\lambda, \lambda \in [0; 1]$, множеством букв $\hat{Y}$, определяется как

$$\mathbf{L}(C_m, \lambda, \hat{Y}) = \{X^{(d)} : \sum_{Y_l \in \hat{Y}} p_c(Y_L / X^{(d)}) > \lambda\}.$$

Саломаа [7] доказал, что вероятностными автоматами с детерминированной функцией переходов представимы только регулярные языки. Из этого факта и из теоремы 1 следует теорема 4.

Теорема 4. Класс языков, представимых дискретными каналами связи с конечной памятью, совпадает с точностью до пустого слова с классом регулярных языков.

ЛИТЕРАТУРА

1. Чирков М. К. Композиции вероятностных автоматов. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 5. Изд-во ЛГУ, 1968.
2. Чирков М. К. Основы общей теории конечных автоматов. Изд-во ЛГУ, 1975.
3. Хинчин А. Я. Об основных теоремах теории информации. УМН, 1956, 11, N 1, 17—75.
4. Мачак К. Об алфавитных отображениях, индуцированных автоматными вероятностными отображениями.— В кн.: Вычислительная техника и вопросы кибернетики, вып. 12. Изд-во ЛГУ, 1975.
5. Гинзбург С. Математическая теория контекстно-свободных языков. М., «Мир», 1970.
6. Гладкий А. В. Формальные грамматики и языки. М., «Наука», 1973.
7. Salomaa A. On events represented by probabilistic automata of different types. — «Canad. J. Math.», 1968, **20**, p. 242—251.

С. П. МАСЛОВ

ПОСТРОЕНИЕ ТРОИЧНЫХ МАГНИТНЫХ ЗУ НА ОСНОВЕ СИСТЕМ *3D* и *2,5D*

Магнитные запоминающие устройства, предназначенные для хранения троично кодированной информации, могут быть осуществлены на базе специальных элементов, обладающих тремя устойчивыми состояниями. Примерами таких элементов являются биакс [1] и трансфлюксор [2]. Возможна также реализация троичного запоминающего элемента и на обычных тороидальных магнитных сердечниках. Вопросы организации троичных ЗУ на таких элементах представляют интерес, обусловливаемый недефицитностью, дешевизной и широким распространением магнитных сердечников.

Существует несколько вариантов троичного запоминающего элемента на сердечниках, различающихся количеством входящих в элемент сердечников, числом и конфигурацией обмоток и т. д. Мы будем рассматривать здесь наиболее простой вариант, в котором для хранения одного троичного разряда (трита) используется пара сердечников. Оба сердечника прошиты определенным образом возбуждающими и воспринимающими проводами, количество которых в разных типах ЗУ различно. При считывании на оба сердечника воздействуют ампервитки, достаточные для их перемагничивания. При записи необходимо в зависимости от значения записываемого трита воздействовать перемагничивающими ампервитками либо на один из сердечников пары, либо на другой, либо на оба вместе или ни на один из них. Таким образом трем значениям трита (1,0 и —1) ставятся в соответствие три различных варианта намагниченности сердечников. Воспринимающий провод проходит всегда через оба сердечника[1], причем направление прошивки таково, что при считывании 1 на нем возникает сигнал одной полярности, при считывании —1 — противоположной, при считывании 0 сигнал отсутствует. Направление намагниченности сердечников при хранении троичного 0 может, вообще говоря, быть различным. В одном

[1] Как будет показано ниже, в троичных ЗУ, подобно двоичным, возможно использование возбуждающих проводов в качестве воспринимающих. В таких ЗУ отдельный воспринимающий провод отсутствует.

случае при считывании перемагничиваются оба сердечника, в другом не перемагничивается ни один из них. И в том и в другом случае на воспринимающем проводе возникает сигнал помехи, который представляет собой разность сигналов, наводимых на участках прошивки, соответствующих каждому сердечнику. Вариант хранения 0, при котором перемагничивания сердечников не происходит, предпочтительнее, поскольку при этом удается получить более высокое значение отношения сигнал/помеха.

Существенным для такого запоминающего элемента является то, что считанный сигнал качественно различается в зависимости от значения хранимого трита. Это обусловливает построение работоспособного троичного ЗУ. То обстоятельство, что по одному проводу передается трехзначный, а не двухзначный сигнал, может способствовать более эффективному использованию оборудования в тракте воспроизведения.

Как видно из описания троичного запоминающего элемента, он, по сути дела, представляет собой сочетание двух двоичных запоминающих элементов, на основе которых реализовано большинство двоичных ЗУ. Вместе с тем известны двоичные ЗУ, выполненные на другой основе, в частности на элементах, содержащих два сердечника на бит. Такие элементы наиболее часто находили применения в ЗУ с непосредственной выборкой (так называемая система 2*D*) [3]. Устройство двоичного запоминающего элемента, содержащего два сердечника, сходно с устройством описанного здесь троичного элемента. Состояние, соответствующее хранению троичного 0, в двоичном варианте просто не используется. По-видимому, по этой причине известные реализации троичных ЗУ на элементах с двумя сердечниками на трит [4, 5] были выполнены по системе 2*D* и представляли собой модификацию соответствующих двоичных ЗУ.

Данная работа посвящена поиску и исследованию возможностей троичной модификации ЗУ на сердечниках, выполненных по системе с совпадением токов (3*D*), и широко распространенных в настоящее время ЗУ, выполненных по так называемой системе 2,5*D*.

На рис. 1 показана матрица троичного ЗУ, организованного по системе 3*D*, 4*W*. (4*W* означает, что каждый сердечник матрицы прошит 4 проводами.) Матрица имеет размер 3×6 и рассчитана на запоминание 9 тритов. Прошивка содержит систему взаимно ортогональных шин *X* и *Y*, воспринимающую шину «счит», проходящую через все сердечники, и две системы шин: «запрет *A*» и «запрет *B*», проходящих через сердечники нечетных и четных строк. Выборку того или иного запоминающего элемента осуществляют, возбуждая вертикальную и горизонтальную шины. Так, например, чтобы произвести считывание с элемента, в состав которого входят

сердечники *A* и *B*, необходимо возбудить шины X_1 и Y_1. В зависимости от значения хранимого трита при этом перемагничивается либо сердечник *A*, либо сердечник *B*, либо ни один из них. На воспринимающей шине «счит», проходящей через все сердечники в диагональном направлении, появляется считанный сигнал. Для записи возбуждают те же шины X_1 и Y_1, пропуская по ним импульсы тока, полярность которых противоположна полярности считывающих импульсов. Одновременно, в соответствии со значением записываемого трита, возбуждают либо шину «запрет *A*», либо шину «запрет *B*», либо обе шины вместе.

Из рис. 1. видно, что матрица троичного ЗУ, в отличие от своего двоичного прототипа, во-первых, не квадратная и, во-вторых, содержит нечетное число запоминающих элементов. Количество возбуждающих шин (как вертикальных, так и горизонтальных) также нечетно. Последнее обстоятельство следует из того, что целая степень тройки — нечетное число. Другая отличительная особенность троичной матрицы заключается в наличии двух систем шин запрета и соответственно двух отдельных формирователей импульсов тока запрета.

Показанная на рис. 1 матрица иллюстрирует идею объединения троичных запоминающих элементов в единый массив. Для разработки реальных ЗУ требуется также найти

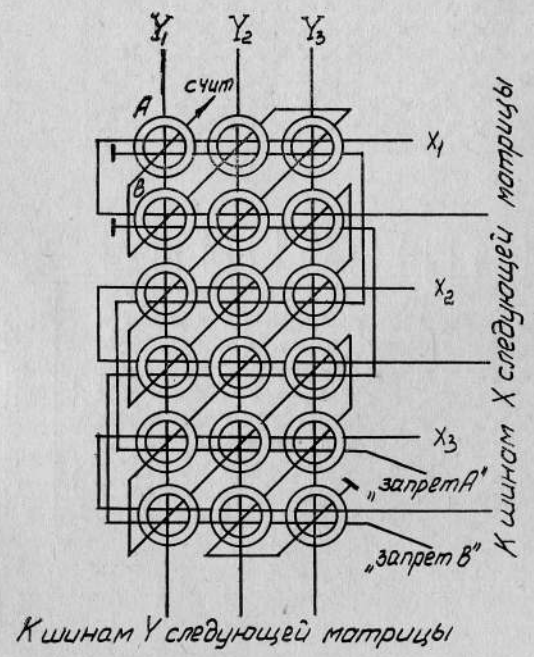

Рис. 1

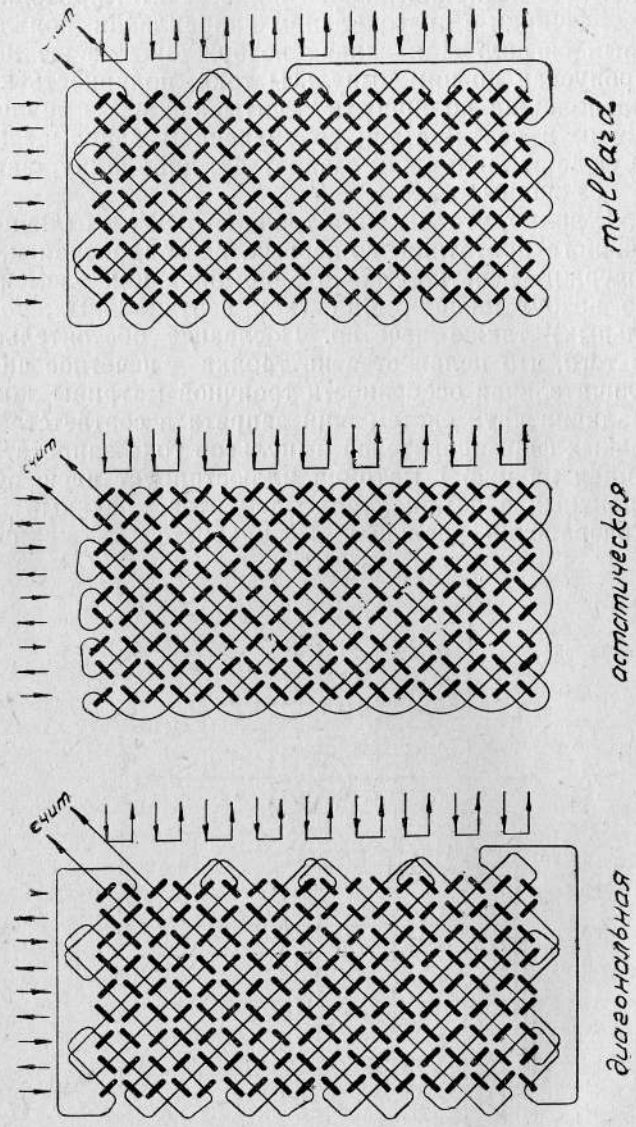

Рис. 2

такую конфигурацию воспринимающей шины (обмотки), при которой сердечники было бы удобно прошивать и, кроме того, обеспечивался бы малый уровень как внешних наводок, так и наводок от возбуждающих шин. На рис. 2 показаны некоторые возможные конфигурации воспринимающей шины троичной матрицы. Приведенные варианты получены на основе ряда известных разновидностей прошивки двоичных матриц (диагональной, астатической, прошивки типа *Mullard*). Чтобы не загромождать чертеж, на рисунке не показаны возбуждающие шины и шины запрета. Стрелки, изображенные по периферии матрицы, указывают направление тока, протекающего по возбуждающим шинам при считывании. Отметим, что в матрицах, представленных на рис. 1 и 2, имеет место компенсация помех от «полувозбужденных» сердечников. Количество «полувозбужденных» сердечников в троичной матрице вдвое превышает количество «полувозбужденных» сердечников в двоичной матрице, предназначенной для устройства с тем же числом слов, что троичное ЗУ.

В настоящее время довольно широкое распространение получили ЗУ, организованные по так называемой системе 2,5*D*. ЗУ этого типа во многом схожи с ЗУ, выполненными по системе 3*D*. Основное отличие заключается в том, что шины выборки одного из наборов (например, набора *Y*) не соединены последовательно для всех разрядных матриц, а управляются независимо в каждом разряде посредством отдельных схем. Благодаря этому отпадает необходимость в обмотках и формирователях тока запрета, упрощается прошивка и возрастает быстродействие. Существует несколько разновидностей системы 2,5 *D*, различающихся в первую очередь числом проводов, проходящих через сердечники. На рис. 3. приведен вариант троичной матрицы емкостью 27 тритов для ЗУ, организованного по системе 2,5 *D*, 3*W*. При считывании возбуждают одну шину *X* и пару шин $Y(Y_A$ и $Y_B)$. На пересечении каждой из возбуждаемых шин *Y* с шиной *X* расположены сердечники *A* и *B*, образующие троичный запоминающий элемент. На воспринимающей шине «счит» при этом возникает сигнал, по которому заключают о значении хранимого трита. Для записи 1 и —1 необходимо возбудить (импульсами противоположной полярности по сравнению со считыванием) шину *X* и либо шину Y_A, либо Y_B. При записи 0 шины Y_A и Y_B не возбуждают. В троичной матрице описанной структуры имеется возможность (по аналогии с ее двоичным прототипом) сократить количество формирователей по координате *X* за счет того, что каждая шина *X* пронизывает во взаимно противоположных направлениях две строки матрицы. Обращение к элементу, находящемуся в первой или во второй строках, осуществляют за счет изменения полярности возбуждающего тока. Для иллюстрации этого на рис. 3 ши-

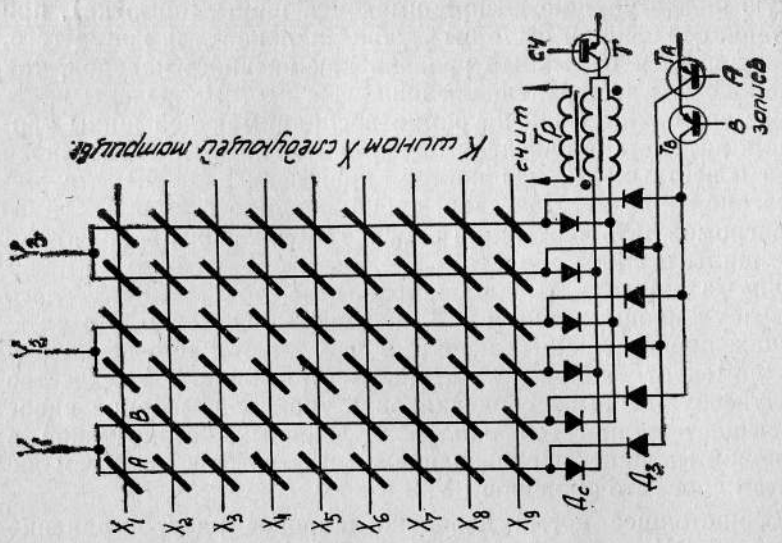

Рис. 4

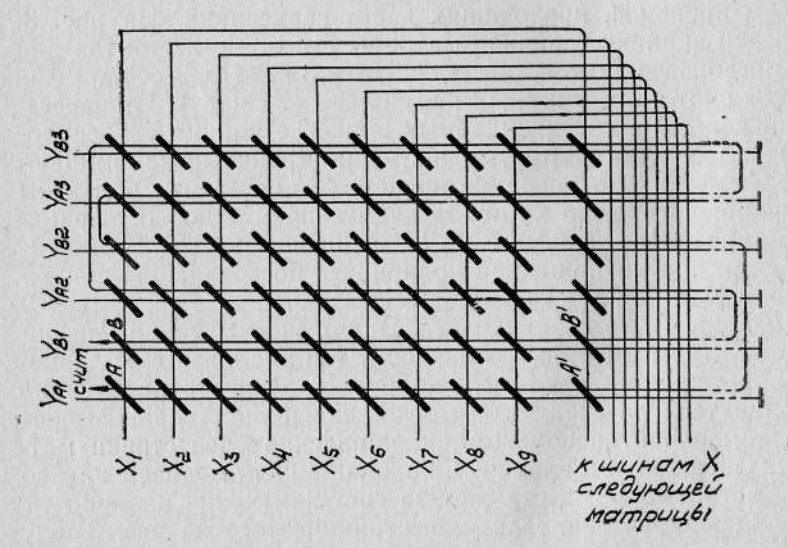

Рис. 3

на X_1 кроме строки, где находятся сердечники A и B, проходит через строку, в которой расположены сердечники A' и B'. Для одной полярности тока в X_1 имеет место обращение к элементу AB, для другой (при условии, что полярность тока в шинах Y не изменилась) — к элементу $A'B'$.

Представляет интерес возможность троичной модификации ЗУ, выполненного по системе 2,5 D; 2 W. Устройства этого типа характеризуются наиболее простой прошивкой сердечников, что обусловлено использованием всего двух шин. Матрица троичного ЗУ, осуществляющая хранение 27 тритов и реализованная на основе системы 2,5 D, 2 W, изображена на рис. 4 вместе с некоторыми элементами управляющей схемы. В отличие от матрицы, изображенной на рис. 3, в рассматриваемой матрице отсутствует воспринимающая шина, а столбцы попарно объединены. Для считывания возбуждают положительными импульсами шины X и Y, на пересечении которых находится избранный запоминающий элемент. Ток, задаваемый в шину Y, вдвое превышает ток, задаваемый в шину X. Как видно из рис. 4, шины Y подключены в параллель каждая к двум столбцам матрицы, и возбуждающий ток разделяется между ними. Для уравнивания токов, протекающих по каждому столбцу, служит балансный трансформатор $Тр$, обмотки которого при считывании через диоды $Д_С$ и транзисторный ключ T подключаются к столбцам матрицы. В результате перемагничивающие ампервитки прикладываются к обоим сердечникам запоминающего элемента. Контур, образованный двумя возбужденными столбцами матрицы, через диоды $Д_С$ связан с обмотками балансного трансформатора. С дополнительной обмотки, намотанной на сердечнике этого трансформатора, снимается считанный сигнал. Отметим, что, поскольку при считывании возбуждаются всегда два столбца матрицы, при обходе контура, образованного этими столбцами, ЭДС помех на сердечниках одного столбца имеет знак, противоположный знаку ЭДС помех, возникающих на сердечниках другого столбца. За счет этого осуществляется компенсация помех от «полувыбранных» сердечников. Число «полувыбранных» сердечников в троичной матрице, организованной по системе 2,5 D, 2 W, как и в ее двоичном прототипе, равно удвоенному числу шин X. Для записи 1 и —1 возбуждают шины X и Y импульсами отрицательной полярности, причем амплитуда тока, задаваемого в шину Y, при записи вдвое меньше, чем при считывании. Одновременно, в зависимости от значения записываемой цифры, открывают либо транзистор $T_А$, либо $T_В$. Таким образом, импульс, поданный на шину Y, проходит через диод $Д_З$ либо по одному, либо по другому столбцу матрицы. При записи 0 шину Y не возбуждают и оба сердечника остаются в том состоянии, в котором они оказались после считывания.

Говоря о количестве оборудования в троичных ЗУ, выполненных на основе рассмотренных здесь схем, можно высказать следующие соображения. Число сердечников в каждой матрице троичного ЗУ вдвое превышает число сердечников в матрице двоичного ЗУ, рассчитанного на то же количество слов. Однако общее число сердечников в троичном ЗУ лишь на 26% превысит число сердечников в двоичном ЗУ той же емкости. Это обстоятельство связано с тем, что троичное слово имеет в 1,59 раза меньше разрядов сравнительно с двоичным словом, содержащим то же количество информации. Соответственно на 26% возрастет трудоемкость прошивки матриц троичного ЗУ по сравнению с эквивалентным ему двоичным. Такая же оценка справедлива, очевидно, для оборудования формирователей токов запрета в ЗУ, выполненном по системе 3 *D*, и для оборудования, связанного с шинами *Y*, для ЗУ, выполненного по системе 2,5 *D*. Что же касается усилителей считывания, являющихся дорогой и сложной частью оборудования ЗУ, то их количество в троичном ЗУ будет в 1,59 раза меньше по сравнению с двоичным ЗУ. Сам по себе усилитель считывания троичного ЗУ несколько сложнее усилителя, предназначенного для двоичного устройства. Это обусловлено тем, что троичный усилитель должен не только фиксировать наличие или отсутствие сигнала, но и определять знак последнего.

ЛИТЕРАТУРА

1. Белоусов Б. В., Граник В. А., Мамонтов С. А., Черненко Ф. Ф. Запоминающее устройство. Авт. свид. № 222471, заявл. 17.5.67, опубл. 22.7.68.
2. Нисневич Д. Г. Троичная ячейка с неразрушающим считыванием. Авт. свид. № 173025, заявл. 5.11.63, опубл. 27.8.65.
3. Бардиж В. В. Магнитные элементы цифровых вычислительных машин. М., «Энергия», 1967, с. 132—138.
4. Брусенцов Н. П., Маслов С. П., Розин В. П., Тишулина А. М. Малая цифровая вычислительная машина «Сетунь». Изд-во МГУ, 1965, с. 84—86.
5. Маслов С. П. Комбинированное троичное запоминающее устройство. — В кн.: Вычислительная техника и вопросы кибернетики, вып. 8. Изд-во ЛГУ, 1971.

В. П. РОЗИН

ПОМЕХИ И КОНФИГУРАЦИЯ МАТРИЦ ТРОИЧНЫХ МАГНИТНЫХ ЗУ

Создание троичных ЗУ на магнитных сердечниках может быть основано на использовании запоминающего элемента (ЗЭ) из двух сердечников для хранения одного троичного разряда (трита). Функционирование такого ЗЭ предполагает: а) независимое управление каждым сердечником пары при записи и б) различную полярность ЭДС сигнала при считывании +1 и —1 либо отсутствие этой ЭДС при считывании 0.

В настоящем выпуске [1] приведены и описаны схемы матриц, на основе которых может быть построено несколько типов троичных ЗУ. Такие ЗУ имеют определенную специфику в сравнении с аналогичными двоичными устройствами. В данной работе проводится анализ этой специфики в вопросах о помехах и о конфигурации матриц.

1. Помехи в троичных ЗУ

Распознавание считанного сигнала в ЗУ на магнитных сердечниках встречает затруднение, в силу того что в воспринимающей обмотке, кроме ЭДС сигнала с выбранного ЗЭ, возникают ЭДС помех. Основная доля этих помех индуцируется сердечниками, расположенными вдоль выбранных строки и столбца матрицы ЗЭ. (Эти сердечники будем в дальнейшем называть полувыбранными, так же как и образованные ими ЗЭ.) Эффективной мерой снижения таких помех является применение специальных способов прошивки матриц воспринимающим проводом. Для всех этих способов прошивки характерно, что одна половина сердечников прошивается этим проводом иначе, чем другая. Таким образом, помехи от сердечников одной и другой половин имеют противоположные полярности и взаимно компенсируются. Известны компенсационные воспринимающие обмотки двух основных типов: диагональные, для которых характерна прошивка сердечников в направлении диагональных рядов, и прямоугольные, в которых воспринимающий провод проходит параллельно сторонам матрицы.