**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
«__» _____ 2017 г.

# МОДЕЛИРОВАНИЕ РАБОТЫ АССОЦИАТИВНОГО ЗАПОМИНАЮЩЕГО УСТРОЙСТВА

Методические рекомендации по выполнению
лабораторной работы №3для студентов специальности 10.03.01

Курск 2017

# Лабораторная работа №3

## Моделирование работы ассоциативного запоминающего устройства (АЗУ)

**Цель работы:** изучить структуру ассоциативного запоминающего устройства (АЗУ), режимы записи, считывания информации, методы адресации. По структурной схеме создать модель ассоциативно-запоминающего устройства (АЗУ), выполняющего поисковые операции, применяя известные методы адресации информации.

**Задача:** По представленной структуре ассоциативного запоминающего устройства (АЗУ) разработать блок-схемы алгоритмов, перевода ключевых слов в числовую форму, а также осуществляющих поисковые операции и моделирующие преобразование числовых значений в набор хеш-адресов. Протестировать программы на языке высокого уровня.

## 1. Теоретическая часть

Ассоциативная память - это массив ограниченного размера из строк памяти. Строка памяти содержит адрес A, данные D и надежность R. Рассмотрим только битовую (логическую) память. Она устроена несколько сложнее, чем аналоговая. Адрес и данные характеризуются количеством бит или байт, а надежность - это число от 0 до 1. При записи новая строка ADR заносится на место одной из наименее надежных строк. Ассоциативная память характеризуется тем, что чтение из нее возможно, даже если в ней нет строки с нужным адресом A. Результат чтения из памяти есть вероятностная сумма всех данных, рассчитанная по их надежности R и близости адреса A к заданному адресу. Можно определять близость адреса сортировкой. Можно по количеству отличающихся бит или байт.

Каждый несовпадающий бит в адресе $A_k$ снижает вес $W_k$ (например, вдвое). В итоге веса используются для определения вероятности того, что соответствующий бит данных D совпадет с битом данных D из строки k.

Наличие ассоциативной таблицы перекодировки (памяти) в нейроне создает два уровня обобщения. Первое обобщение возникает внутри нейрона при совпадении строк памяти. В этом случае мы повышаем надежность одной из строк и снижаем до 0 надежность второй, переводя ее, таким образом, в резерв для новых дан-

ных. Поскольку одна таблица может использоваться несколькими нейронами, второе обобщение возникает при совпадении таблиц. Тогда одна из них становится общей и для нее повышается надежность, а вторая переходит в резерв для проверки других общих таблиц. Если резерв таблиц становится слишком большим, можно добавить новые (скрытые) нейроны. Это должна позволять топология сети.

Как только дело доходит до программирования, становится ясно, как долго будет думать однопроцессорная нейронная сеть. Ведь для получения выходных данных от одного нейрона нужно использовать всю ею память полностью.

Для поиска двух совпадающих строк в таблице нужно выполнить двойной цикл по строкам. Для поиска совпадающих таблиц в сети нужен двойной цикл по таблицам. Поэтому алгоритм должен быть очень эффективным. Например, надо сразу отказаться от числа входов, выходов, размера таблиц и т.д. не кратных 8. Нейрон должен иметь минимум 1 байт на выходе и 2 байта на входе (если нет внутренней обратной связи). Таблица перекодировки должна иметь размер в 256 или 256x256 байт. Другие значения неудобны для быстрой обработки. Передача данных между нейронами тоже должна выполняться байтами, а не битами. То есть, все 8 входных бит одного нейрона должны принимать информацию от 8 выходных бит другого нейрона. Байты не расщепляются. Логически, это не имеет значения, а расчет ускорится. Тем не менее, выбор и сравнение данных между ассоциативными строками происходит побитно. Для ускорения можно не делать полные циклы. Поскольку ассоциативная таблица перекодировки не полна, то нейрон может "терять" входную информацию от датчиков общения с внешним миром. Чтобы сгладить потери, надо подавать (дублировать) входную информацию сразу нескольким нейронам.

В настоящее время можно выделить два основных подхода к проблемам ассоциативной памяти.

Первый из них, распространенный среди разработчиков ВТ, заключается в создании новых принципов организации и/или управления памятью, именуемых адресацией по содержанию, согласно которым поиск информации осуществляется исходя из ее содержимого, а не по месту ее расположения в памяти.

Второй подход является более абстрактным: память трактуется как некоторое семантическое представление знаний, описывае-

мое с помощью реляционных структур. Обстоятельства, способствующие развитию запоминающих устройств с адресацией по содержанию.

Человечество столкнулось с проблемой "информационного взрыва". Для ее решения предлагалось ввести такую систему хранения документов и данных, которая основывалась бы не на их нумерации, а на формировании "ассоциации". В связи с изобретением в 1955-1960гг. адаптивных или обучающихся устройств, в том числе перцептронов, высказывались предположения о возможности воспроизведения интеллектуальных функций и создания систем, основанных на принципах самоорганизации. Первоначально перцептронные системы именовались "ассоциативными".

Вследствие выяснилось, что эффективность обработки информации в указанных системах была недостаточна для практического воплощения искусственного интеллекта. Ввиду этого было признано для реализации информационных процессов средства, предоставляемыми машинными языками высокого уровня.

## Ассоциативное запоминающее устройство

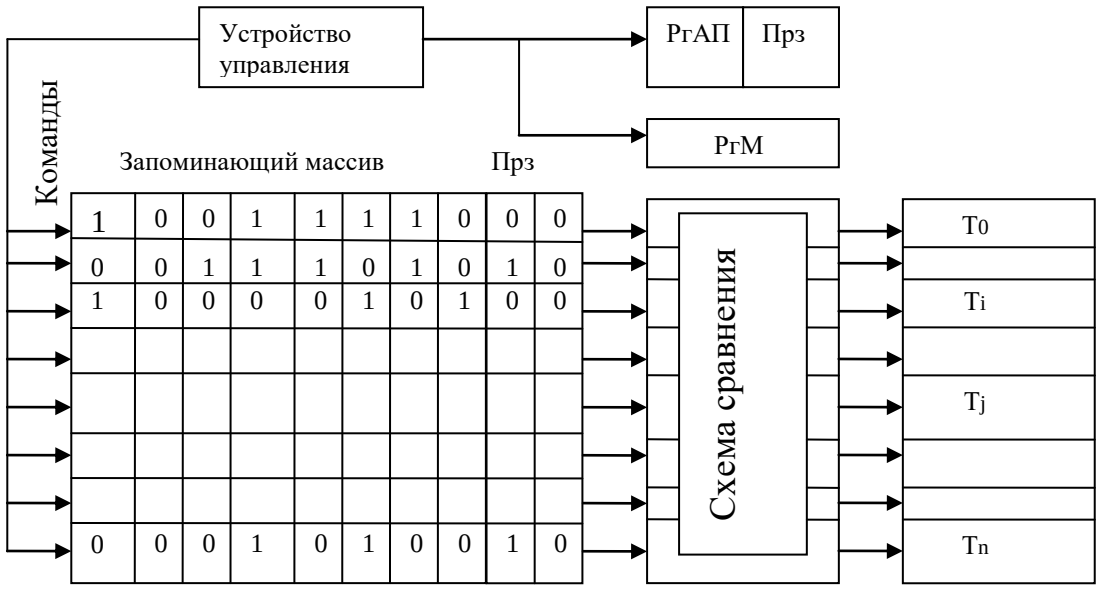



Рис. 1. Ассоциативное запоминающее устройство

В состав ассоциативного запоминающего устройства (АЗУ) входят:

- запоминающий массив;
- регистр ассоциативных признаков Рг АП;
- регистр маски Рг М;
- регистр индикаторов адреса со схемами сравнения на

входе.

**Операция записи в АЗУ производится с учетом следующих правил:**

1. При записи не указывается номер ячейки.
2. Один из разрядов каждой ячейки используется для ее занятости. Если ячейка свободна, признак занятости (Прз) равен нулю, занята - 1.
3. При записи новой информации Прз=0 в соответствующем разряде РгАП и определяет все свободные ячейки. В одну из этих ячеек устройство управления (УУ) помещает новую информацию.

**Выборка из АЗУ производится следующим образом:**

1. Из устройства управления (УУ) в РгАП передастся код признака искомой информации. Код - произвольное число разрядов от 1 до m.
2. Код поступает на схему сравнения, если он используется полностью.
3. Если необходимо использовать часть кода, то разряды маскируются с помощью РгМ.
4. Перед работой все разряды регистра индикаторов адреса устанавливается в 1. После этого производится опрос первого разряда всех ячеек (строк) запоминающего массива, их содержимое сравнивается с содержимым РгАП. Если первый разряд i-той ячейки не совпадает с содержимым РгАП. то соответствующий этой ячейке разряд регистра индикатора адреса сбрасывается в 0, если содержимое ячейки и РгАП совпадают, то данный разряд остается равным 1. Операция сравнения повторяется с остальными строками до тех пор, пока не будет произведено все сравнение с РгАП. В результате сравнения останутся равными 1 те регистры индикаторов адреса, которые соответствуют ячейкам, содержащим информацию, совпадающую с записанной в РгАп.

## 1.1 Основные определения и концепции

Память с адресацией по содержанию (ПАС) называется запоминающее устройство, состоящее из ячеек, а которых хранятся данные. Выборка и загрузка в эти ячейки производится в зависимости от содержащейся в них информации. Процессором с адресацией по содержанию называется адресация по содержанию память, кото-

рая допускает выполнение сложных преобразований информации: последняя хранится в некоторой совокупности ячеек, выбираемых в соответствии с их содержанием.

**Рассмотрим вариант ПАС.** ПАС состоит из двух частей: справочника и памяти данных (рис.2). Подобная конфигурация допускает только сравнение ключевой информации (ключевое слово) одновременно со всеми словами, записанными в справочнике. Каждая ячейка справочника представляет собой регистр, рассчитанный на хранение одного слова. Он дополнен специальными комбинационными логическими схемами, которые предназначены для сравнения содержимого регистра с ключевым словом, поступающим одновременно на вход всех ячеек. В каждой ячейке имеется выходная линия, которая возбуждается при совпадении записанного в ней слова с ключевым.

Возбуждение может затронуть не более одной линии. Память данных представляет собой обычную память произвольного доступа с линейной выборкой. Выходные шина справочника выполняют функцию адресных линий, следовательно, необходимость в дешифраторе адреса отпадает. Время срабатывания (время доступа к ПАС) составляет 50 нс.

Особое значение для массовой обработки информации имеет базовая операция ПС. На основе этой операции строятся параллельные процессоры наиболее известного класса - ассоциативные параллельные процессоры (АПП). Большое внимание, уделяемое АПП, объясняется, во-первых, сравнительной простотой аппаратной реализации поиска по совпадению, а, во-вторых, тем, что существуют достаточно эффективные алгоритмы, основанные на использование этой операции.

В процессе развития АПП при разработке методов массовой обработки создавались алгоритмы для решения как информационно-логических, так и вычислительных задач. Рассмотрим массовые операции поиска.

Аргументами каждой операции поиска служат массив информации, элементами которого являются двоичные слова (строки), и одна или две специальные двоичные кодовые комбинации, определяющие условия поиска. Результат операции поиска - некоторое подмножество элементов исходного массива (строк), в которое входят все элементы, удовлетворяющие заданному условию поиска.

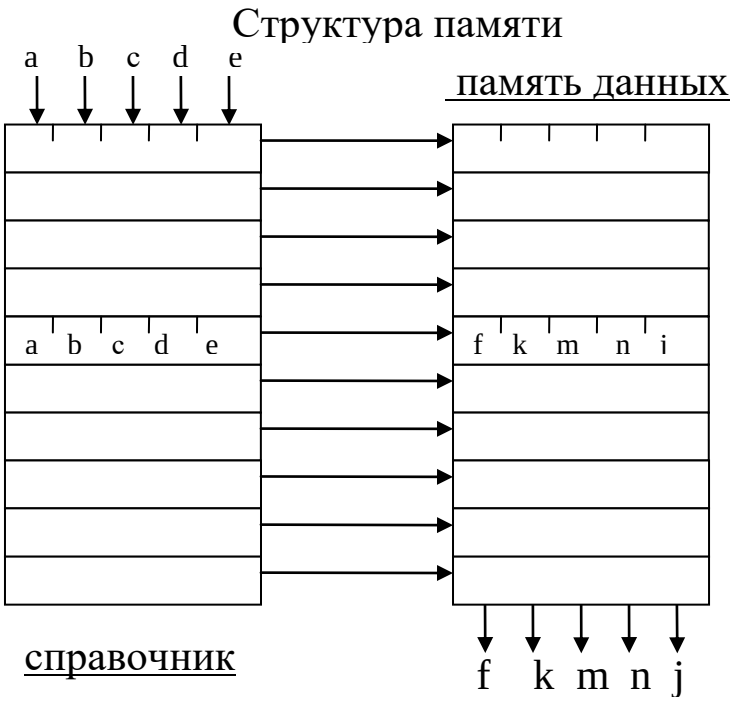



Рис.2 Структура памяти с адресацией по содержанию (ПАС)

Поиск ведется не по всему элементу массива информации, а по некоторой его части (полю), называемой признаком. Перечислим основные операции поиска:

**1. Поиск по совпадению.** Задается некоторая кодовая комбинация - эталон X. Результат - все элементы, признаки которых $A_i$ совпадают с эталоном.

**2. Поиск по интервалу.** Задаются эталоны $X_B$ и $X_H$, соответствующие верхней и нижней границами интервала. Результат - все элементы, признаки которых $A_i$ (в данном случае они должны рассматриваться как числа) удовлетворяют условию $X_h>A_i>X_b$ (либо $X_h<A_i<X_b$).

**3. Поиск всех больших.** Если задан только эталон, соответствующий нижней границе, то операция сводится к поиску всех элементов, признаки которых больше (не меньше) эталона $A_i>X_b$ ($A_i \geq X_h$).

**4. Поиск всех меньших.** Если задан только эталон, соответствующий верхней границе, то операция сводится к поиску всех элементов, признаки которых меньше (не больше) эталона $A_i>X_b$ ($A_i \geq X_h$).

**5. Поиск ближайшего числа.** Задается эталон X. Признаки и эталон рассматриваются как числа. Результат операции - элемент, признак которого A отличается от X меньше, чем признаки всех других элементов. Иначе говоря, абсолютная величина разности $\Delta=X-A_i$

минимальна.

**6. Поиск максимума**, если в качестве эталона X выбрать число, большее чем максимально возможное значение признака.

**7. Поиск минимума**, если эталон X меньше, чем минимально возможное значение признака.

## 2. Хеширование

Идея хеширования заключается в том, что элемент данных заносится в память по адресу, который легко вычислить, зная содержимое ключевого слова, присвоенного этому элементу.

Хеширование является самым быстродействующим из известных методов программного поиска. Данный метод весьма удобен тем, что не требует никакого упорядочивания, ни сортировки ключевых слов. Недостатком этого метода является то, что расходуется несколько больше памяти, нежели при использовании других методов поиска.

На выбор ключевых слов не накладывается практически никаких ограничений. В качестве ключевых можно использовать обычные имена или произвольные числовые коды, причем к ним не требуется добавлять какие-либо контрольные ветки или символы. Длина ключевых выбирается произвольно. В вычислениях участвует обычно несколько первых символов.

Преобразование ключевых слов в допустимые адреса памяти выполняется с помощью некоторой функции хеширования. Содержимое ключа мы обозначим через К, а число h (К) назовем вычисленным адресом или хеш-адресом. Если двум различным ключам, $К_1$ и $К_2$, соответствуют одинаковые адреса $h(К_1) = h(К_2)$, то говорят, что возникла коллизия (конфликт). Коллизии всегда устраняются одним и тем же способом - по вычисленному или резервному адресу. Процесс формирования хеш-функции включает обычно два этапа: выбор способа перевода ключевых слов в числовую форму; выбор алгоритма преобразования числовых значений в набор хеш-адресов.

### 2.1 Программная реализация

Имя в памяти хранится в виде двоичного кода, ему можно сопоставить некоторое число. На практике адрес всегда намного уже того диапазона чисел, в который преобразуются все имена.
Необходимы какие-то способы сжатия.

Пример.

Численное значение имени определяется его двумя первыми буквами

(31 буква русского алфавита)

$31^2 = 961$ различных комбинаций из двух букв.

Пусть А = 0, Б = 1,...Я = 33

каждой паре букв присваивается целое число, записанное по основанию 33.

Пример.

$$БА = 1*33^1 + 0*33^0 = 33$$

Вычисление адреса преобразования в с/с 33.

Если конфликт, то необходимо подобрать такую резервную ячейку, которую можно было бы легко отыскать. Следующую пустую ячейку.

## Перевод ключевых слов в числовую форму

Всякая информация, вводимая в ЭВМ, должна быть закодирована. Чаще всего для этой цели используется код ASCCI (Американская стандартная кодировка), в которой все символы представлены восемью двоичными цифрами (битами), образующими один байт. При проведении арифметических преобразований это число выступает в качестве основания системы счисления.

Предположим, что имеется алфавит, в котором каждому символу i согласно некоторому правилу поставлен в соответствие номер $d_i \in \{0, 1, 2, \ldots, w\text{-}1\}$ . Тогда строку, образованную из этих символов и записанную в форме $k = d_N d_{N-1} \ldots d_1 d_0$ можно рассматривать как представление некоторого целого числа в системе с основанием w.

Его значение определяется по формуле

$$v = \sum_{i=0}^{N} d_i w^i$$

**Метод деления**

Предположим, что хеш-адрес должен целиком занимать область памяти B...B+H, располагаясь в ней последовательно.

Функцию хешированияh(v) можно определить как

$$h(v) = v \bmod H + B$$

где v = v (K) - числовое представление ключа, [(v mod A) - остаток от деления v на A]

Метод деления является наиболее распространенным методом вычисления хеш-функций. Но H - задается как константа и число нечетное. Длина таблицы не должна выражаться степенью основания, по которому производится перевод ключей в числовую форму.

Длина таблицы - простое число (25, 27, 31). Особое внимание, когда размер таблицы хеширования является степенью основания машинной системы счисления (n=2,10).

При этом

$$v \bmod H = d_N d_{N-1} \ldots d_1 d_0$$

$d_N d_{N-1} \ldots d_1 d_0$ - младшее значение цифры в машинном представлении v. Они объединяются с B и задают хеш-адрес.

Помнить, что преобразование ключевых слов в числовую форму должно производиться по основанию, не равному степени основания с/с, которая реализуется в применяемой ЭВМ.

**Метод извлечения битов**

Один из старейших методов и самый простой. Ключевое слово рассматривается как битовая строка. Двоичное число, соответствующее хеш-адресу, образуется просто путем сцепления нужного количества битов, извлекаемых из определенных позиций внутри указанной строки. Число таких битов должно быть достаточным для адресации всех элементов таблицы хеширования:

(1 и 0) должны появляться с приблизительно равной частотой.

Битовую строку условно делят на сегменты (двоичное кодирование). Отбирают хеш-адрес тот, в котором цифры распределяются наиболее равномерно.

**Метод квадрата**

В соответствии с этим методом числовое значение ключевого слова v возводится в квадрат, после чего биты хеш-адреса извлекаются из средней части результата.

Необходимо проверять, не содержит ли величина v чрезмерное количество нулей.

При реализации тех же операций поиска в универсальной ЭВМ каждая из них обычно представляет собой процедуру, для которой составляется стандартная программа. В связи с этим рассматриваемые процессоры можно назвать аппаратными процедурными модулями.

В отличие от традиционных ЭВМ здесь элементарными операциями являются групповые, массовые обработки. Так, при упо-

рядочении массива в универсальной ЭВМ основной операцией является сравнение двух чисел, а в системе, использующей специализированные процессоры, в качестве основной можно применить, например, такую мощную базовую операцию, как поиск максимального числа в массиве. Этим и определяется высокая эффективность обработки.

## 2.2 Процессор, ориентированный на операцию поиска максимума

Рассмотрим двумерную однородную структуру размером N×*m* (рис.3), в запоминающих элементах которой записаны m-разрядные признаки N элементов обрабатываемого массива таким образом, что каждый признак занимает одну строку матрицы, а одноименные разряды всех признаков расположены в одноименных столбцах матрицы (старшие разряды—слева).

Для поиска максимума используется известный алгоритм поразрядного сравнения всех признаков, который состоит в следующем:

**1-й шаг.** Просматривается содержимое запоминающих элементов левого (первого) столбца, т. е. старшие разряды всех *N* признаков. Если все эти разряды содержат нули, то на следующем шаге просматриваются вторые разряды всех N признаков. Если же в первом столбце имеются как нули, так и единицы, то на втором шаге просматриваются только те признаки, которые имели в первом разряде единицы.

**j-й шаг**. Просматривается содержимое запоминающих элементов j-го столбца (j-е разряды признаков) в тех строках, которые были выделены на (j-l)-м шаге. Если все эти разряды содержат нули, то на следующем шаге просматриваются (j+l)-е разряды тех же самых строк. Если в просматриваемых на j-м шаге запоминающих элементах имеются как нули, так и единицы, то на (j+l)-м шаге просматриваются только строки, соответствующие единицам. Выделенное на последнем (*m*-м) шаге подмножество строк (в частном случае - одна строка) содержит максимальные признаки.

Покажем, как можно реализовать описанный алгоритм с помощью двумерной итеративной сети с двумя направлениями распространения межэлементных сигналов.

В горизонтальном направлении необходимо иметь цепь, просматривающую последовательно, слева направо, содержимое запо-

минающих элементов и продолжающую этот просмотр, если в данном запоминающем элементе содержится единица, либо если все просматриваемые в данном столбце запоминающие элементы содержат нули. Эту работу может выполнить двоичный канал, реализующий в каждой ячейке итеративной сети логическую функцию

$$z' = z(a \vee y),$$

где *z′*—сигнал на боковом выходе ячейки; z—сигнал на боковом входе; а— содержимое запоминающего элемента (сигнал на внешнем входе); *y*— переменная, характеризующая содержимое просматриваемых ячеек данного столбца: y=1, если все они содержат нули и 0- в противном случае.

Вся система логических функций ячейки итеративной сети, реализующей поиск максимума, имеет следующий вид:

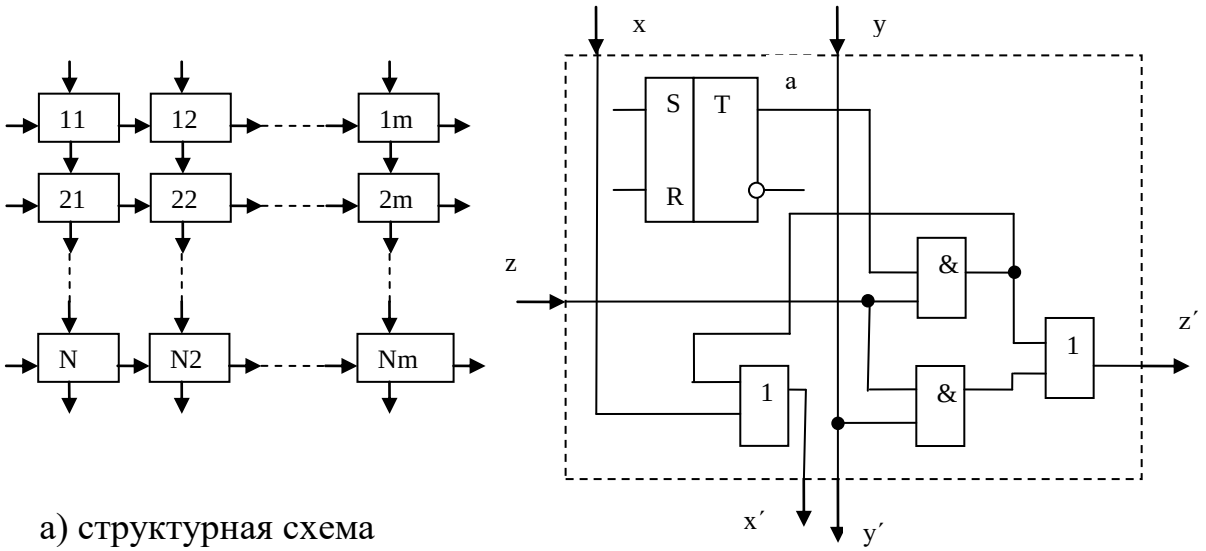
z′ = z(a V y) – горизонтальный канал;
x′ = x V ay
y′ = y_ } - вертикальные каналы;
$y_0 = \overline{x}'_N$ вспомогательная функция.

Процессор для поиска минимума можно построить аналогичным способом. Чтобы путем параллельного поразрядного сравнения выделить минимальные признаки, необходимо в процессе просмотра строк продолжать их просмотр, если в запоминающем элементе содержится нуль либо если все просматриваемые в данном столбце запоминающие элементы содержат единицы.

Эти функции может выполнить та же итеративная сеть, которая построена для поиска максимума, если на ее внешние входы подать отрицания переменных, хранимых в соответствующих запоминающих элементах. Тогда система логических функций приобретает вид:

$$z' = z(\overline{a} \vee y);$$
$$x' = x \vee \overline{a}z;$$
$$y' = y$$
$$y_0 = \overline{x}'_N .$$

Для краткости процессор для поиска максимума (наибольшего) будем называть $а_Б$-процессором, а для поиска минимума (наименьшего) - $а_м$-процессором.




а) структурная схема

б) функциональная схема ячейки

Рис.3 Процессор, ориентированный на операцию поиска максимума ($\alpha_Б$ – процессор).

## 2.3 Процессор, ориентированный на операцию поиска всех больших и всех меньших чисел

Общая структура этого процессора и размещение признаков такие же, как у предыдущего, однако, в данном случае необходимо иметь регистр $X$ для граничного признака (эталона) и дополнительный «проходной» вертикальный канал в каждом столбце для ввода во все ячейки данного столбца соответствующего разряда эталона (рис. 4).

Используется следующий алгоритм. В каждой строке (одновременно во всех строках) производится поразрядное сравнение признака $A_i = a_{i1}, a_{i2} \ldots a_{im}$, хранимого в запоминающих элементах строки, с эталоном $X = x_1,x_2 \ldots x_m$. Для каждого разряда возможны три ситуации:

$$a_{ij} = x_i, \quad a_{ij} < x_i, \quad a_{ij} > x_i.$$

Просмотр начинается со старших разрядов. Если во всей строке не встретилось ни одного неравенства, значит, $A_i=X$. Если первым встретилось неравенство $a_{ij}<x_i$, $A_{ij}<X$ независимо от результатов дальнейшего сравнения, поскольку все дальнейшие (младшие) разряды имеют меньшие веса, чем j-й. Аналогично, если первым встретилось неравенство $a_{ij}>x_i$, $A_{ij}>X$.

Для реализации этого алгоритма необходимо организовать в каждой строке процессора одновременную итеративную сеть с одним направлением распространения межэлементарных сигналов, выполняющую описанный выше просмотр. Один из возможных ва-

риантов – двухканальная сеть, в каждой ячейке которой реализуются логические функции:

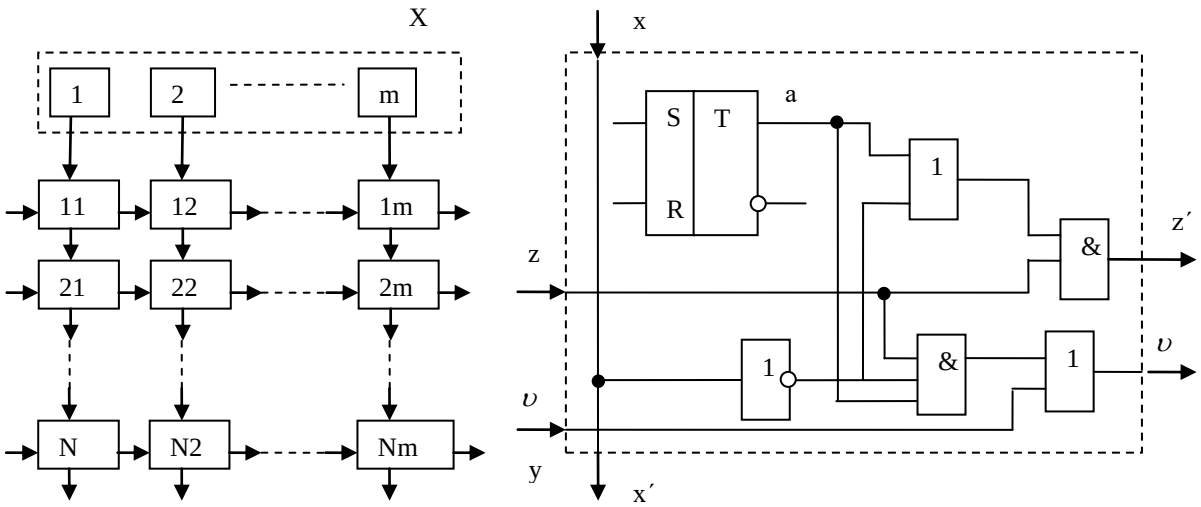
$$z' = z(\overline{a} \vee y);$$
$$v' = v \vee za\overline{x},$$

где *z*, *z'*, *v*, *v'* - соответственно входные и выходные сигналы каналов *z* и *v*; a – содержимое запоминающего элемента; *x* – сигнал в вертикальном канале (знание соответствующего разряда эталона).

Система логических функций ячейки итеративной сети, реализующей поиск всех больших и всех меньших чисел, имеет вид:

$$z' = z(a \vee \overline{x});$$
$$v' = v \vee za\overline{x},$$
$$x' = x$$

На рис. 4 приведена функциональная схема ячейки процессора, ориентированный на операцию поиска всех больших и всех меньших чисел (ε-процессор).




а) структурная схема

б) функциональная схема ячейки

Рис.4. Процессор, ориентированный на операцию поиска всех больших и всех меньших чисел (ε – процессор)

**2.4 Процессор, ориентированный на операцию поиска по интервалу.**

Общая структурная схема этого процессора такая же, как у рассмотренных выше, но имеются два внешних регистра *(X* и У) для хранения эталонов, соответствующих верхней и нижней границам интервала (рис. 5) и два вертикальных «проходных» канала.

Используется следующий алгоритм: в каждой строке производится поразрядное сравнение (начиная со старших разрядов) при-

знака $A_i=a_{i1}, a_{i2},\ldots a_{im}$ с верхним эталоном $X=x_1,x_2,...x_m$ и одновременно сравнение $A_i$, с нижним эталоном $Y=y_1y_2...y_m$ - Если при сравнении $A_i$ с $X$ первым встретилось неравенство $a>x$, то $A_i>X$, т. е. результат поиска отрицательный. Если при сравнении $A_i$, с $Y$ первым встретились неравенство $a<y$, $A_i<Y$, что также соответствует отрицательному результату поиска. В остальных случаях результат положительный, т. е. признак $A$, находится в заданном интервале:

$$Y \le A_i \le X.$$

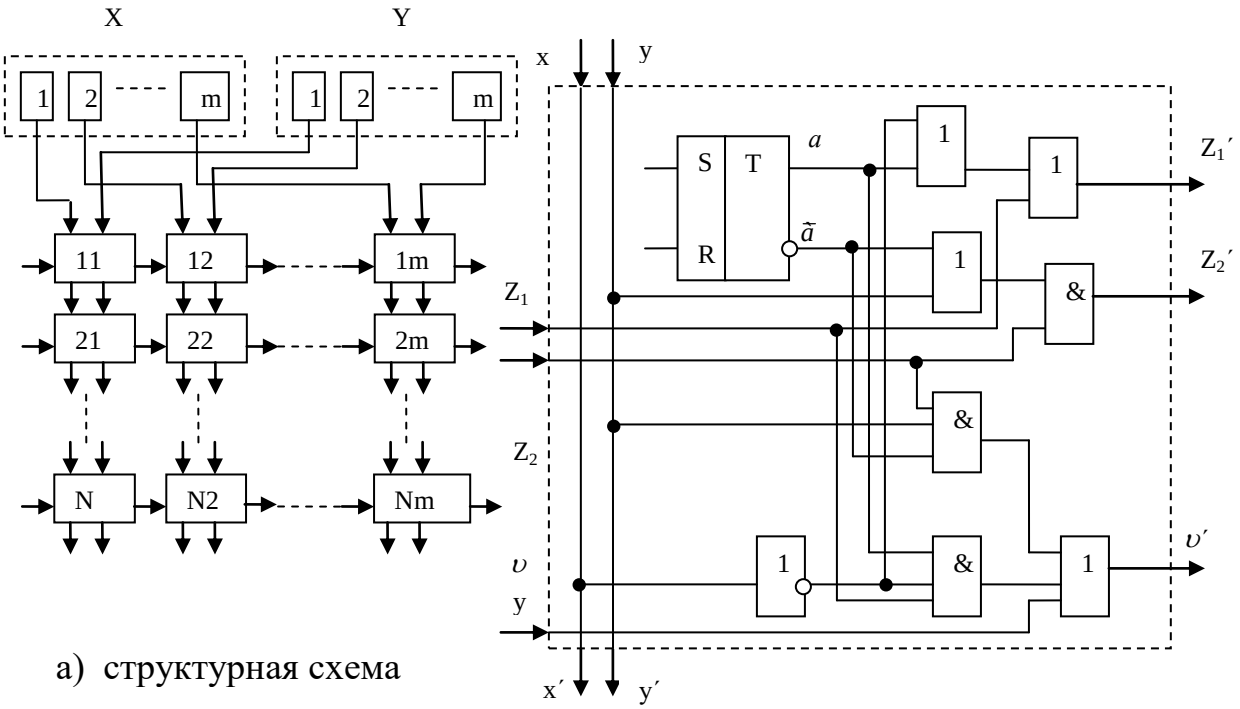
Реализовать этот алгоритм можно, например, используя свойства рассмотренных ранее каналов ε-процессоров. Если организовать канал $z' = z(a \vee \bar{x})$, то сигнал $z_1=1$ будет сохраняться в нем до тех пор, пока a ≥ x. Во втором канале $z' = z(\bar{a} \vee y)$ сигнал $z_2$=l будет сохраняться до тех пор, пока $a \le y$. Тогда результат поиска можно выработать с помощью третьего канала $v' = v \vee z_1 a\bar{x} \vee z_2\bar{a}y$. При граничном сигнале $v_0=0$ $v'$ принимает значение 1 только в тех случаях, когда в процессе сравнения появляется ситуация $a > x(z_1 a\bar{x} = 1)$ либо $a < y(z_2\bar{a}y = 1)$. Такие ситуации соответствуют отрицательному результату поиска. Если же на правой границе матрицы $v'=0$, то признак данной строки находится в заданном интервале. Система логических функций ячейки имеет вид:

$$z_2' = z_{12}(\bar{a} \vee y);$$
$$z_1' = z_1(a \vee \bar{x});$$
$$v' = v \vee z_1 a\bar{x} \vee z_2\bar{a}y;$$
$$x' = x;$$
$$y' = y.$$

На рис. 5 приведена функциональная схема ячейки.

Процессор для поиска по интервалу ($\varepsilon_И$ -процессор) обладает большими функциональными возможностями, чем $\varepsilon_Б$ и $\varepsilon_М$ - процессоры. Так, если в качестве нижнего граничного признака $Y$ использовать число, меньшее чем минимально возможное значение признака массива, то $\varepsilon_И$ - процессор выделит все $A_i \le X$. Если в качестве верхнего граничного признака $X$ использовать число, большее чем максимально возможное значение признака, то будут выделены все $A_i \le Y$.

Если задать $X=Y=Z$, то будет выполняться поиск по совпадению $A_i=Z$.




а) структурная схема

б) функциональная схема ячейки

Рис.5 Процессор, ориентированный на операцию поиска по интервалу ($\varepsilon_И$ –процессор)

## 2.5 Процессор, ориентированный на операцию поиска ближайшего числа

Рассмотрим сначала случай поиска ближайшего большего. Сравнивая эталон *X* с признаками массива (одновременно во всех строках процессора), отметим в каждой строке разряд с первым встретившимся неравенством *a>x*. Предположим, что во всех отмеченных строках отметки 'казались в различных столбцах. Тогда достаточно отыскать ту строку, в которой отметка расположена правее всех остальных (т.е. в разряде с наименьшим весом). Очевидно, результат будет находиться именно в этой строке, так как из всех разностей *A, - X* разность в данной строке минимальна.

В общем случае в правом столбце может оказаться несколько отметок. Это означает, что в соответствующих строках находятся числа, для которых разности *Ai -X* не превосходят $2^l$ (где *l*—номер правого столбца). Ясно, что результатом поиска должно быть минимальное из этих чисел. Поскольку в старших разрядах (до *l-го* включительно) все рассматриваемые числа равны, достаточно определить минимум по младшим *l*-1 разрядам. Выделенная таким образом строка и содержит искомый результат – число, ближайшее большее к эталону *X*.

Первый этап описанного алгоритма можно реализовать например, с помощью каналов ε-процессоров. Полная система логических функций, ячейки имеет вид:

$$z' = z(\bar{a} \vee x); \qquad s' = s$$
$$v' = v(a \vee s)t \vee z\bar{a}x; \qquad t' = t;$$
$$y' = y \vee z\bar{a}x; \qquad t_0 = \bar{y}'_N;$$
$$r' = r \vee av; \qquad s_0 = \bar{r}'_N;$$
$$x' = x$$

Значения граничных сигналов следующие: $z_0$=l, $v_0$=0, $y_0$=0, $r_0$=0. Результат поиска определяется по значению сигнала $v'$ на правой границе матрицы. $v'$=l означает, что в данной строке содержится число, ближайшее большее к заданному.

На рис. 6 приведена функциональная схема ячейки.

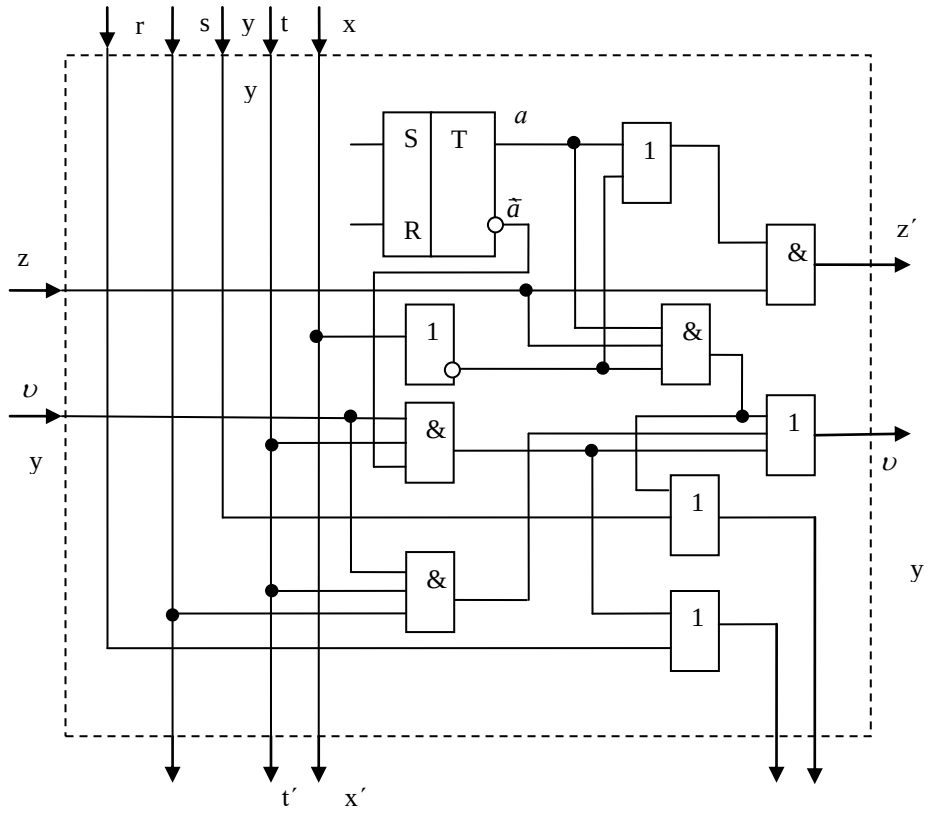



Рис.6 Процессор, ориентированный на операцию ближайшего числа

Поиск ближайшего меньшего выполняется аналогично.

Сначала выделяются все числа, меньшие эталона. Среди них определяются те, для которых разности $X—A_i$ не превосходят $2^l$, затем находится максимальная из этих разностей.

В соответствии с этими операциями здесь используются каналы, аналогичные каналам $\varepsilon_M$-и $\varepsilon_Б$ - процессоров. Получается сле-

дующая система логических функций ячейки:

$$z' = z(\bar{a} \vee x); \qquad s' = s$$
$$v' = v(a \vee s)t \vee z\bar{a}x; \qquad t' = t;$$
$$y' = y \vee z\bar{a}x; \qquad t_0 = \bar{y}'_N;$$
$$r' = r \vee av; \qquad s_0 = \bar{r}'_N;$$
$$x' = x$$

Значения граничных сигналов следующие. $z_0$=l, $v_0$=0, $y_0$=0, $r_0$=0.

Чтобы осуществить поиск ближайшего по абсолютной величине, необходимо отметить все строки, содержащие числа, большие эталона и меньшие его. Затем выделить строки, имеющие отметки в самом правом столбце.

Если результат находится среди чисел, больших эталона, то, очевидно, во всех отмеченных разрядах правого столбца а=1 (а в соответствующем разряде эталона *x=0).* В этом случае результат определяется путем поиска минимума по младшим разрядам.
Если же результатом является число, меньшее эталона, то в отмеченных разрядах *a=0* (соответственно *x=1)* и результат определяется путем поиска максимума по младшим разрядам.

Система логических функции ячейки может быть получена из систем, описывающих две предыдущие ячейки.

Специализированные однородные процессоры, ориентированные на операции поиска ближайшего большего и меньшего, будем называть соответственно $v_Б$- и $v_M$- процессором, а процессор для поиска ближайшего по абсолютной величине - $v_A$- процессором, $v_Б$- и $v_M$ -процессоры обладают более широкими функциональными возможностями, чем другие рассмотренные выше процессоры.

Рассматривается другой аппаратный метод обеспечивающий поиск ближайшего числа за одну операцию. Суть этого метода состоит в том, что в ассоциативное ЗУ записывается заранее составленная таблиц входами которой являются всевозможные значения двоичных чисел заданной разрядности *m,* а выходами соответствующие этим числам (ближайшие) признаки заданного массива. Поиск ближайшего сводится, таким образом, к поиску по совпадению: при подаче на вход произвольного двоичного числа *X* в ассоциативном ЗУ выделяется единственная строка, содержащая во входной зоне это число, после чего из выходной зоны дани строки считывается соответствующее значение $A_i$ – ближайший признак.

## 2.6 Базовые операции специализированных однородных процессоров

Для удобства дальнейшего изложения материала присвоим базовым операциям поиска, выполняемым в описанных процессорах, условные обозначения. Эти обозначения приведены в табл. 5, где указаны также типы процессоров, реализующих соответствующие операции.

Каждая из рассмотренных выше элементарных ячеек содержит один двоичный запоминающий элемент, в котором хранится соответствующий двоичный разряд массива-аргумента. Можно сказать, что матричное ЗУ в целом хранит входной образ, а комбинационная схема распознает те или иные свойства входного образа. В ряде случаев матричное ЗУ может содержать не один, а два двоичных запоминающих элемента или более в каждой ячейке. Так, могут понадобиться запоминающие элементы для записи выходного образа, для запоминания внутренних состояний ячеек (если для обработки используется итеративная сеть с памятью) и т. д.

Таблица 1

| Базовая операция | Процессор, выполняющий данную операцию |
|---|---|
| Поиск по совпадению | $\varepsilon_Б$, $\varepsilon_М$, $\varepsilon_И$, $\nu_Б$, $\nu_М$ |
| Поиск максимума (наибольшего) | $\alpha_Б$, $\nu_М$ |
| | $\alpha_М$, $\nu_Б$ |
| Поиск минимума (наименьшего) | $\varepsilon_Б$, $\varepsilon_М$, $\varepsilon_И$, $\nu_Б$, $\nu_М$ |
| | $\varepsilon_Б$, $\varepsilon_М$, $\varepsilon_И$, $\nu_Б$, $\nu_М$ |
| Поиск всех больших | $\varepsilon_И$ |
| | $\nu_Б$ |
| Поиск всех меньших | $\nu_М$ |
| | $\nu_А$ |
| Поиск по интервалу | |
| Поиск ближайшего большего | |
| Поиск ближайшего меньшего | |
| Поиск ближайшего по абсолютной величине | |

Важное значение имеет организация цепей записи и считывания в матрице запоминающих элементов. Практические возможности и схемы здесь в значительной степени зависят от элементной базы и технологии. Поэтому рассмотрим только структурные особенности нескольких вариантов движения информации в матричных ЗУ.

Если при выполнении этой операции по некоторым вертикальным управляющим шинам поступают сигналы маскирования записи, то в соответствующих столбцах содержимое запоминающих элементов не изменяется.

Таблица 2

| № п.п варианта | Методы перевода ключевых слов в числовую форму | Базовая операция |
|---|---|---|
| 1 | деления | Поиск по совпадению<br>Поиск максимума (наибольшего)<br>Поиск ближайшего меньшего |
| 2. | извлечения битов | Поиск всех больших<br>Поиск ближайшего по абсолютной величине<br>Поиск по интервалу |
| 3. | квадрата | Поиск ближайшего большего<br>Поиск минимума (наименьшего)<br>Поиск всех меньших |

**3. Задание**

1. Составить блок-схему алгоритма работы ассоциативно - запоминающего устройства (АЗУ), выполняющего базовые операции.
2. Составить программу на языке высокого уровня, моделирующую работу преобразователя ключевых слов в числовую форму.
3. Промоделировать (тестировать) базовые операции указанного варианта на вычислительной машине последовательной структуры.

Данные для выполнения лабораторной работы №1 находятся в таблице 2.

**4. Содержание отчета**

Отчет должен содержать:

- титульный лист;
- задание и структуру АЗУ;
- блок-схему алгоритма работы АЗУ;
- текст программы;
- результаты работы программы.

**Контрольные вопросы.**

1. Дайте определение ассоциативного - запоминающего устройства.
2. Назовите основные блоки АЗУ.
3. Как производится операция записи информации в АЗУ.
4. Как производится выборка информации из АЗУ.
5. С какой целью применяется маскирование в АЗУ. Как осуществляется эта операция.
6. Назовите части памяти с адресацией по содержанию (ПАС). Чему равно время доступа к ПАС.
7. Перечислите основные поисковые операции АЗУ.
8. В чем заключается идея Хеширование.
9. Назовите достоинства и недостатки операции Хеширование. Как происходит преобразование ключевых слов в допустимые адреса памяти.
10. Что такое Хеш – адрес.
11. Какие основные методы перевода ключевых слов в числовую форму вам известны.
12. Как осуществляется поиск максимума с помощью ориентированного процессора.
13. Какие логические функции реализуются в процессоре, ориентированного на операцию поиска всех больших и всех меньших чисел.
14. Как осуществляется операция поиска на интервалах. Какой алгоритм при этом используется.
15. Как осуществляется операция поиска ближайшего числа в ориентированном ε- процессоре. Какие логические функции при этом применяются.

## Библиографический список

1. Ассоциативные запоминающие устройства. Кохонен Т., М., Мир, 1982. – 384с.
2. Ассоциативная память. Кохонен Т., М., Мир, 1980. – 229с.
3. Параллельные процессоры для управляющих систем. Фет Я.И., М., Энергоиздат, 1981. – 160с.
4. Ассоциативные запоминающие устройства. Крайзмер Л.П., Бородаев Д.А., Гутенмахер Л.И., Л., Энергия, 1967. – 184с.