Автоматизация вывода формулы чередования адресов

Как известно, при переборе адрес в двоичном коде, каждый последующий бит изменяется вдвое реже, чем предыдущий. То есть, бит 0 изменяется на каждом адресе, бит 1 – на каждых двух адресах, бит 2 – на каждых четырёх и т.д. Отсюда следует, что если начинать перебор с «круглого» адреса (в идеале – с нуля), то бит 0 изменит своё значение первым, бит 1 – вторым, бит 2 – третьим и т.д. Если смотреть только на вновь изменившиеся биты и игнорировать те, которые уже ранее изменялись, то можно выявить последовательность изменения битов, а значит – отсортировать их в порядке частоты изменения. Если же изменение битов не подчиняется частотным законам, то это может быть выявлено последующим перебором адресов по получившейся формуле и сравнением расчётной последовательности с исходной. При несовпадении последовательностей, можно сказать, что исходная последовательность не является перебором адреса с перемешанными адресными линиями.

Эта закономерность позволяет легко вывести формулу чередования адресов (в случае начала перебора адреса с «круглого» значения) по следующему алгоритму:

1) Взять первое значение в качестве эталонного
2) Очистить список уже менявшихся битов.
3) Взять очередное исследуемое значение. Произвести операцию ИСКЛЮЧАЮЩЕЕ ИЛИ с эталонным значением. Биты, получившиеся в результате операции являющся изменившимися по отношению к эталону.
4) Найти такой бит, который ещё не представлен в списке уже менявшихся битов. Если не найдено ни одного бита – перейти к шагу 2. Если найдено более одного бита – выйти с сообщением об ошибке.
5) Поместить бит, найденный на шаге 4 в результирующую формулу на очередную позицию
6) Поместить бит, найденный на шаге 4 в список уже менявшихся битов
7) Если в исходной последовательности есть ещё значения – перейти на шаг 2.

После выполнения данного алгоритма, следует перенести в результирующую формулу номера тех битов, которые не были обработаны алгоритмом, так как они не перемешаны, а следовательно – должны располагаться в порядке возрастания номера бита.

Для того, чтобы не требовать от пользователя вводить последовательность, начинающуюся строго с «круглого» бита, можно предложить следующий алгоритм:

1) Найти в списке такие участки, где изменяется максимальное количество битов при переходе от элемента K на элемент K+1. Первый из таких элементов обозначим, как N.
2) Если число изменившихся битов равно K и $N=2^K-1$, то список начинается с «круглого» числа, то есть, его можно разбирать с начала. Иначе – начинать разбор с элемента N+1.

Упрощение ручного ввода номеров страниц

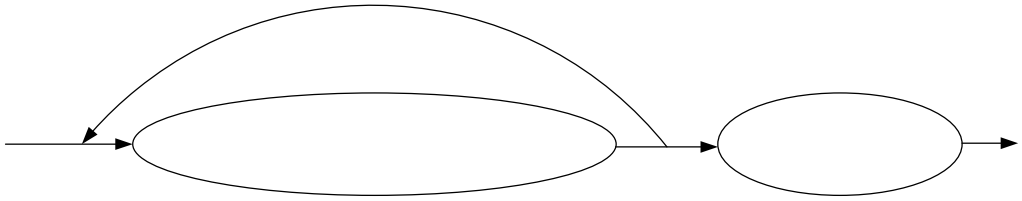
До тех пор, пока не будет сделано устойчивое формирование списка страниц такой длины, какая необходима для автоматического вывода формулы чередования адресов, необходимо обеспечить возможность ввода этого списка оператором. При ручном вводе необходимо избегать монотонных данных, так как пользователь может легко ошибиться, например,

при вводе последовательности из 0x40 значений: 2240, 2242, 2244, 2246, 2248… Кроме того, время, требуемое на ввод будет неоправданно высоким.

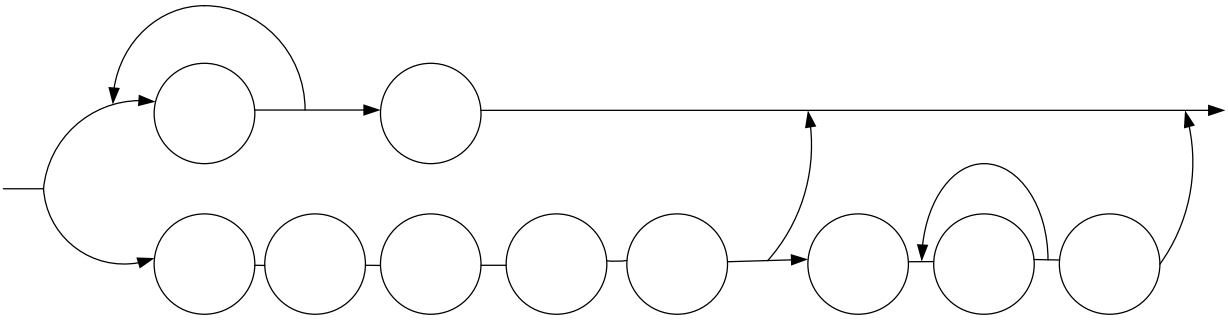
В связи с этим, необходимо разработать язык, позволяющий описывать равномерно нарастающие последовательности чисел. Статистика, накопленная при изучении чередования адресов, позволяет предъявлять следующие требования к языку:

- Следует иметь возможность описать произвольную последовательность чисел, после перебора которых следует увеличить каждое, после чего – снова начать перебор
- Шаг приращения должен быть произвольным, а не обязательно 1.

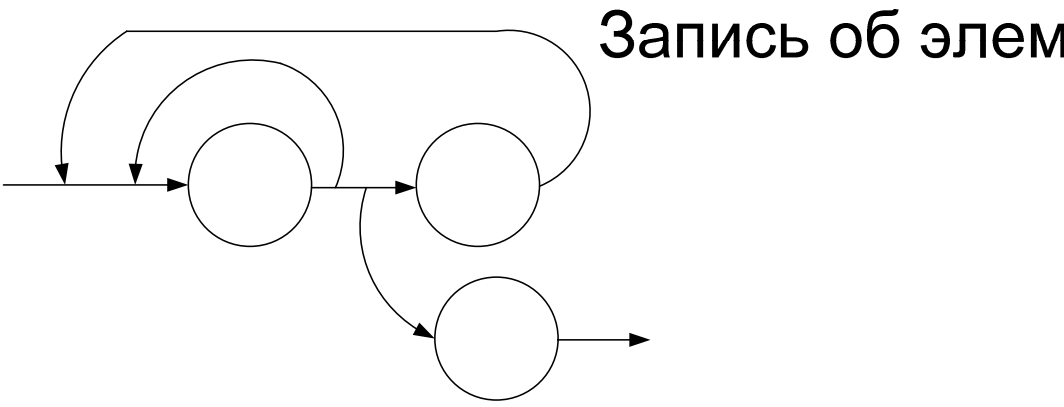
Таким образом, перечень страниц описывается следующей синтаксической диаграммой:



То есть, перечень состоит из произвольного набора записей об элементах. Признак конца записей – символ конца строки (код 0x00). Каждая запись описывается следующей синтаксической диаграммой:



Верхняя ветка определяет обычную запись в форме числа (набора цифр). Числа обязательно разделяются запятыми. Нижняя ветка определяет запись в виде пары списков, заключённых в скобки и разделённых двумя точками. Каждый список описывается следующей синтаксической диаграммой:



То есть, состоит из чисел, разделённых запятыми. Конец списка определяется по закрывающей скобке.

Число элементов в первом и втором списке должно совпадать.

После второго списка, при необходимости, может стоять шаг приращения, заключённый в квадратные скобки. Если шаг не указан, считается, что приращение ведётся на 1 шаг.

Рассмотрим примеры задания списков, на основе которых следует выводить формулу чередования адресов.

Пример 1.

Чередование «через ½ микросхемы при общем числе страниц 0x20000.

Список должен иметь формат:

0,10000

Пример 2.
Чередование идёт через EU, размером 0x80 страниц.

Список должен иметь формат:

0,80

Пример 3. Чередование идёт по правилу:

0, 80, 2, 82, 4, 84

Список должен иметь вид:

(0,80)..(7F,FF)[2]

Пример 4. Чередование идёт по правилу:

0, 80, 10000, 10080, 2, 82, 10002, 10082, 4, 84,10004, 10084

Список должен иметь вид:

(0,80,10000,10080)..(7F,FF,1007F,100FF)[2]

Пример 5. Сначала перебираются все страницы одного и того же стираемого блока, размером 0x80 страниц, затем - перебираются все страницы стираемого блока, отстоящего на половину ёмкости (как и в предыдущих примерах, выберем ёмкость, равную 0x20000 страниц, то есть, половина равна 0x10000). Следовательно, чередование идёт по правилу *0, 1, 2, …, 7F, 10000, 10001, …, 1007F*

Список должен иметь вид:

(0)..(7F),(10000)..(1007F)

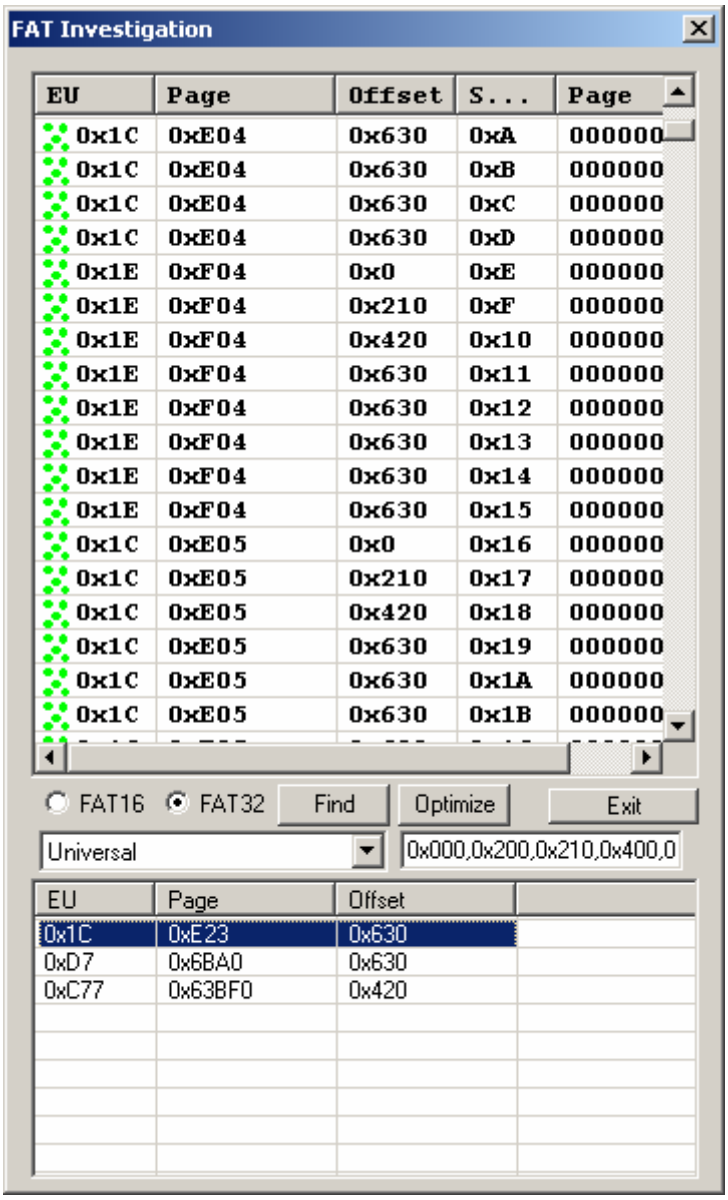
Примечание: Запятая между двумя парами списков не относится к обязательным требованиям, она может быть опущена, а может быть и вставлена. Оба варианта записи являются приемлемыми, так как имеются аргументы как в пользу того, что запятая нужна (единство стиля, все элементы, включая составные, разделены запятой), так и в пользу того, что она не нужна (составные элементы сами по себе являются чётко выраженной структурой и не требуют разделителя).

Пример 6. Пусть аргументы для примера 5 выведены на основе результатов работы какого-либо анализатора, поэтому номера страниц будут начинаться не с нуля, а со значения, допустим, 0x2480. Тогда список должен иметь вид:

(2480)..(24FF),(12480)..(124FF)

Результирующая формула при этом будет той же, что и в результате расчёта для примера 5.

Пример 7. Анализатор FAT показывает последовательность E00,F00,E01,F01,E02,F02…



Но так как размер FAT невелик, более детально выявить порядок страниц невозможно (FAT заканчивается в районе страниц E23-F23)