Архитектура.

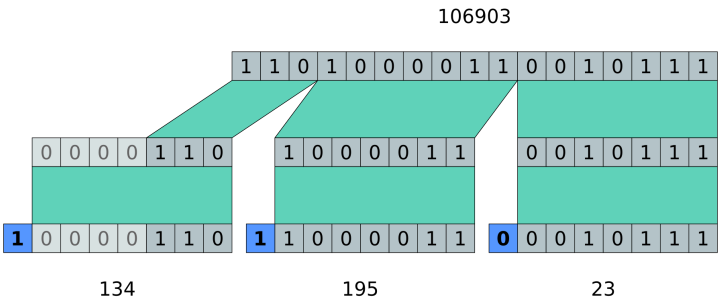
- Архитектура планируется стековая, безадресная.
- Как такового отдельного стека (операндов) не предполагается, это область памяти, на которую указывает пара регистров.
- В дальнейшем для работы со стеком операндов предполагается небольшой кэш ~ на пару десятков слов.
- Стек локальных переменных также расположен в памяти
- Предполагается что оба этих стека расположены в одной области памяти и растут навстречу друг другу, при встрече возникает аппаратная ошибка.
- Инструкции упаковываются алгоритмом Vluint7, каждая инструкция представлена опкодом и аргументами. И опкод и аргументы закодированы Vluint7
- Два младших разряда опкода - число аргументов, облегчим жизнь декодеру.
- Потоки управления и исполнения разделены. Т.е. есть два независимых декодера - потока управления и потока исполнения (strands). И два счетчика команд.
- Инструкция из потока управления может запустить новый strand, после чего дожидается конца его работы (когда потоку исполнения не встретится стоп-инструкция). После возврата, поток управления продолжает работу.

Подготовка.

Распаковщик Vluint7

Как мы условились, поток инструкций - это записанные подряд числа, упакованные Vluint7. В этом алгоритме число (не важно, 64-х, 32-х или 16-разрядное) записывается как последовательность байт, в каждом из которых 7 значащих разрядов и один управляющий, который означает- закончена запись числа или нет. Так, 32-х разрядное значение может потребовать от 1 до 5 байт. Но поскольку идентификаторы инструкций или сдвиги до данных (из которых предположительно состоит код) обычно небольшие числа, такая запись довольно компактна.

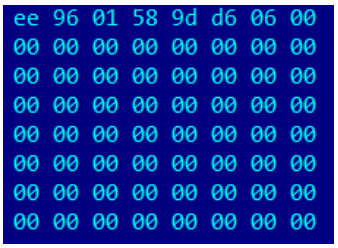
Есть два варианта записи - начиная с младших или со старших разрядов. Второй вариант показан на Фиг.5.2.1, но мы будем использовать друго, он представляется чуть более простым в реализации.



Фиг.5.2.1 Кодирование методом Vluint7.(отсюда),

Для тестирования модуля распаковки Vluint7 выберем следующие данные:
три числа 0x4b6e, 0x58, 0x1ab1d, в бинарном виде это выглядит так:
00000000 00000000 01001011 01101110
00000000 00000000 00000000 01011000
00000000 00000001 10101011 00011101
цветами выделены блоки по 7 разрядов.

Всего получилось уложиться в 7 байтов, они показаны на Фиг.5.2.2



Фиг.5.2.2 Содержимое памяти, для тестирования выбран блок в 64 байта.

Модуль vluint7 имеет интерфейс:

```
 3    module vluint7 (
 4       input wire clk,
 5       input wire reset,
 6       input wire beg,
 7       input wire [`MEM_ADDR_WIDTH-1:0] addr,
 8
 9       output logic [`MEM_ADDR_WIDTH-1:0] addr_out,
10       output logic rd,
11       output logic [`WORD_WIDTH-1:0] data
12      );
```

Фиг.5.2.3 интерфейс модуля

Здесь:

- clk: синхроимпульс
- reset: сброс состояния
- beg: сигнал к распаковке
- addr: адрес начала распаковки
- addr_out: адрес на котором закончилась распаковка
- rd: сигнал об окончании распаковки
- data: распакованные данные

Распаковка начинается с приходом сигнала beg

```
39      always @ (posedge beg)begin
40          loc_rd <= 0;       // ожидание чтения памяти
41          loc_beg <= 1;      // начинаем читать память
42          data <= 0;         //
43          loc_shift <= 0;    // распаковка с младших разрядов
44          loc_addr <= addr; //
45          rd <= 0;           // результат не готов
46          working <= 1;      // распаковываем
47      end
```

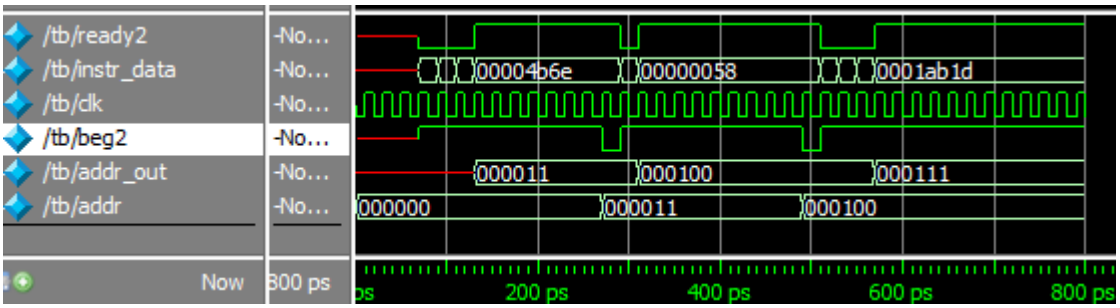
Фиг.5.2.4 начало работы

Собственно распаковка:

```
51      always @ (posedge loc_wrd) begin
52        if (working) begin
53          data <= data | (tmp_data[6:0] << loc_shift);
54          loc_addr ++;
55          if (tmp_data[7]) begin
56            loc_shift += 7;
57            loc_beg <= 1;
58          end else begin
59            rd <= 1;
60            working <= 0;
61            addr_out = loc_addr;
62            loc_beg <= 0;
63          end
64        end
65      end
```

Фиг.5.2.4 распаковка

Всё довольно просто, по окончании чтения памяти, сохраняем текущие 7 разрядов в их позицию, наращиваем адрес чтения, отдаём команду на чтение и наращиваем позицию сдвига данных.



Фиг.5.2.3 результат работы симулятора

Эмулятор (в данном случае ModelSim от Altera, тут без него не обойтись) демонстрирует нам что распакованы три числа, прочитано 7 байтов памяти.