29 Регулярное сложение

Задача. Боб узнал, что если в криптографических программах имеются условные переходы и условия этих переходов определяются обрабатываемыми данными (но не их размерностями), то эти программы подвержены атакам, основанным на замерах времени или питания. Боб решил переписать программы шифровальной машины Amgine так, чтобы все вычисления были регулярными, т. е. не содержали бы небезопасных условных переходов. Помогите Бобу переписать следующую функцию, реализующую сложение больших чисел:

```
word zzAdd(word c[], const word a[], const word b[], size_t n) {
  register word carry = 0;
  register word w;
  size_t i;
  for (i = 0; i < n; ++i) {
    w = a[i] + carry;
    if (w < carry)
      c[i] = b[i];
    else
      w += b[i], carry = w < b[i], c[i] = w;
  }
  w = 0;
  return carry;
}
```

Функция написана на языке Си. Большие числа задаются массивами из `n` машинных слов. Буферы `a` и `c`, `b` и `c` либо не пересекаются, либо совпадают.

Решение. В прекрасной книге [Уоррен Генри Мл. Алгоритмические трюки для программистов, М.: Изательский дом «Вильямс», 2003] собраны примеры необычных манипуляций над машинными словами. В том числе представлены способы реализации предикатов сравнения `x == y` и `x < y` с помощью одних только арифметических операций и сдвигов.

Воспользуемся трюками Уоррена:

```
#define safeEq(x, y)\
(~((x) - (y) | (y) - (x)) >> (8 * sizeof(word) - 1))

#define safeLess(x, y)\
((~(x) & (y) | ((~(x) | (y)) & (x) - (y))) >> (8 * sizeof(word) - 1))

word zzAddSafe(word c[], const word a[], const word b[], size_t n)
{
  register word carry = 0;
  register word w;
  size_t i;
  for (i = 0; i < n; ++i)
  {
    w = a[i] + b[i] + carry;
    carry &= safeEq(w, a[i]);
    carry |= safeLess(w, a[i]);
    c[i] = w;
  }
  w = 0;
  return carry;
}
```

□