Proof of Part #1

Invarant: $c + \sum_{k<i} as[k] = \sum_{k<i} (as'[k] + bs[k])$

Basis, ✓
At begining of loop i is 0 so bothe sides = 0

~~When i size of bs~~
Goal: prove true for i+1 case

$as'[k] = as[k]$ for every $k > i$

$c + \sum_{k<i+1} as[k] = \sum_{k<i+1} (as'[k] + bs[k])$

$\sum_{k<i} (as'[k] + bs[k]) c^k + sum\, c^i$

$= \sum_{k<i} (as'[k] + bs[k]) c^k + (as'[i] + bs[i] + c) c^i$

This is true because sum is really just calculating original A + b + carry then storing in c' 'place'

$= \sum_{k<i} as[k] \cdot c^k + (as'[i] + bs[i]) c^i$

because $(sum >> 32)c + (uint32\_t)sum = Sum$

we can say: $\sum_{k<i} (as'[k] + bs[k]) c^k + sum \cdot c^i$

$= \sum_{k<i} (as'[k] + bs[k]) c^k + (sum >> 32 \cdot c + (uint32\_t) sum) c^i$

Once $as[i] = (uint32\_t) sum$,

becomes: $\sum_{k<i+1} as[k] \cdot c^k + c = \sum_{k<i+1} (as'[k] + bs[k]) c^k + sum \cdot c^i$

Cont. The second for loop then allows us to correct the equation

The loop propagates all the carries, so everything is now accounted for and we are left with

$$\sum_{k<i+1} as[k] + \overset{(carry)}{\overset{\downarrow}{c}} = \sum_{k<i+1} (as'[k] + bs[k])$$

∴ Our hypothesis is true

* Sum is removed because for every case the carry is calculated and passed on

only applies if num is similar to ↘ 0xffffffff;

Proof Part 2

At the start of the partialprod32 for loop we start with i=0 so the following is true

~~$\sum_{k<i} (bs[k] \times d)$~~

$$\sum_{k<i} (bs[k] \times d) = \sum_{k<i} as[k] + c$$

We then use a temp variable to store the lower half of $bs[i] \times d$ which gets placed in as[i] and upper half that gets stored in as[i+1]

On other iterations of the loop, we must complete addition and make sure to propogate the carry.

the way I have set this up is by shifting temp$ (the storage for our sums) and checking for a carry bit this is then saved and recursivly added in the next loop

This maintains the carry at any given iteration of the loop, Once this is accounted for, we can assume the proof holds for the i+k case.

Part 3

bigmul simply calls all the functionality of partialprod32 but also accounts for a carry the flows into an index the previous iteration could not hadle. If there is a carry, it is passed into the next iteration and added in the corresponding index.