aky Oct 8, 57

Suggested Method to prevent unnecessary carries in adding single precision no. to double precision accumulator.

In adding single precision numbers to double precision accumulator, with opposite signs one has the time penalty of propagating a carry the full length of the low order part of the adder in spite of the fact the final answer is unchanged.

An example using 5 bits being subtracted from a 10 bit accumulator with an exponent offset of 2,

original nos.

acc. 10101.01010 mem −11011 (with exp. 2 less)

True answer: 10101.11010
−00110.11000 ← these zeros not significant
00010
01111

machine method with complement checking.

True: 10101.11010
adder + 11001.00111 ← +1 introduced carry which will not propagate not sig. out
answer (1) 01111.00010

Complement: 01010.00101
adder + 00110.11000 ← same as orig. no.
comp. answer 10000.11101

In this case there are no unnecessary carries in the complement adder but there are in the true adder.

Suggested method:

Subtract 1 in low order part of no. from memory & fill in non-significant zeros by 1's before going into shifter.

mem -11010|11111 for true adder only.

True adder: ← no carry introduced here

```
 10101|11010
+11001|01(000)
-------------
(1)01111|00010|
```

↳ no carries necessary

Complement adder: ← same as before

```
 01010|00101
 00110|11(000)
-------------
 10000|11101
```

↳ no carries necessary.

Main drawback: This implies an extra subtract cycle on the word from memory before starting the regular add.

Consider case where accumulator is mostly zeros.

acc. 10000|0000 mem -10000 (exp 2 bits)

changed to -01111|11111

↳ subtract of one causes carry here