Full Adder

locked

by *IEEEXtreme*

Statistics

Difficulty: Hard
Publish Date: Oct 12 2016

Three of the biggest challenges in this problem are:

- working with the different bases
- managing the very large operands, and
- correctly aligning the result.

The following solution while reads operands as a person (from most significant symbol to the least significant symbol), numbers are rearranged and the process is done as on paper, from LSB to MSB while keeping track of the carry. The following solution is written in Python 3, however we can safely assume that an equivalent solution could be programmed to any programming language. Detailed comments are written starting with the character #.

```
# ---------------------------------------------------------
# ---------------------------------------------------------
# Get basic data
base_len, base = input().split(' ')

# First number, strip any spaces
line1 = input()
num1 = line1.strip()

# Second number, strip any spaces, strip also the plus sign
line2 = input()
num2 = line2.strip().lstrip('+').strip()

# This is needed to reprint everything the same
max_line_len = max(len(line1), len(line2))

# We need to switch from Most Significant Symbol (MSS)
# .. to the Least Significant Symbol (LSS)
num1 = num1[::-1]
num2 = num2[::-1]

# To generilise the solution
# .. we put "zero" numbers ahead, in order to make the
# .. two number of the same length
num_digits = max(len(num1), len(num2)) + 1
num1 += base[0] * (num_digits - len(num1))
num2 += base[0] * (num_digits - len(num2))

# Each cycle adds only one symbol
# .. starting from the first one
# .. which now is the LSS
pos = 0
d1 = num1[pos]
d2 = num2[pos]

# We need to know when we have carry
carry = 0

# This will hold the final result
# .. caution: it is not a number but a string
result = ''
while pos < num_digits - 1:
    # We add everything in decimal
    # .. according to their position in the given base
    res = base.find(d1) + base.find(d2) + carry

    # .. and we rebase to find the correct symbol
    # .. from decimal to the custom one
    result += base[ res % len(base) ]

    # .. but we don't care to have the carry in a diffirenet base
    # .. as it is just 0/1 decimal and added as is
    carry = int(res / len(base))

    # .. and then init the next cycle
    pos += 1
    d1 = num1[pos]
    d2 = num2[pos]

# Print as expected
print('%d %s' % (int(base_len), base))
```

```
print(line1)
print(line2)
print('-' * max_line_len)

# Switch back to MSS to print it correctly
result = result[::-1]
print(' ' * (max_line_len - len(result)) + result)
```