# Fixed-Point Number (Addition/Multiplication)

#### Integers

□ Generally we use 8-bits, 16-bits, 32-bits or 64-bits to store integers.

□ 20 = 16+4 = 
$$(0001\ 0100)_2$$
 8-bits  
□ 20 = 16+4 =  $(0000\ 0000\ 0001\ 0100)_2$  16-bits  
 $\leftarrow_{pad\ zeros\rightarrow}$ 

■ We use 2's complement format for the notation of negative signed numbers:

$$20 = (0...01\ 0100)_2$$
  
 $-20 = (1110\ 1100)_2$  8-bits  
 $-20 = (1111\ 1111\ 1110\ 1100)_2$  16-bits  
Sign bit

#### Integers

- □ How to store integers in registers?
- □ Consider that we have 8-bit registers.
- $\square$  20 = (10100)<sub>2</sub>
- □ As 8-bit integer: (r1)
  - $r1 = 20 = (0001\ 0100)_2$
- □ As 16-bit integer: (r1 r2)
  - r1 = 0 = (0000 0000)<sub>2</sub>
  - $r2 = 20 = (0001 \ 0100)_2$
  - $(r1 r2) = 20 = (0000 0000 0001 0100)_2$
- □ As 32-bit integer: (r1 r2 r3 r4)
  - -1 r1 = r2 = r3 = 0 =  $(0000\ 0000)_2$
  - $r4 = 20 = (0001\ 0100)_2$
  - $(r1 \ r2 \ r3 \ r4) = 20 = (0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0001 \ 0100)_2$

#### Integers

- □ Represent 123456789 in 32-bit integer:
  - 123456789 = (111 0101 1011 1100 1101 0001 0101)<sub>2</sub>
  - Convert to 32-bits:

- **•** 0000 0111 0101 1011 1100 1101 0001 0101
- $r1 = (0000 \ 0111)_2^{12} = 0x07 = 7^{13}$
- $r2 = (0101 \ 1011)_2 = 0x5b = 91$
- $r3 = (1100 \ 1101)_2 = 0xcd = 205$
- $r4 = (0001 \ 0101)_2 = 0x15 = 21$
- (r1 r2 r3 r4) = 0x075bcd15
   = (0000 0111 0101 1011 1100 1101 0001 0101)<sub>2</sub>
   = 123456789

#### Integers (2's complement)

- ☐ Given following values of registers, find the value of (r4 r3 r2 r1)?
- □ r1 = 72, r2 = 100, r3 = 250, r4 = 255

```
r1 = 72 = (0100\ 1000)_2

r2 = 100 = (0110\ 0100)_2

r3 = 250 = (1111\ 1010)_2

r4 = 255 = (1111\ 1111)_2

(r4 r3 r2 r1) = (1111\ 1111)_2 1111 1010 0110 0100 0100 1000)<sub>2</sub>

The number is negative! Take 2's complement:

(0000\ 0000\ 0000\ 0101\ 1001\ 1011\ 1011\ 1000)_2 = 367544
```

g-bits

□ Assume that you have an operator that adds only two digits:

Each digit is a number in a base b.

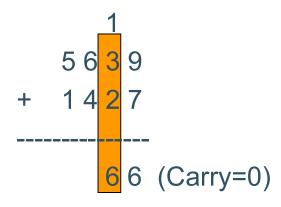
Note that the sum of two single-digit yields one digit and extra one bit at most!

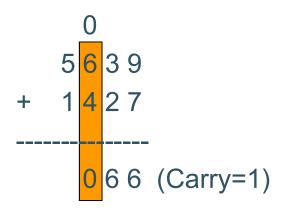
□ Assume that we have an operator that adds only two digit. How can we add two numbers with multiple digits?

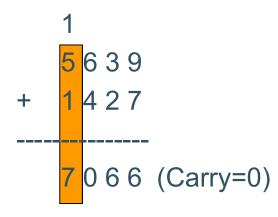
Solution: Add digits individually

Also add carry!



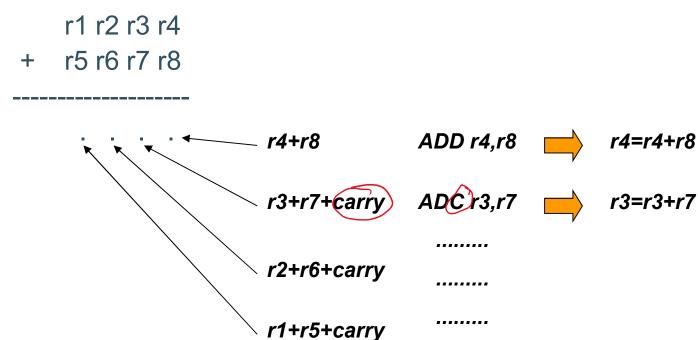






Now consider that we are working in base 256.

Put each digit in a register so that we'll have 4 register for each 32-bit number.



- What about signed numbers?
- □ Use 2's complement for negative numbers and just add! Ignore the last produced carry.
- □ How does it work? Explained later...
- What about subtraction?
- □ Subtraction can easily be implemented by taking 2's complement of the second operand first and then applying addition:
  - A-B = A+(-B)

Assume that you have an operator that multiplies only two digits:

Each digit is a number in a base b.

b=10 => numbers: 0-9

b=2 => numbers: 0-1

b=28=256 => numbers: 0-255

Note that the product of two single-digit yields two digits at most!

□ You have more than one digit to multiply:

By using the operator, we can calculate: 7x8 = 56

7x5 = 35

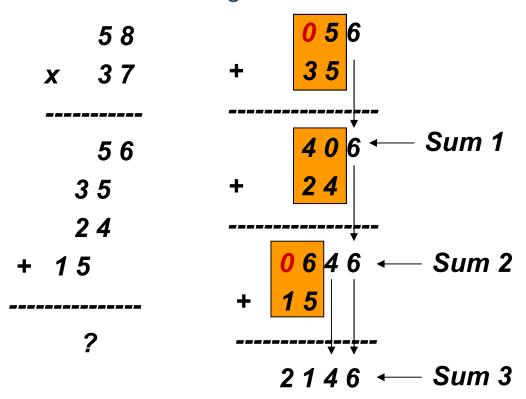
3x8 = 24

3x5 = 15

□ How can we use these values to calculate the result?

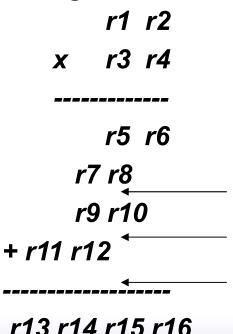
```
58
x 37
-----56
35
24
+ 15
```

□ We can use integer addition to find the result.



- This operation is equivalent to 16-bit multiplication using 8-bit multiplication and 8-bit addition.
- Note that the number of digits in the result is equal to the sum of the number of input digits.

- □ Now assume that the digits are in base  $2^8 = 256$ . (8-bit are necessary for each digit)
- □ Then, 16-bit multiplication is done by using 8-bit multiplication and 8-bit addition. Each 8-bit register can hold only one digit!



In fact, we do not need 16 registers to accomplish 16-bit multiplication.

If we compute the partial sums, we can re-use the registers which hold the values that are unnecessary.

#### **Fixed-Point Numbers**

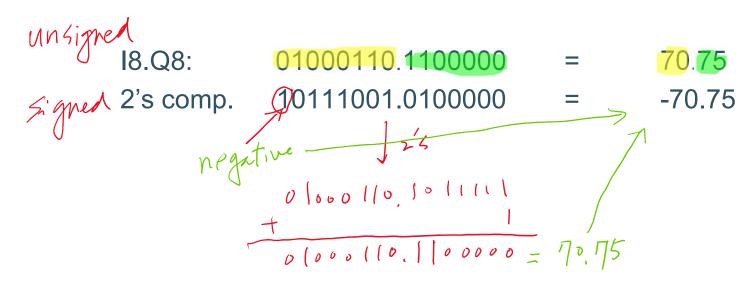
- □ Fixed-point numbers are generally stored in "In.Qm" format (sometimes referred as Qn.m format)
- n = number of bits in integer part.
- m = number of bits in fractional part.
- □ Example: I8.Q16

27	2 <sup>6</sup>	2 <sup>5</sup>	24	<b>2</b> <sup>3</sup>	<b>2</b> <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>		2 <sup>-1</sup>	<b>2</b> -2	2 <sup>-3</sup>	2-4	2 <sup>-5</sup>	2-6	2-7	2-8	2 <sup>-9</sup>				2 <sup>-13</sup>	2 <sup>-14</sup>	2 <sup>-15</sup>	2 <sup>-16</sup>
0	0	1	0	1	1	1	0	•	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

$$= 32 + 8 + 4 + 2 + 1/2 + 1/4 + 1/16$$

#### Signed Fixed-Point Numbers

- Positive fixed-point numbers are the same as unsigned fixedpoint numbers.
- Negative fixed-point numbers are obtained by simply calculating
   2's complement as they are integers.



#### **Fixed-Point Addition**

- □ Fixed-point addition is the same as integer addition!
- □ Align two fixed point number and apply integer addition:

#### **Unsigned Fixed-Point Multiplication**

- Unsigned fixed-point multiplication is similar to integer multiplication.
- □ Consider the following multiplications:

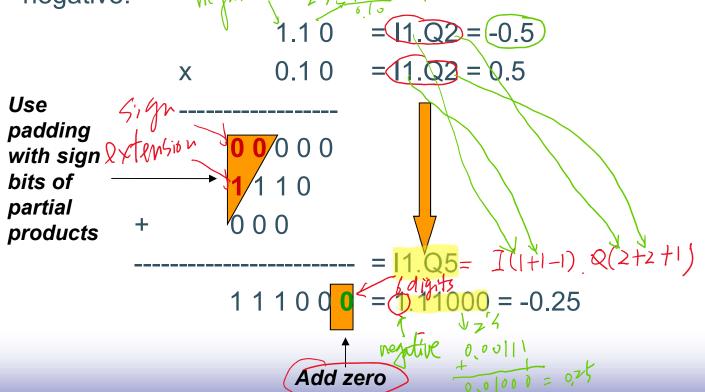
Just multiply like integer multiplication. Align the numbers according to the point (.)

# Signed Fixed-Point Multiplication

□ Use 2's complement format for fixed-point numbers.

$$(Ia.Qb) * (Ic.Qd) = I(a+c-1) . Q(b+d-1)$$

Take 2's complement of the last partial product if multiplier is negative!



#### Signed Fixed-Point Multiplication

