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

  ←pad zeros→

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

$$20 = (0...01\ 0100)_2$$
$$-20 = (\boxed{1}110\ 1100)_2 \quad \text{8-bits}$$
$$-20 = (\boxed{1}111\ 1111\ 1110\ 1100)_2 \quad \text{16-bits}$$

←pad ones→

Sign bit

Integers

- How to store integers in registers?
- Consider that we have 8-bit registers.
- $20 = (10100)_2$
- As 8-bit integer: (r1)
  - $r1 = 20 = (0001\ 0100)_2$
- As 16-bit integer: (r1 r2)
  - $r1 = 0 = (0000\ 0000)_2$
  - $r2 = 20 = (0001\ 0100)_2$
  - $(r1\ r2) = 20 = (\underbrace{0000\ 0000}_{r_1}\ \underbrace{0001\ 0100}_{r_2})_2$
- As 32-bit integer: (r1 r2 r3 r4)
  - $r1 = r2 = r3 = 0 = (0000\ 0000)_2$
  - $r4 = 20 = (0001\ 0100)_2$
  - $(r1\ r2\ r3\ r4) = 20 = (\underbrace{0000\ 0000}_{r_1}\ \underbrace{0000\ 0000}_{r_2}\ \underbrace{0000\ 0000}_{r_3}\ \underbrace{0001\ 0100}_{r_4})_2$

Integers

- Represent 123456789 in 32-bit integer:
  - $123456789 = (\underbrace{111\ 0101\ 1011\ 1100\ 1101\ 0001\ 0101}_{27\text{-bits}})_2$
  - Convert to 32-bits:
  - $\underbrace{0000\ 0111}_{r1}\ \underbrace{0101\ 1011}_{r2}\ \underbrace{1100\ 1101}_{r3}\ \underbrace{0001\ 0101}_{r4}$
  - $r1 = (0000\ 0111)_2 = 0x07 = 7$
  - $r2 = (0101\ 1011)_2 = 0x5b = 91$
  - $r3 = (1100\ 1101)_2 = 0xcd = 205$
  - $r4 = (0001\ 0101)_2 = 0x15 = 21$

  - (r1 r2 r3 r4) = 0x075bcd15

    $= (0000\ 0111\ 0101\ 1011\ 1100\ 1101\ 0001\ 0101)_2$

    $= 123456789$

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

8-bits

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

r2 = 100 = $(0110\ 0100)_2$

r3 = 250 = $(1111\ 1010)_2$

r4 = 255 = $(1111\ 1111)_2$

negative ⇒ what's the negative number?

(r4 r3 r2 r1) = $(1111\ 1111\ 1111\ 1010\ 0110\ 0100\ 0100\ 1000)_2$

The number is negative! Take 2's complement:

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

= -367544

Integer Additon

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

```
            A
+           B
-----------------
carry → C   S ← sum
```

Sum is a digit.

But carry is just a bit, can either be zero or one.

Each digit is a number in a base b.

b=10 => numbers: 0-9

b=2 => numbers: 0-1

$b=2^8=256$ => numbers: 0-255

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

Integer Additon

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

```
     5 6 3 9
+    1 4 2 7
------------
        ?
```

Solution: Add digits individually

Also add carry!

Integer Additon

```
      5 6 3 9
+     1 4 2 7
--------------
            6  (Carry=1)
```

Integer Additon

```
          1
      5 6 3 9
+     1 4 2 7
--------------
          6 6  (Carry=0)
```

Integer Additon

```
        0
      5 6 3 9
+     1 4 2 7
--------------
        0 6 6  (Carry=1)
```

Integer Additon

```
      1
      5 6 3 9
+     1 4 2 7
---------------
      7 0 6 6  (Carry=0)
```

Integer Additon

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.

```
      r1 r2 r3 r4
+     r5 r6 r7 r8
--------------------
       .  .  .  .
```

r4+r8	***ADD r4,r8***	⇨	***r4=r4+r8***
r3+r7+carry	***ADC r3,r7***	⇨	***r3=r3+r7***
	....		
r2+r6+carry	***.........***		
r1+r5+carry	***.........***		

Integer Additon

- 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)

Integer Multiplication

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

```
          A
x         B
---------------
        C D
```

Each digit is a number in a base b.

b=10 => numbers: 0-9

b=2 => numbers: 0-1

$b=2^8=256$ => numbers: 0-255

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

Integer Multiplication

- You have more than one digit to multiply:

```
      58
x     37
----------
       ?
```

By using the operator, we can calculate:

7x8 = 56

7x5 = 35

3x8 = 24

3x5 = 15

Integer Multiplication

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

```
        5 8
   x    3 7
   ----------
        5 6
      3 5
      2 4
+   1 5
---------------
```

Integer Multiplication

- We can use integer addition to find the result.

```
     5 8
x    3 7
---------
     5 6
   3 5
   2 4
+ 1 5
-----------
    ?
```

```
     0 5 6
+    3 5
----------------
     4 0 6   ← Sum 1
+    2 4
----------------
   0 6 4 6   ← Sum 2
+  1 5
----------------
   2 1 4 6   ← Sum 3
```

- *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.*

Integer Multiplication

- 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!

```
          r1  r2
  x       r3  r4
     ------------
          r5  r6
      r7 r8
      r9 r10
+ r11 r12
------------------
 r13 r14 r15 r16
```

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

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
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

= 46.8125

Signed Fixed-Point Numbers

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

unsigned	I8.Q8:	01000110.1100000	=	70.75
signed	2's comp.	10111001.0100000	=	-70.75

negative

2's

01000110.10111111

+ 1

01000110.11000000 = 70.75

Fixed-Point Addition

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

```
     r1 r2 . r3 r4
+    r5 r6 . r7 r8
  -------------------
     ____ . ____
```

Unsigned Fixed-Point Multiplication

- Unsigned fixed-point multiplication is similar to integer multiplication.

- Consider the following multiplications:

58		5.8	I1.Q1	Ia.Qb
x 37		x 3.7	I1.Q1	Ic.Qd
-----------		-----------		
2146		21.46	I2.Q2	
I(a+c).Q(b+d)				

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!

negative

2's 0.01 + 1 = 0.10 = 0.5

```
              1.1 0      = I1.Q2 = -0.5
    x         0.1 0      = I1.Q2 = 0.5
         Sign ------------------
         Extension
          0 0 0 0 0
          1 1 1 0
    +     0 0 0
    ----------------------   = I1.Q5 = I(1+1-1) . Q(2+2+1)
                                6 digits
          1 1 1 0 0 0      = 1.11000 = -0.25
```

Use padding with sign bits of partial products

Add zero

negative

↓ 2's

0.00111 + 1 = 0.01000 = 0.25

Signed Fixed-Point Multiplication

- Example:

negative

1 1.0 1 = I2.Q2 = -0.75

x 1.1 0 1 = I1.Q3 = -0.375

$I(2+1-1).Q(2+3+1) = \underbrace{I(2).Q(6)}_{\text{8-digits}}$

1 1 1 1 1 0 1

0 0 0 0 0 0

1 1 1 0 1

$2's(1101) = 0010 + 1 = 0011$

\+ 0 0 1 1

2's complement of the last partial product

$\underbrace{0\ 0\ 0\ 1\ 0\ 0\ 1\ \boxed{0}}_{\text{8-digits}}$ = I2.Q6 = 00.010010 = 0.28125

discard carry-out

Add zero