Fixed Pt / 1

Fixed Point Arithmetic

FIXED POINT

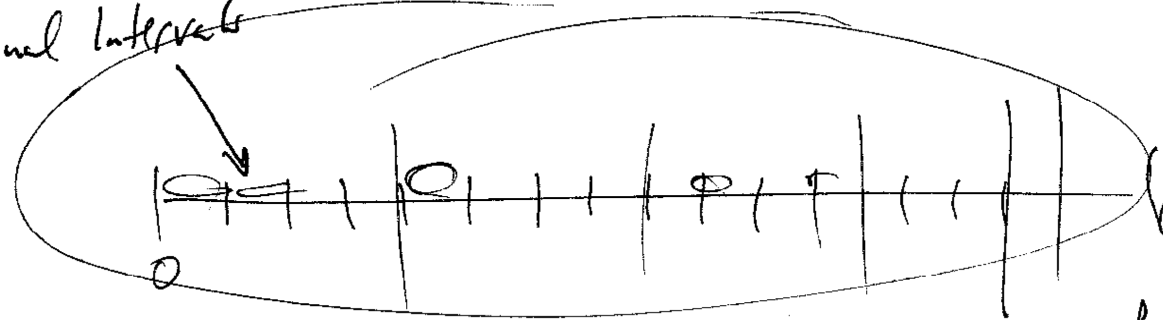
* INTRO
  - WHAT IS FIXED POINT
  - RELATION TO FLOATING POINT
  - WHY SEVERAL CHOICES

* ARCHITECTURAL CHOICES
  - 2'S COMPLEMENT
  - 1'S COMPLEMENT
  - SIGNED MAGNITUDE
  - LONG INTEGERS
  - BCD
  - RESIDUE NUMBERS

* MICROARCHITECTURE MECHANISMS
  - ADDITION (THE CARRY PROBLEM)
    - LAC
    - KOGGE-STONE
    - THE POWER BALL (2X FREQUENCY)
  - MULTIPLICATION (THE ITERATION PROBLEM)
    - BOOTH'S ALGORITHM

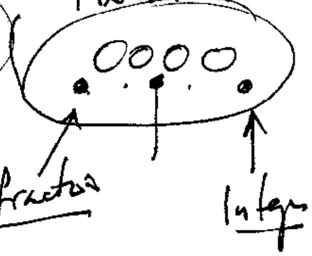
Fixed Point

Equal Intervals



11 . 11

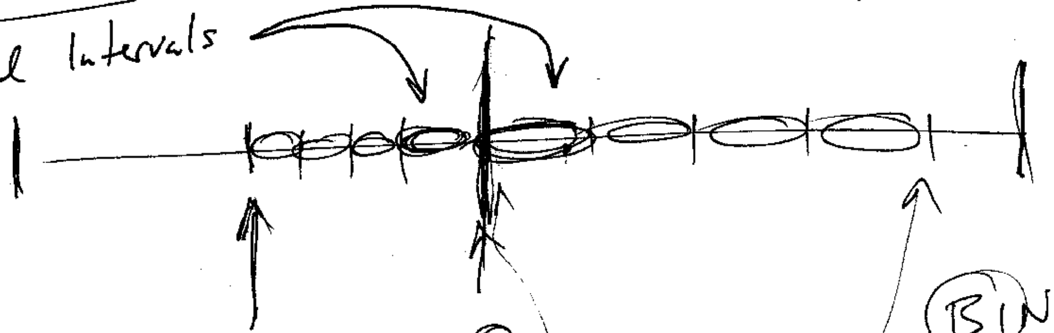
Fixed Point



1111

Floating Point

NOT-equal Intervals



$1.00\text{—}0 \times 2^k$
$10.0\text{—}0 \times 2^k$
100.00×2^k

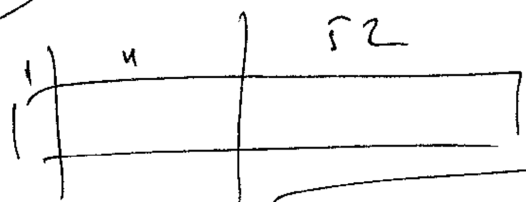
$1.00\ldots \times 2^k$

$1.00. \times 2^{k+1}$

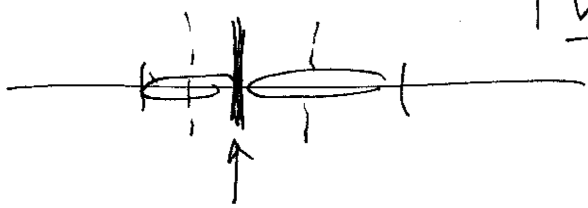
$1.00 \times 2^{k+2}$

$1.00 \times 2^{k+3}$

BINAID



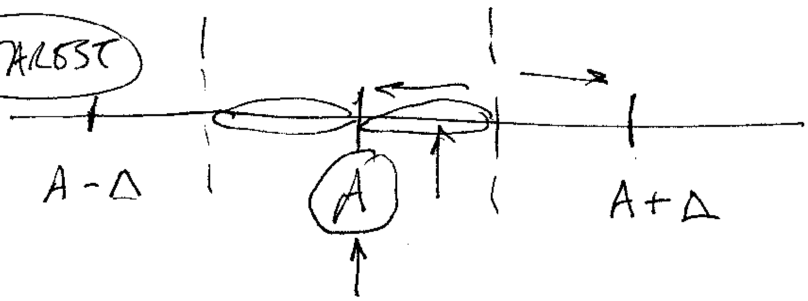
WOBBLE



$-\infty$ 0 $+\infty$

ROUNDING

NEAREST



$1.\boxed{0010} \times 2^k$

$1.0010\,\cancel{0011} \times 2^k$

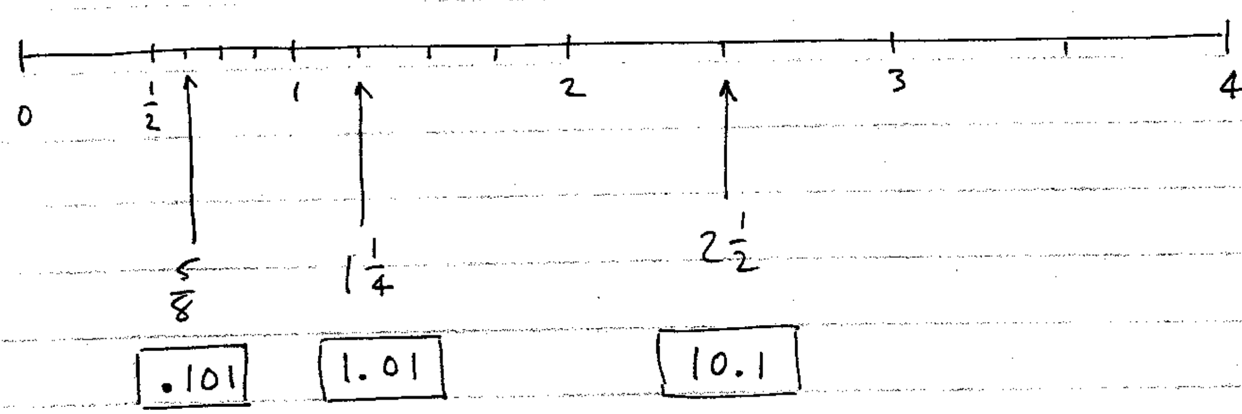
1.0011×2^k

FIXED POINT VS. FLOATING POINT

FIXED POINT : BINARY POINT ALWAYS IN THE SAME PLACE

00000.	.00000	00.000
⋮	⋮	⋮
11111.	.11111	11.111
$0 \le X \le 31$	$0 \le X \le \frac{31}{32}$	$0 \le X \le 3\frac{7}{8}$
$\boxed{\text{INTERVAL} = 1}$	$\boxed{\text{INTERVAL} = \frac{1}{32}}$	$\boxed{\text{INTERVAL} = \frac{1}{8}}$

FLOATING POINT : BINARY POINT **MOVES** FROM BINADE TO BINADE



IN ABOVE EXAMPLE, 2 BITS OF FRACTION.
THEREFORE 3 SIGNIFICANT DIGITS.

NOTE : ~~FLOATING~~ BINARY POINT MOVES

NOTE : INTERVAL CHANGES

Fixed Pt. /5

Computer Arithmetic
(Integers)

* Why several choices for representation

* The Choices

- 2's complement
- 1's complement
- Signed magnitude
- Long integers — Karatsuba's Trick
- Decimal (BCD)
- Residue Arithmetic

* Why several choices?

Application space should drive architecture

- Compute intensive, low I/O
- Arbitrarily large precision
- Generally within a fixed set, with option to go to multiples of that size

* The concept of “Long Integer” vs “Short Integer”

2's Complement | Fixed Pt/7

1's complement

0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101 ← 5
6	6	0110
7	7	0111
−7	−8	1000
−6	−7	1001
−5	−6	1010 ← −5
−4	−5	1011
−3	−4	1100
−2	−3	1101 ← −3
−1	−2	1110
−0	−1	1111

1's Complement   2's Complement

0011

1100

1

1101 ← −3

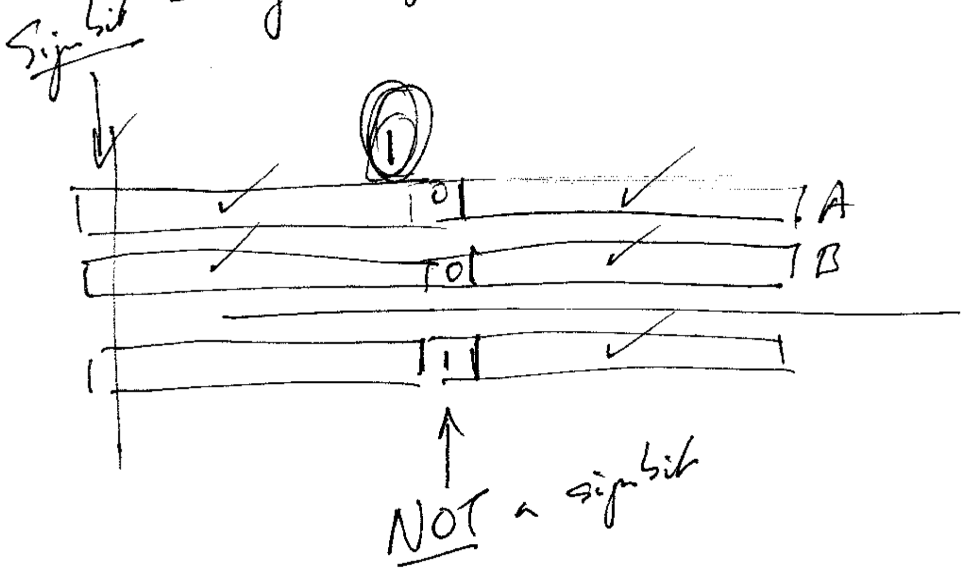
−5 + 3

111111

1

−3 + 5

Long Integers:



Ridiculous!

ADDR

ADC

Three Most Common Schemes

- 2's complement
- 1's complement
- Signed magnitude

Example: (A 4-bit data path)

Representation	What is being Represented: 2's Comp	1's Comp	Sign-Mag
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	-8	-7	0
1001	-7	-6	2
1010	-6	-5	2
1011	-5	-4	3
1100	-4	-3	4
1101	-3	-2	5
1110	-2	-1	6
1111	-1	-0	7

* Why 2's Complement?(Easy for Computer)

* Why 1's Complement (Self-Delusion)

* Why Signed-Magnitude? (Easy for Humans)

BCD Arithmetic

1	1	
6	5	7
2	9	4
8	E	B
9	5	1

6	5	7
0110	0101	0111
0010	1001	0100
0000	1110	1011

0110	0101	0111
0110	0110	0110
1100	1011	1101
0010	1001	0100
1111	0101	0001
		1

DECIMAL ARITHMETIC

(OR, VARIABLE LENGTH, PACKED BCD.)

* EACH DECIMAL DIGIT REPRESENTED BY A 4 BIT CODE

* SPECIAL ALU OR 3 CYCLES PER ITERATION

* A VALUE REQUIRES TWO ELEMENTS (ADDR, LENGTH)

* EXAMPLE: ADD 283 TO 598

WITH BINARY ALU IN ONE CYCLE:

```
  0010 1000 0011
  0101 1001 1000
  --------------
  1000 0001 1011
```

GARBAGE → 8 1 B

WHY?

WITH CONSTANT 666 AND THREE CYCLES

(1) 283 + 666 → 8E9
(2) 8E9 + 598 → E8*1*
(3) E81 − 600 → 881

$\boxed{\text{WHY SUBTRACT 600?}}$

Fixed Pt. / 11

Residue Arithmetic (Example)

$p_1 = 3$
$p_2 = 4$

$p = 7$
$N = 8$

0	00
1	11
2	22
3	03
4	10
5	21
	⋮

$p_1 = 7$
$p_2 = 8$
$p_3 = 9$

1

$56 \times 9 = \boxed{504}$

12 00

7 8 9

$19 = 5\ 3\ 1$
$24 = 3\ 0\ 6$

$\boxed{1\ 0\ 6}$

19
24
76
38

$\boxed{456}$

19
24
43

3

$456 \div 7 = 65$, remainder 1

$456 \div 9 = 50$, remainder 6

137

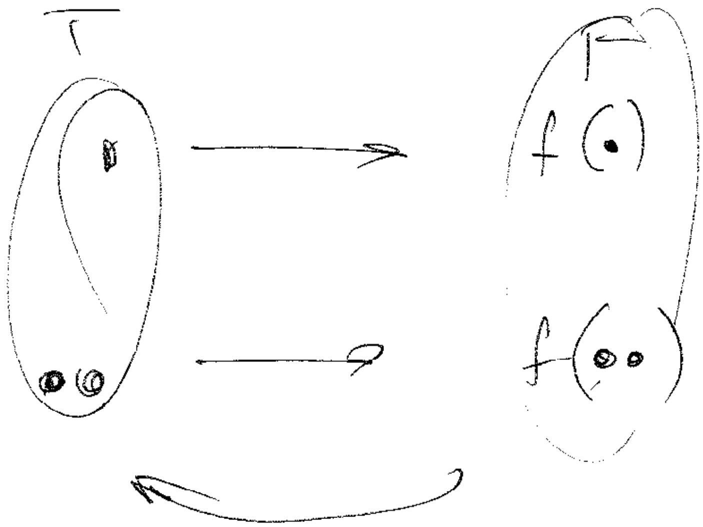
Residue Arithmetic (Proof)

$$A = p_1 m + a_1$$
$$B = p_2 n + a_2$$
$$A \times B =$$

$$A = p_1 m + a_1$$
$$B = p_1 n + b_1$$

$$(p_1 m + a_1)(p_1 n + b_1)$$
$$p_1^2 mn + p_1 m b_1 + p_1 a_1 n + a_1 b_1$$

$$A + B = p_1 m + p_1 n + a_1 + b_1$$



ResidueArithmetic

* When?

- Inputs, outputs are short integers
- Intermediate results may be very large
- Internally compute intensive, as opposed to having to do substantial I/O

* How?

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
a,b \arrow[r, "\text{Transform to Residue Representation}", "\text{(Slow)}"'] & f(a),\ f(b) \arrow[d, "\text{Perform the operation (VERY FAST)}"] \\
a * b & f(a) * f(b) \arrow[l, "\text{Inverse transform}"', "\text{(Slow)}"]
\end{tikzcd}
$$

Residue Arithmetic (Continued)

* In Greater Detail,

  - Pick A Set Of Moduli $p_1, p_2 \cdots p_k$ Such That They Are All Relatively Prime.

  - We Can Represent X As $X_1 X_2 \cdots X_k$ Where $X_i = X \bmod p_i$

  - If $0 \leq X < \prod p_i$, Or More Realistically

    If $-\frac{\prod p_i}{2} \leq X < +\frac{\prod p_i}{2}$,

    Then This Representation For X is <u>Unique</u>

  - From Which $X+Y$ And $X*Y$ Can Be Computed Concurrently By K Processing Elements, Each One Computing The Result $\bmod p_i$.

* Why Don't We Do It?

  - Transformations Expensive
  - Comparisons Unwieldly

Fixed 11/15

RESIDUE ARITHMETIC (EXAMPLES)

AS IN CLASS : $p_1 = 7$, $p_2 = 8$, $p_3 = 9$; $\prod p_i = 504$

FOR THESE EXAMPLES, LET'S USE ONLY POSITIVES.

$$0 \leq X < 503$$

(1) REPRESENTATIONS :

$$19 = 5\ 3\ 1$$
$$24 = 3\ 0\ 6$$

(2) ADDITION :

$$\begin{array}{r} 19 \\ +24 \\ \hline 43 \end{array} \qquad \begin{array}{ccc} 5 & 3 & 1 \\ 3 & 0 & 6 \\ \hline 1 & 3 & 7 \end{array}$$

(3) MULTIPLICATION :

$$\begin{array}{r} 19 \\ 24 \\ \hline 76 \\ 38 \\ \hline 456 \end{array} \qquad \begin{array}{ccc} 5 & 3 & 1 \\ 3 & 0 & 6 \\ \hline 1 & 0 & 6 \end{array}$$

(4) WHY IT WORKS :

$$A * B = (m p_i + a) * (n p_i + b)$$
$$= p_i (m n p_i + a n + b m) + a b$$

$$(A * B) \bmod p_i = a b$$

Residue Arithmetic (Continued)

Inverse Transformation

Let X Be Represented As $X_1 X_2 X_3$.
What Is X?

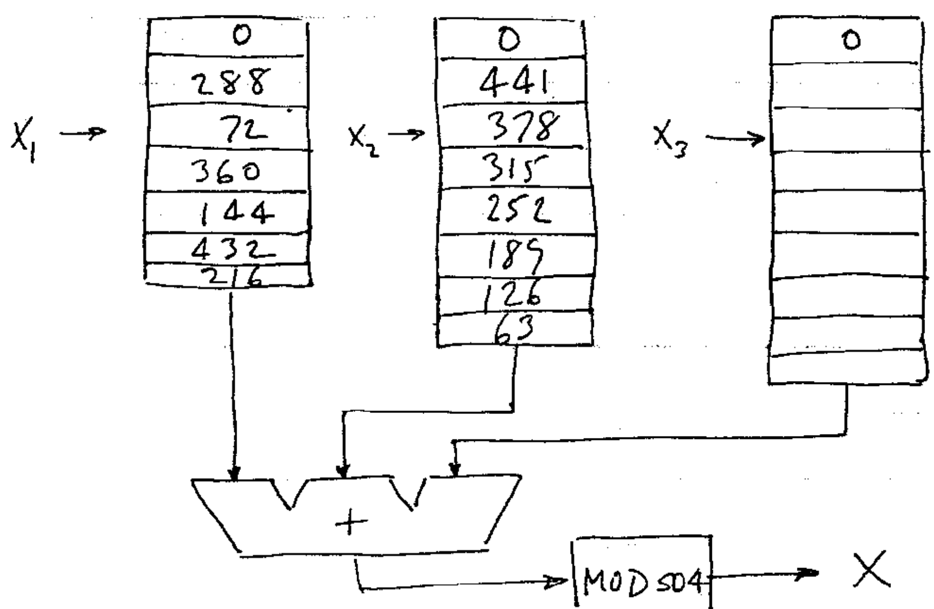
$$X_1 X_2 X_3 = X_1(100) + X_2(010) + X_3(001)$$

100 Is A Multiple Of 72 That Has A Residue Of 1 For P = 7. i.e. 288.

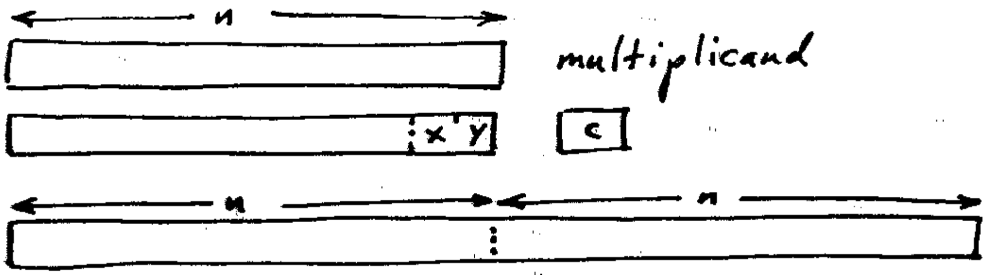
Similarly, 010 Is A Multiple Of 63. i.e. 441
Similarly, 001 Is A Multiple Of 56. i.e. 280.

Thus X Can Be Obtained By Adding
$X_1 * 288 + X_2 * 441 + X_3 * 280$, And
~~B~~ Finding The Residue Mod 504.

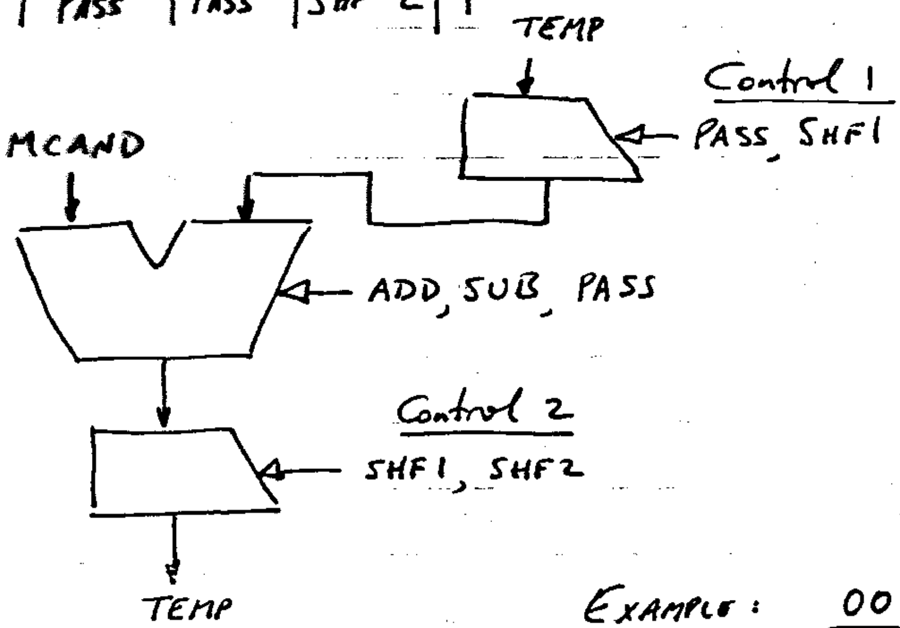
A Simpler Hardware Mechanism:



Booth's Algorithm (A Variation)



x	y	c	Control 1	ALU	Control 2	c'
0	0	0	PASS	PASS	SHF 2	0
0	1	0	PASS	ADD	SHF 2	0
1	0	0	SHF 1	ADD	SHF 1	0
1	1	0	PASS	SUB	SHF 2	1
0	0	1	PASS	ADD	SHF 2	0
0	1	1	SHF 1	ADD	SHF 1	0
1	0	1	PASS	SUB	SHF 2	1
1	1	1	PASS	PASS	SHF 2	1



Example: 00 10 11 = 11

Step 1: 00 11 (−1)

2: 01 (−1)

3: (+1)

$1\times 4^2 - 1\times 4^1 - 1\times 4^0$