COL216
Computer Architecture

Floating Point Arithmetic

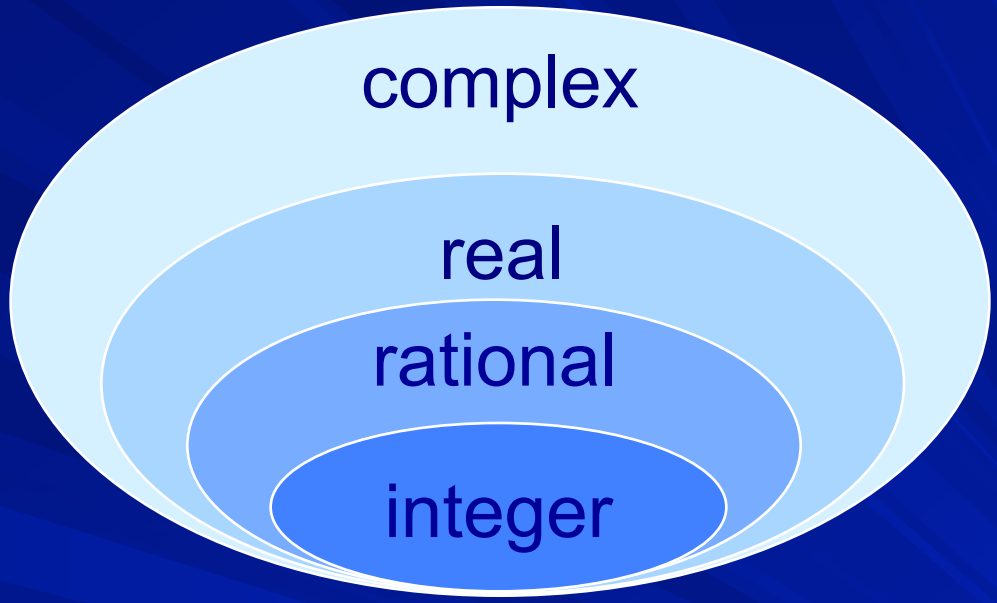
15th, 18th February, 2016

Outline of this lecture

- Need for floating point numbers
- Representation
- IEEE 754 standard
- Overflow and underflow
- Floating point operations
- Accuracy
- Special numbers

Need to go beyond integers

- integer 7
- rational 5/8
- real $\sqrt{3}$
- complex 2 - 3 i



Extremely large and small values:

- distance pluto - sun = $5.9\ 10^{12}$ m
- mass of electron = 9.1×10^{-28} gm

Representing fractions

- Integer pairs (for rational numbers)

5	8	= 5/8

- Strings with explicit decimal point

-	2	4	7	.	0	9

- Implicit point at a fixed position

0	1	0	0	1	1	0	1	0	1	1	0	0	0	1	0	1	1

implicit point

- Floating point

  fraction x base $^{\text{power}}$

Numbers with binary point

$101.11 = 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$

$= 4 + 1 + 0.5 + 0.25 = 5.75_{10}$

$0.6 = 0.1001\mathbf{1001}1001\mathbf{1001}1001.....$

$.6 \times 2 = 1 + .2$

$.2 \times 2 = 0 + .4$

$.4 \times 2 = 0 + .8$

$.8 \times 2 = 1 + .6$

FP numbers with base = 2

$(-1)^S \times F \times 2^E$

S = sign

F = fraction (fixed point number)

usually called mantissa or significand

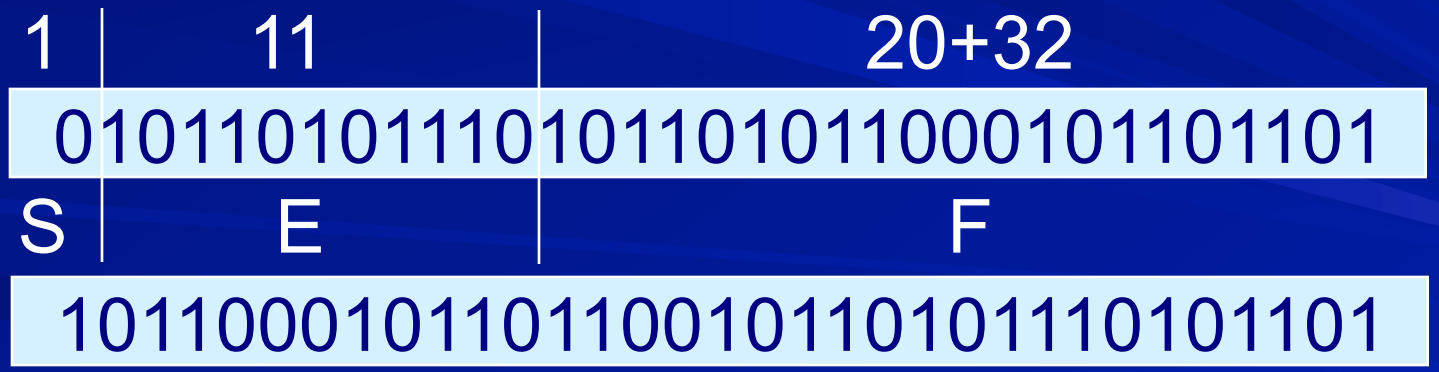
E = exponent (positive or negative integer)

- How to divide a word into S, F and E?
- How to represent S, F and E?

IEEE 754 standard

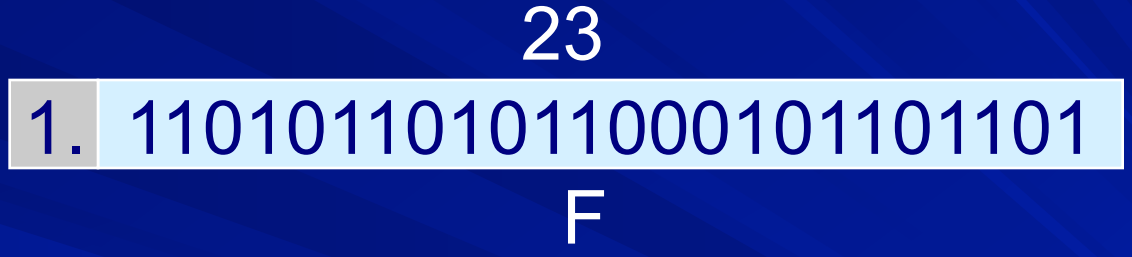
- Single precision numbers

- Double precision numbers

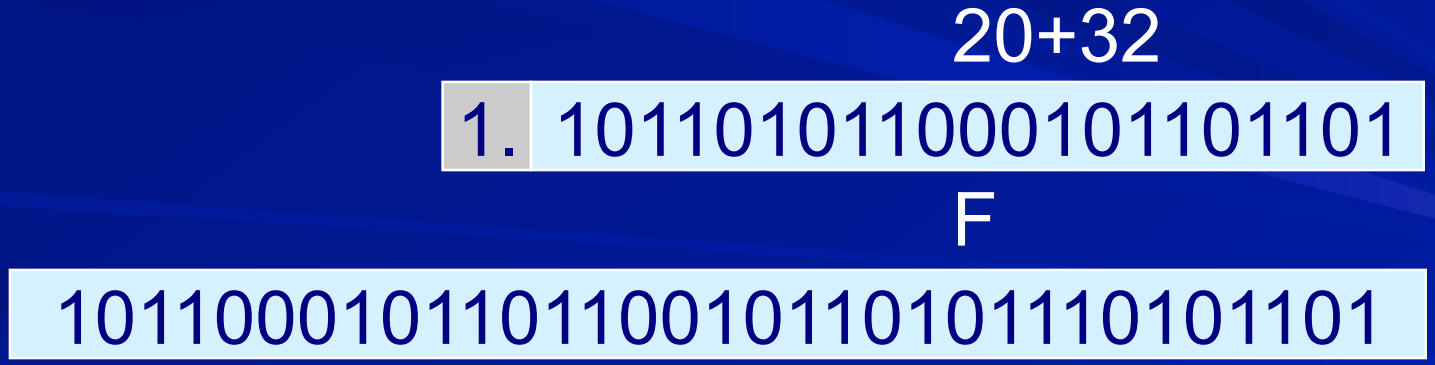


Representing F in IEEE 754

- Single precision numbers



- Double precision numbers



Value range for F

- Single precision numbers

$1 \leq F \leq 2 - 2^{-23}$ or $1 \leq F < 2$

- Double precision numbers

$1 \leq F \leq 2 - 2^{-52}$ or $1 \leq F < 2$

These are “normalized”.

Representing E in IEEE 754

- Single precision numbers

8

10110101 — 54

E — (+ bias 127)

- Double precision numbers

11

10110101110 — 431

E — (+ bias 1023)

Value range for E

- Single precision numbers

  $-126 \leq E \leq 127$

  (all 0's and all 1's have special meanings)
- Double precision numbers

  $-1022 \leq E \leq 1023$

  (all 0's and all 1's have special meanings)

Representing zero

- All bits of F are zero, no explicit “1” to the left

  (not a normalized value)
- E can have any value, we choose to have all bits zero
- Sign bit is also zero

- Exactly same as integer zero!

Testing and comparing

- Test for zero, negative and positive for FP numbers is same as that for integers
- Magnitude comparison of FP numbers is same as magnitude comparison for integers

  Why?

  Because exponent is in biased notation and is located to the left of mantissa

Overflow and underflow

largest positive/negative number (SP) = $\pm(2 - 2^{-23}) \times 2^{127} \cong \pm 2 \times 10^{38}$

smallest positive/negative number (SP) = $\pm 1 \times 2^{-126} \cong \pm 2 \times 10^{-38}$

largest positive/negative number (DP) = $\pm(2 - 2^{-52}) \times 2^{1023} \cong \pm 2 \times 10^{308}$

smallest positive/negative number (DP) = $\pm 1 \times 2^{-1022} \cong \pm 2 \times 10^{-308}$

Floating point operations

- Add/subtract

$[(-1)^{S1} \times F1 \times 2^{E1}] \pm [(-1)^{S2} \times F2 \times 2^{E2}]$

suppose E1 > E2, then we can write it as

$[(-1)^{S1} \times F1 \times 2^{E1}] \pm [(-1)^{S2} \times F2' \times 2^{E1}]$

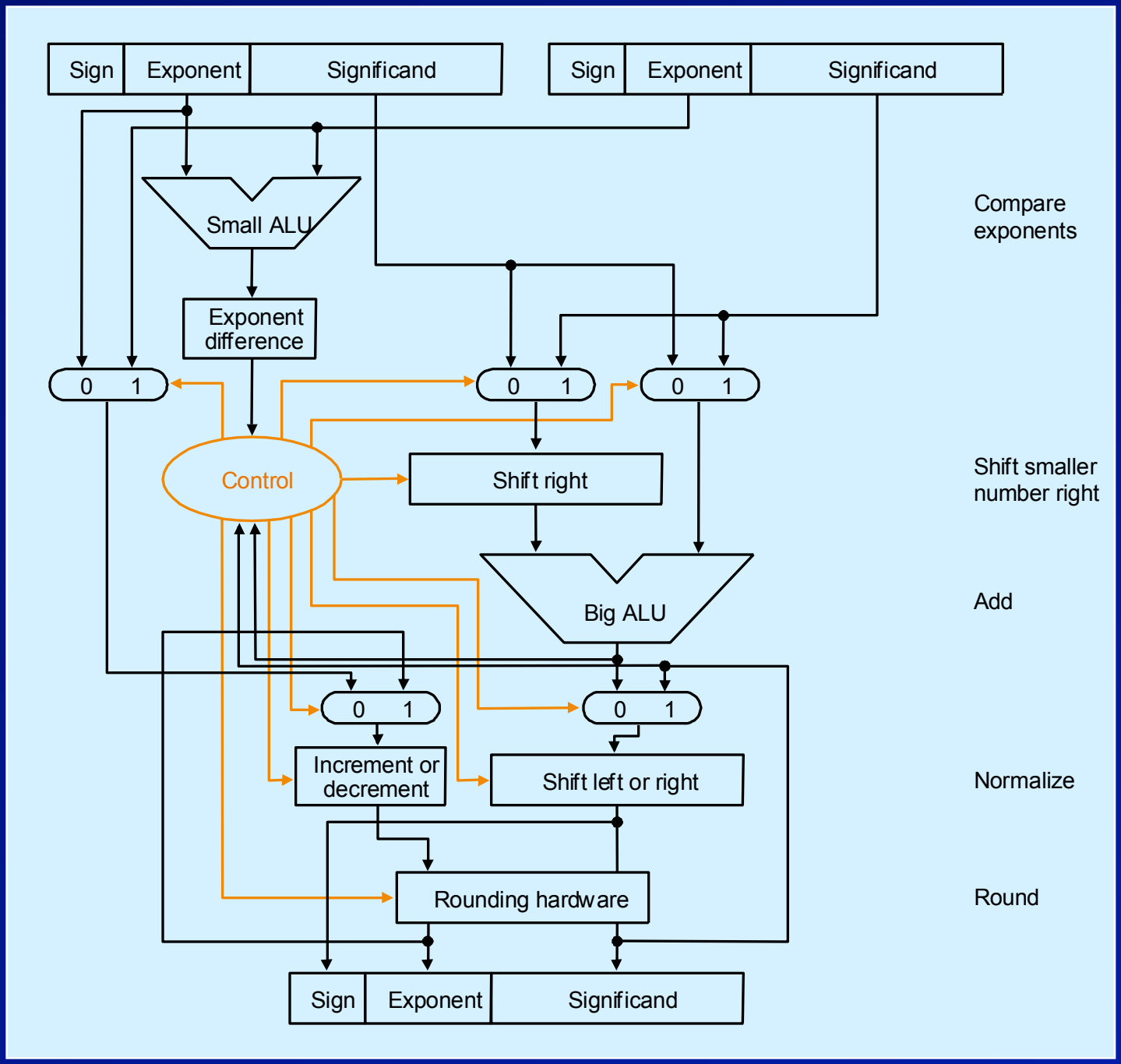
where $F2' = F2 / 2^{E1-E2}$,

The result is

$(-1)^{S1} \times (F1 \pm F2') \times 2^{E1}$

It may need to be normalized

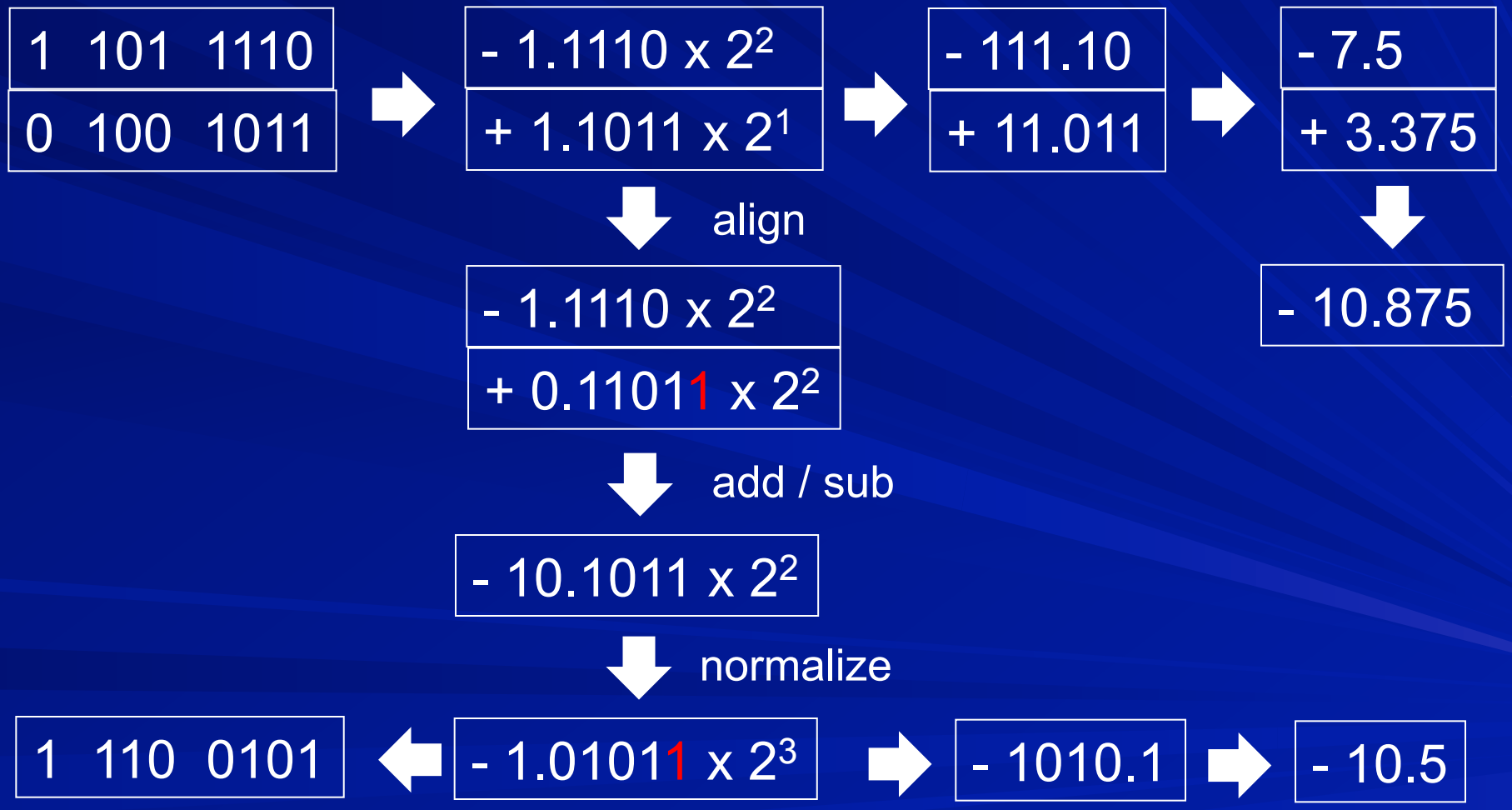
FP Add/Sub Unit



Small word size example

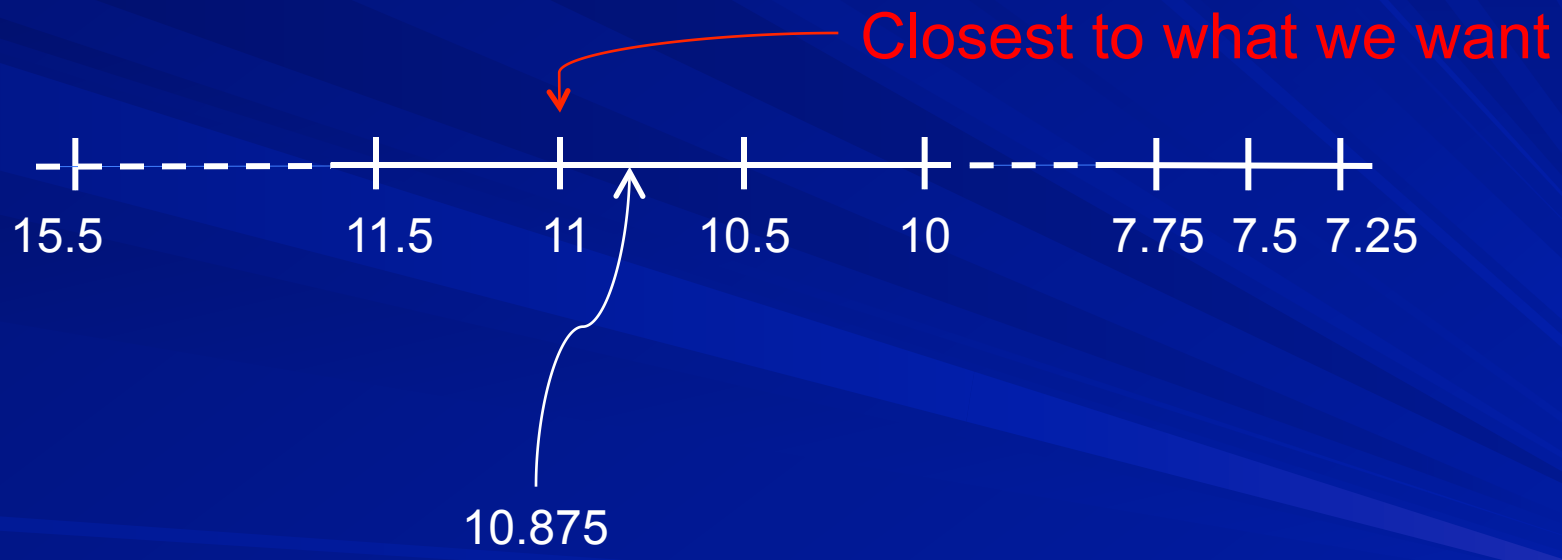
- Range of E = -2 to 3
  - represented by 001 to 110 (bias = 3)
- Range of F = 1.0 to 1.9375
  - represented by 1.0000 to 1.1111
- resolution = .0625 x 2^E

FP subtraction example

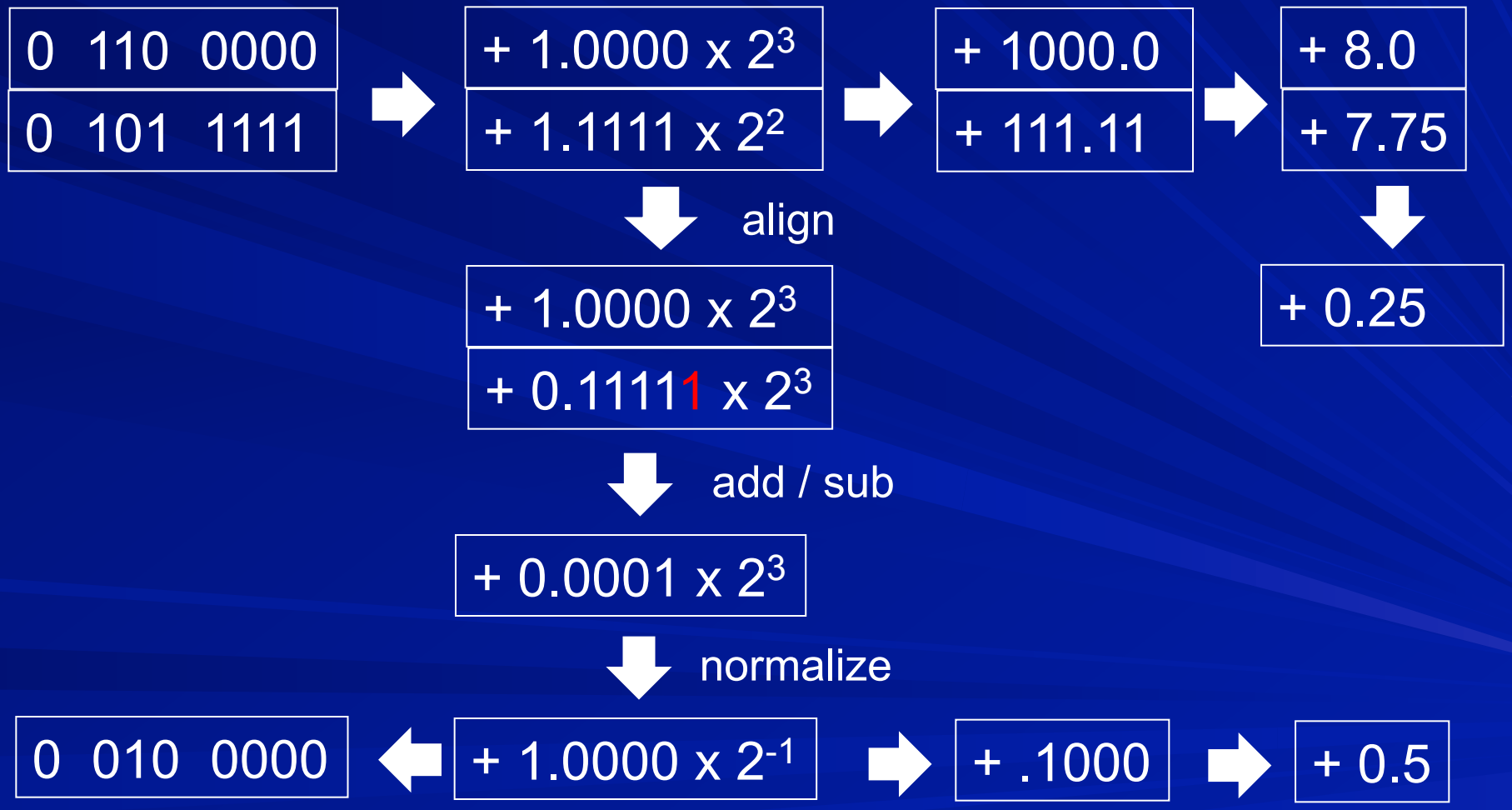


Can we do better?

The values that can be represented



Another FP subtraction example



Can we represent 0.25 ?

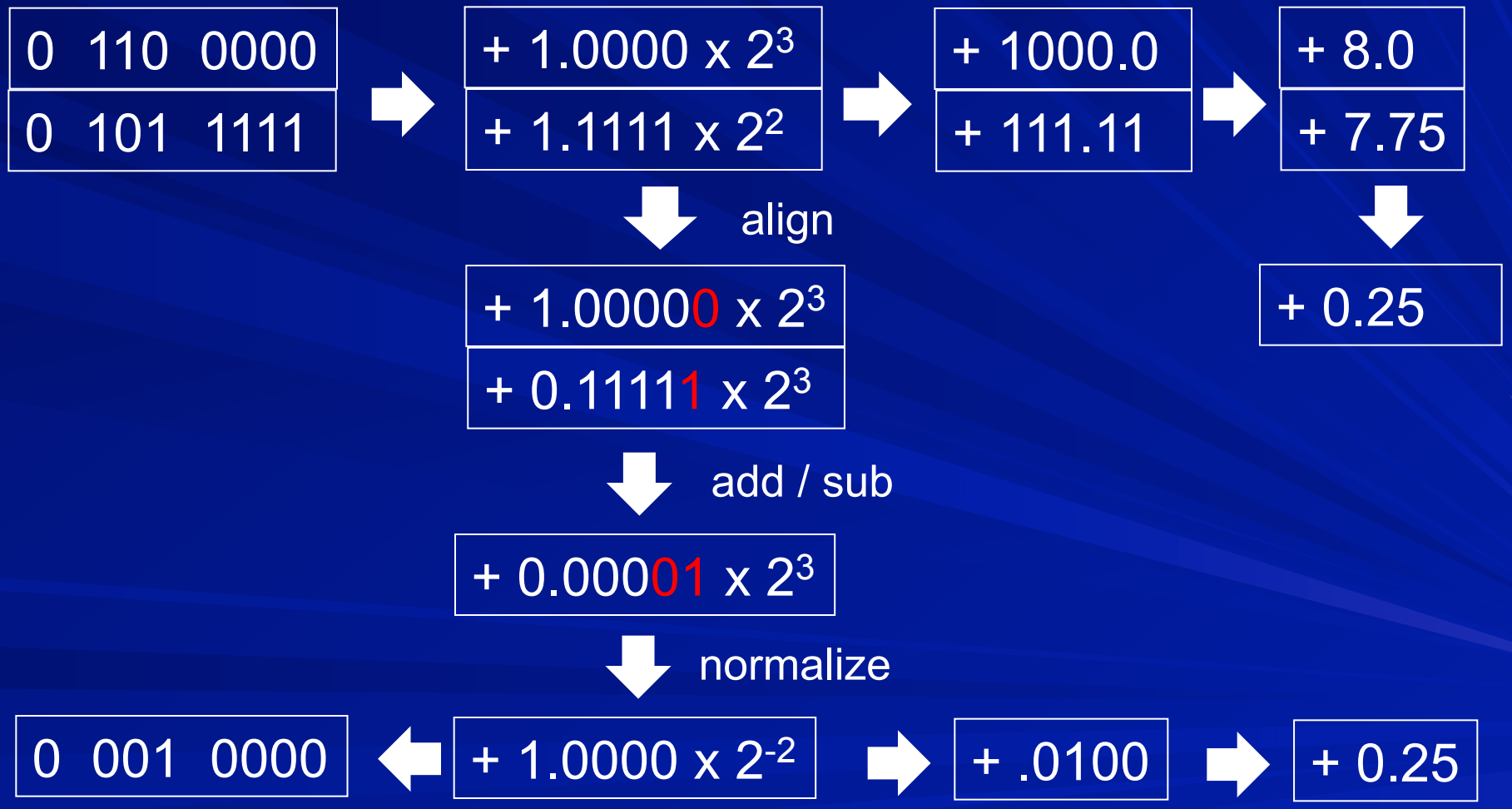
Yes.

$0.25 = 1.0 \times 2^{-2}$

0 0 0 1 0 0 0 0

So where is the problem?

If we had more bits to work with…



How many bits more?

- How many bits we lose in alignment that can potentially be recovered during normalization?
- many bits lost during alignment when difference in operands is large
- many bits brought in during normalization when difference is small

Improving precision of computation

- Precision is lost when some bits are shifted to right of the rightmost bit or are thrown
- Three extra bits are used internally -

  G (guard), R (round) and S (sticky)
  - G and R are simply the next two bits after LSB
  - S = 1 iff any bit to right of R is non-zero

1.	11010110101100010110110 1	GRS

Rounding using G, R and S

- if G=1 & R=1, add 1 to LSB
- if G=0 & R=0 or 1, no change
- if G=1 & R=0, look at S
  - if S=1, add 1 to LSB
  - if S=0, round to the nearest “even”
    i.e., add 1 to LSB if LSB = 1

Floating point operations

- Multiply

$[(-1)^{S1} \times F1 \times 2^{E1}] \times [(-1)^{S2} \times F2 \times 2^{E2}]$

$= (-1)^{S1 \oplus S2} \times (F1 \times F2) \times 2^{E1+E2}$

Since $1 \leq (F1 \times F2) < 4$,

the result may need to be normalized

Floating point operations

- Divide

$[(-1)^{S1} \times F1 \times 2^{E1}] \div [(-1)^{S2} \times F2 \times 2^{E2}]$

$= (-1)^{S1 \oplus S2} \times (F1 \div F2) \times 2^{E1-E2}$

Since $.5 < (F1 \div F2) < 2$,

the result may need to be normalized

(assume $F2 \neq 0$)

Special numbers

Single precision		Double precision		object
exponent	mantissa	exponent	mantissa	
0	0	0	0	zero
0	nonzero	0	nonzero	denorm
1-254	any	1-2046	any	norm
255	0	2047	0	infinity
255	nonzero	2047	nonzero	NaN