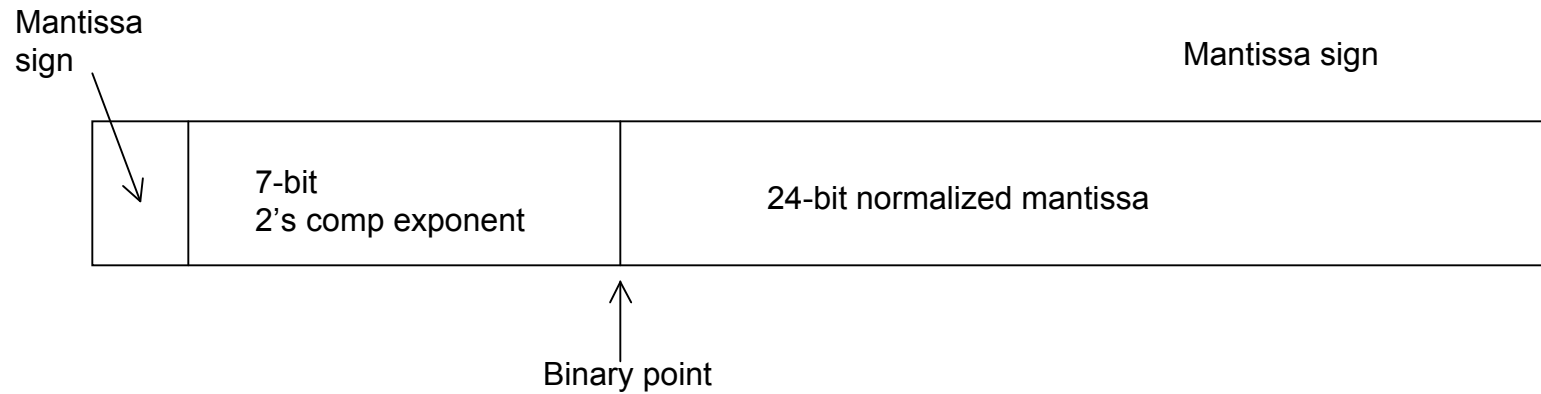


Floating Point Numbers

- Consider 7 digits (decimal) -- 10^7 values
- Unsigned Range $0 \rightarrow 10^7-1$
- Signed Range $-(10^7/2) \rightarrow (10^7-1)/2$
- If we dedicate 2 digits for exponent and 5 for mantissa:
 - $-(10^5/2 \times 10^{99}) \rightarrow ((10^5-1)/2 \times 10^{99})$
 - Larger range, but lower precision
 - Avg Precision =
(largest-smallest)/(number of values)

Typical Representation (32 bits)



Binary fractions (revisited)

$$B = b_{n-1} b_{n-2} \dots b_0 \text{ } \blacksquare \text{ } b_{-1} b_{-2} \dots b_{-n}$$

$$= b_{n-1}x2^{n-1} + b_{n-2}x2^{n-2} + \dots b_0x2^0 + \\ b_{-1}x2^{-1} + b_{-2}x2^{-2} + \dots b_{-n}x2^{-n}$$

E.g.,

$$.25_{\text{ (base 10) }} = .01_{\text{ (base 2) }} \dots \text{ but } \dots$$

$$.10_{\text{ (base 10) }} = .00011001100110011\overline{0011} \dots \text{ (base 2) }$$

2's Complement of Binary Fractions

- $4.75_{(\text{base } 10)} = 0100.110_{(\text{base } 2)}$
- 2's complement of 4.75 (i.e., -4.75)
(n=4) $2^4 - 4.75 = 11.25$ (unsigned)
 $= 1011.010$

Consider: $0100.110_{(\text{base } 2)}$

Bit by bit complement

$$\begin{array}{r} 1011.001 \\ + \quad .001 \\ \hline 1011.010 = -11.25 \text{ (correct)} \end{array}$$

1's Complement of Binary Fractions

- $-4.75_{\text{(base 10)}} = 1011.001$ (bit by bit rule)
- However, general rule $(-A = (2^n - 1) - A)$
($n=4$) $(2^4-1)-4.75 = 10.25 = 1010.010$ (wrong!)

But....

$$(2^4-.001)-4.75 = 11.125 = 1011.001 \text{ (correct)}$$

Sign/Magnitude of Binary Fractions

- $4.75_{\text{(base 10)}} = 0100.110$
- $-4.75_{\text{(base 10)}} = 1100.110$ (bit by bit rule)

...and

- $0.75 = 0.110$
- $-0.75 = 1.110 = \dots 11111.110$ (sign extended)

Normalized Mantissas

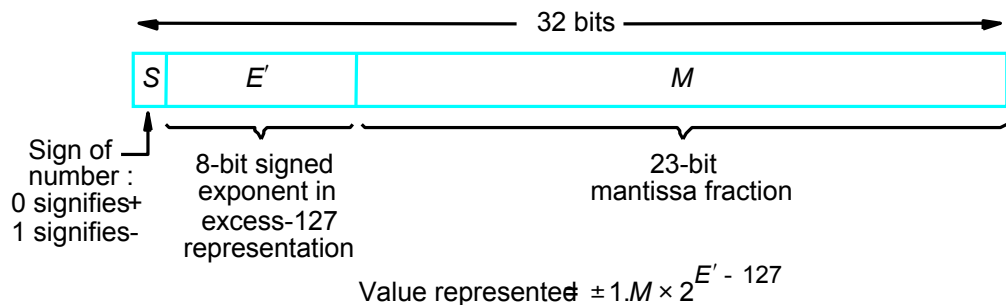
- Consider
 - $.00101110 \times 2^k$
Should be normalized to...
 - $.10111000 \times 2^{k-2}$
- And
 - 101.11000×2^k
 - $.10111000 \times 2^{k+3}$
- Thus, magnitude of stored mantissa M
$$\frac{1}{2} \leq M < 1$$

Exponent Base

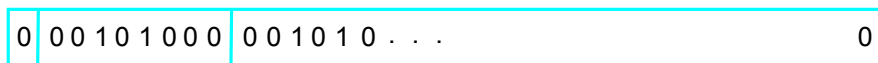
- IEEE format is base 2, but other bases are used
- E.g., IBM 360/91 and later mainframes all used base 16 as well as base 16 mantissa “digits”.

Floating Point Arithmetic

- ADD/SUBTRACT
 1. Align operands (to larger exponent)
 2. Add/subtract mantissas
 3. Re-normalize (adjust exponent)
- MULTIPLY/DIVIDE
 1. Add/subtract exponents
 2. Multiply/divide mantissas
 3. Normalize and “round/truncate/

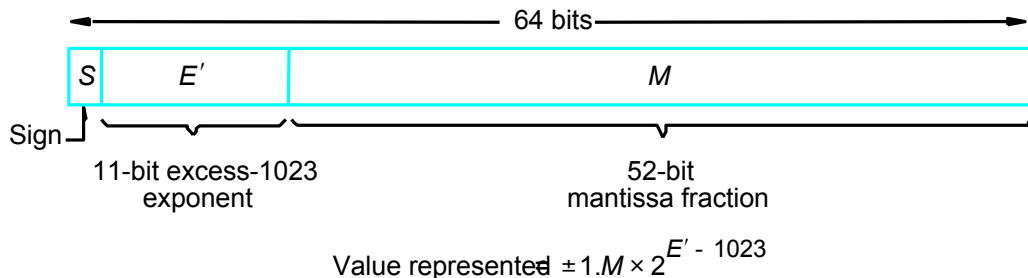


(a) Single precision



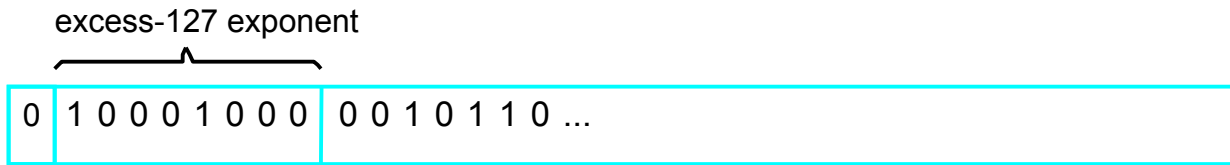
Value represented $1.001010 \dots 0 \times 2^{-87}$

(b) Example of a single-precision number



(c) Double precision

Figure 6.24. IEEE standard floating-point formats.



(There is no implicit 1 to the left of the binary point.)

Value represented = $+0.0010110.. \times 2^9$

(a) Unnormalized value



Value represented = $+1.0110.. \times 2^6$

(b) Normalized version

Figure 6.25. Floating-point normalization in IEEE single-precision format.

32-bit operands $\left\{ \begin{array}{l} A : S_A, E_A', M_A \\ B : S_B, E_B', M_B \end{array} \right\}$

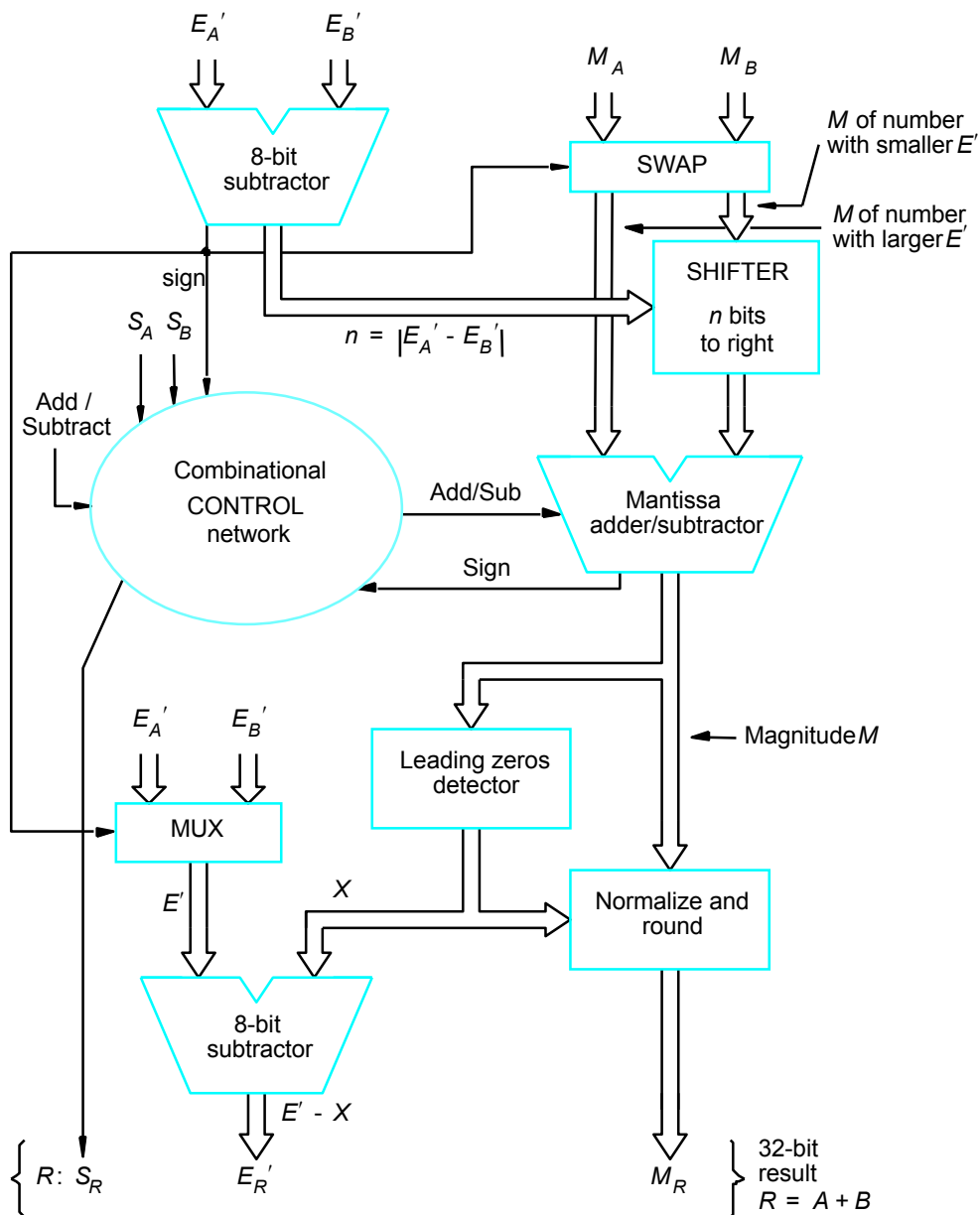


Figure 6.26. Floating-point addition-subtraction unit.

Normalizing Rounding/Truncation

- In the process of aligning and shifting, there may be some loss of significance. How to decide to throw away bits/digits?
- Consider a number of the form $M.D$
- $|M|$ - m digits to be stored
- $|D| = d$ digits to the right - not stored
- 2 quantities to judge a rounding scheme:
 - Average Error
 - Bias

Method 1: Chopping

- M.D -- Ignore D
- $M_{\text{estimate}} = M$
- $0 \leq \text{error} < 1$
- $|\text{error}_{\text{avg}}| \approx 1/2$
- Strongly biased upwards (not symmetrical around 0)

Method 2: Jamming

- M.D -- Force lower order digit of M to '1'
- If M even:
 $M_j > M.D$ by $1-.D$
 $-1 \leq \text{error} < 0$
- If M odd:
 $M_j < M.D$ by $.D$
 $0 \leq \text{error} < 1$
- $-1 \leq \text{error} < 1$ (slightly biased down)
- $|\text{error}_{\text{avg}}| \approx 1/2$
- Much less biased than chopping but slightly higher average error (includes endpoint 1)

Method 3. Von Neumann Rounding

- If $D = 0$
 - Drop D (chop)
- If $D \neq 0$
 - Jam
- If $D = 0$, error = 0
- If $D \neq 0$, $-1 < \text{error} < 1$
- $|\text{error}_{\text{avg}}| = 1/2$ (no bias)

Method 4. Rounding

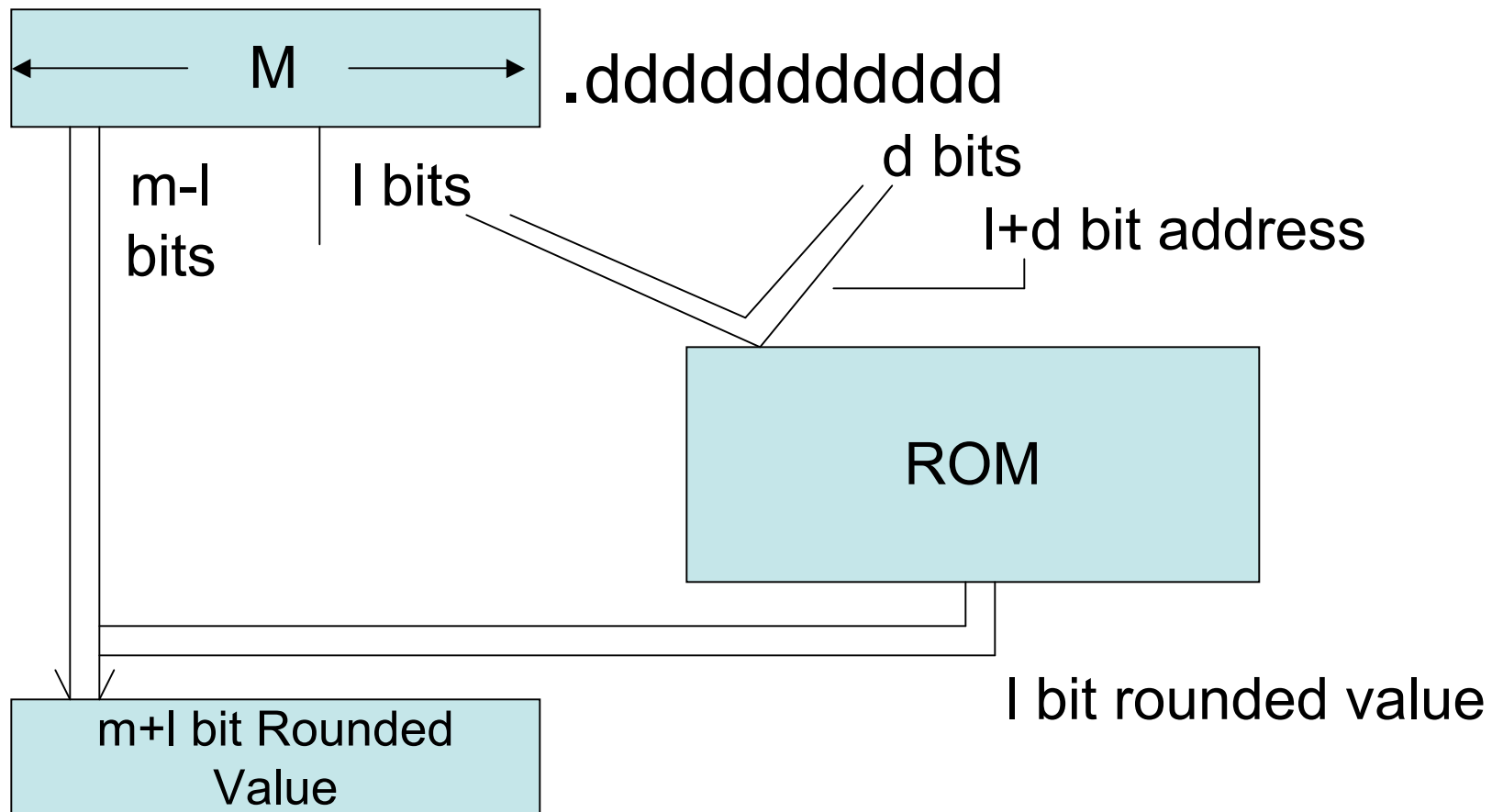
- Add $1/2$ and truncate
- If $0 \leq D < .5 \rightarrow$ chop
 $0 \leq \text{error} < .5$
- If $.5 \leq D < 1 \rightarrow$ (result too big by $1-D$)
 $-.5 \leq \text{error} < 0$
- $-.5 \leq \text{error} < .5$ (slight bias down)
- $|\text{error}_{\text{avg}}| \approx 1/4$

Method 5: R* or Perfect Rounding

- Like Rounding, except when $D = 0.5$ round to even number
 - M even: $M.5 \rightarrow M$; $\text{err} = 0.5$
 - M odd: $M.5 \rightarrow M+1$; $\text{err} = -0.5$
- $-.5 \leq \text{error} \leq .5$ (no bias)
- $|\text{error}_{\text{avg}}| = 1/4$
- Slightly higher Average error than rounding?

Method 6: ROM Rounding

- Problem with either previous rounding scheme is the ripple carry of adding $1/2$



Note: when $1 \text{ bits} = 11111\dots 1$, always rounded down to avoid ripple carry

How much D?

- In reality, large numbers of digits of D are not stored during arithmetic operations.

- Guard digits:

M.ddddddd $\rightarrow$ M.G

$D < 1/2 \rightarrow G=0$

$D \geq 1/2 \rightarrow G=1$

- Sticky Bit

M.G S

S is set if any 1 it is shifted through the LSB position

- $D > 1/2$ if G & S

$D = 1/2$ If G & S

$D < 1/2$ if G

Division by Repeated Multiplication

- Problem $Q = N/D$

Approach: Find a series of factors:

$R_0, R_1, R_2 \dots R_k$ such that

$$F = D \cdot R_0 \cdot R_1 \cdot R_2 \cdot \dots R_k \rightarrow 1$$

$$\begin{aligned} \text{Then } Q &= NF/D = N(D \cdot R_0 \cdot R_1 \cdot \dots R_k)/D \\ &= N \cdot R_0 \cdot R_1 \cdot \dots R_k \end{aligned}$$

Division by Repeated Multiplication (cont)

- Assume a bit-normalized fractional mantissa $M = .1xxxxxx$ (this is D)

Notice, $D \geq 1/2$, thus D can be expressed in the form $D = 1 - x$, where $0 < x \leq 1/2$

- Choose $R_0 = 1 + x$
then $D \cdot R_0 = 1 - x^2 = F_1$
 $0 < x^2 \leq 1/4$

so

$$3/4 \leq F_1 < 1$$

Division by Repeated Multiplication (cont)

- Choose $R_1 = 1 + x^2$
then $F_2 = F_1 \cdot R_1 = (D \cdot R_0) \cdot R_1$
 $= 1 - x^4$

$$0 < x^4 \leq 1/16$$

so

$$15/16 \leq F_2 < 1$$

or

$$F_2 = .1111xxxxx$$

Division by Repeated Multiplication (cont)

- Choose $R_2 = 1 + x^4$

$$\begin{aligned}\text{then } F_3 &= F_2 \cdot R_2 = \\ &= 1 - x^8\end{aligned}$$

$$0 < x^8 \leq 1/256$$

so

$$255/256 \leq F_3 < 1$$

or

$$F_3 = .11111111\text{xxxxx}$$

Division by Repeated Multiplication (Algorithm)

- Given N/D_0
- $D_0 = 1 - x$; or $x = 1 - D_0$
- $R_0 = 1 + x = 1 + (1 - D_0) = 2 - D_0$
(2's complement of D_0)

Step 1:

$$Q_1 = N \cdot R_0$$

$$F_1 = D_0 \cdot R_0$$

$$R_1 = 2\text{'s comp of } F_1$$

$$\begin{aligned} \text{Since } F_1 &= 1 - x^2 \\ R_1 &= 2 - (1 - x^2) \\ &= 1 + x^2 \quad \checkmark \end{aligned}$$

Division by Repeated Multiplication (Algorithm)

Step 2:

$$Q_2 = Q_1 \cdot R_1$$

$$F_2 = F_1 \cdot R_1$$

$$R_2 = 2\text{'s comp of } F_2$$

...

Step 5:

$$Q_5 = Q_4 \cdot R_4$$

$$F_5 = F_4 \cdot R_4$$

$$R_5 = 2\text{'s comp of } F_5$$

Final Step:

$$Q_6 = Q_5 \cdot R_5 \leftarrow \text{Answer in}$$

11 - multiplications and

6 - 2's comp operations