

Binary Floating-Point Numbers



■ $F = (-1)^s M \beta^E$

■ Significand M

- pure fraction $[0, 1\text{-ulp}]$
or $[1, 2)$ for $\beta=2$

■ Normalized form

- significand has no leading zeros
 - ▲ maximum # of significant digits
 - ▲ easy comparison
 - ▲ $0.01101 \times 2^{110} \Rightarrow 0.11010 \times 2^{101}$
- normalization for $\beta=2^k$
 - ▲ at least one nonzero bit in the first k position
 - ▲ $0.00000110 \times 16^{101} \Rightarrow 0.01100000 \times 16^{100}$
- $M_{\min}=1/\beta$ (0.100...)
 $M_{\max}=1-\text{ulp}$ (0.111...)

■ Biased exponent

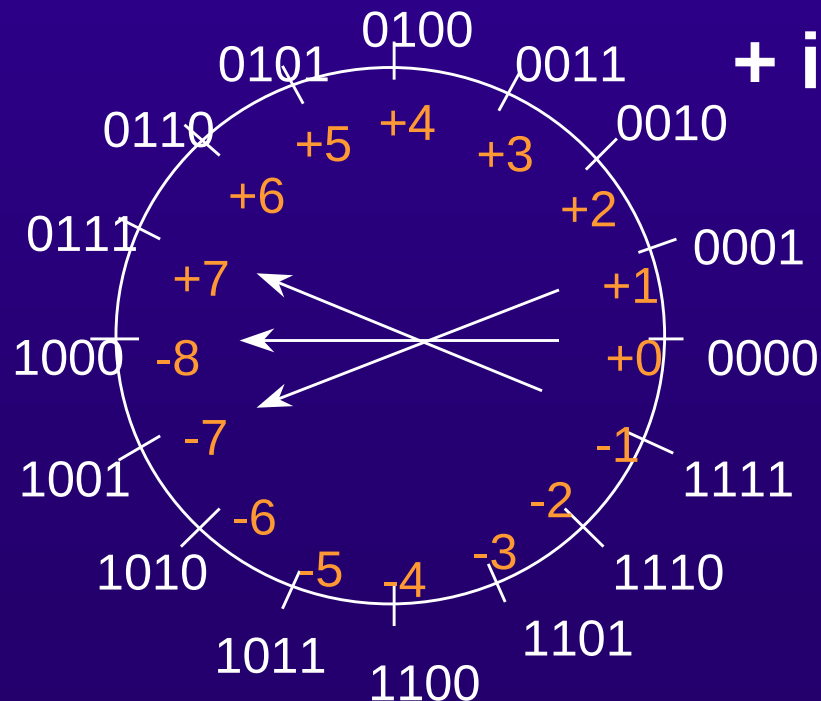
- $E = E^{\text{true}} + \text{bias}$

- $-2^{e-1} \leq E^{\text{true}} \leq 2^{e-1}-1$

- if $\text{bias} = 2^{e-1}$, $0 \leq E \leq 2^e - 1$

$\Rightarrow$ **excess 2^{e-1} code $\equiv$ two's complement**

+ inversion of sign bit



zero: $M=0$, $E=0$

- **biased exponent: can compare exponent as if they were unsigned numbers**
 - **+normalization: compare exponents, then compare significands**
- ⇒ compare floating-point numbers as if they were integers in signed-magnitude representation**



■ Range

- Positive floating-point number F^+

$$M_{\min} \beta^{E_{\min}} \leq F^+ \leq M_{\max} \beta^{E_{\max}}$$

$E > E_{\max}$: exponent overflow

$E < E_{\min}$: exponent underflow

- range of $F^+ = \text{range of } -F^-$

- **IBM 370**



$$\beta=16$$

$$F = (-1)^S M 16^{E-64}$$

$$F_{\min}^+ = 16^{-1} \times 16^{-64} \approx 5.4 \times 10^{-79}$$

$$F_{\max}^+ = (1 - 16^{-6}) \times 16^{63} \approx 7.23 \times 10^{75}$$

- **DEC/VAX**

1	8	23
S	exponent E	significand f

$$\beta=2$$

$$F = (-1)^S 0.1f 2^{E-128}$$

E = 0 is reserved for zero

$$F_{\min}^+ = 0.1 \times 2^{1-128} = 2^{-128}$$

$$F_{\max}^+ = (1-2^{-24}) \times 2^{255-128} = (1-2^{-24}) \times 2^{127}$$

■ Floating-point operations

- Multiplication ($F_3 = F_1 \times F_2$)

- ▲ sign bit

$$S_3 = S_1 \oplus S_2$$

- ▲ exponent

$$E_3 = E_1 + E_2 - \text{bias}$$

$$\Rightarrow E_3 > E_{\max} \rightarrow \text{overflow}$$

$$E_3 < E_{\min} \rightarrow \text{underflow}$$

- ▲ significand

$$1/\beta \leq M_i < 1 \Rightarrow 1/\beta^2 \leq M_1 M_2 < 1$$

- ▲ postnormalization

if $M_3 = M_1 M_2 < 1/\beta$, shift left M_3 and decrease E_3 (may cause underflow)

- **Division ($F_3 = F_1 / F_2$)**
 - ▶ **exponent**
 $E_3 = E_1 - E_2 + \text{bias}$
 - ▶ **significand**
 $1/\beta \leq M_i < 1 \Rightarrow 1/\beta \leq M_1 / M_2 < \beta$
 - ▶ **postnormalization**
if $M_3 = M_1 / M_2 \geq 1$, shift right M_3 and increase E_3 (may cause overflow)
 - ▶ $M_2 = 0$: division by zero
 - ▶ $M_1 = M_2 = 0$: undefined

- **Addition/Subtraction**
 - ▶ Shift right the significand of the smaller operand by $|E_1 - E_2|$ base- β positions
 - ▶ addition: $1/\beta \leq M < 2$
 - if $M \geq 1 \Rightarrow$ postnormalization
 - ▶ subtraction: $0 \leq |M| < 1$
 - if $|M| \leq 1/\beta \Rightarrow$ postnormalization

example

$$F_1 = (0.100000)_{16} \cdot 16^3$$

$$F_2 = (0.FFFFFFF)_{16} \cdot 16^2$$

$$F_1 = 0.1000000 \cdot 16^3$$

F₂ aligned **0. 0 F F F F F · 16³**

$$F_1 - F_2 = 0.0000001 \cdot 16^3$$

postnormalization 0. 1 0 0 0 0 0 · 16⁻²

guard digit

F₁ **0. 1 0 0 0 0 0 0 0 · 16³**

F₂ aligned **0. 0 F F F F F F · 16³**

$F_1 - F_2$ 0. 0 0 0 0 0 0 0 **1** · 16^3

postnormalization 0. 1 0 0 0 0 0 0 **0** · 16⁻³

■ IEEE floating-point standard

- Single precision



$$F = (-1)^S \cdot 1.f \cdot 2^{E-127} \quad (1 \leq E \leq 254)$$

$$F_{\max}^+ = (2 - 2^{-23}) \cdot 2^{254-127} = (1 - 2^{-24}) \cdot 2^{128} \\ \approx 3.8 \cdot 10^{38}$$

$$F_{\min}^+ = 1.0 \cdot 2^{1-127} = 2^{-126} \approx 10^{-38}$$

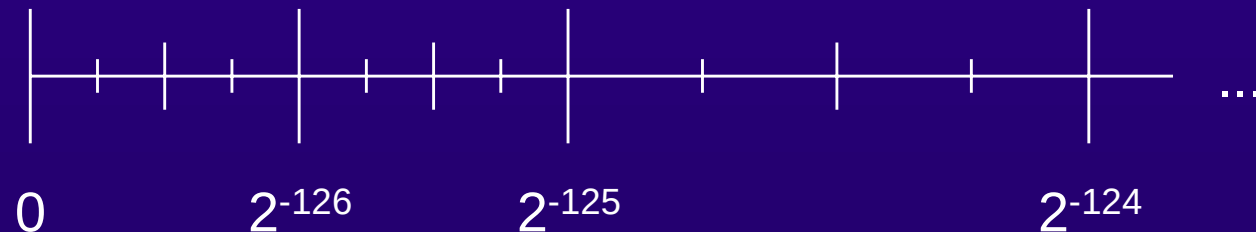
$$\text{Precision} = 2^{-23} \approx 10^{-7}$$

- ▶ $E = 255$ is reserved for $\pm \infty$ and NAN
- ▶ $E = 0$ is reserved for zero and denormalized numbers

zero: $E = 0, f = 0$

denormalized numbers: $E = 0, f \neq 0$

$$F = (-1)^S 0.f 2^{-126}$$



- **Double precision**

1

11

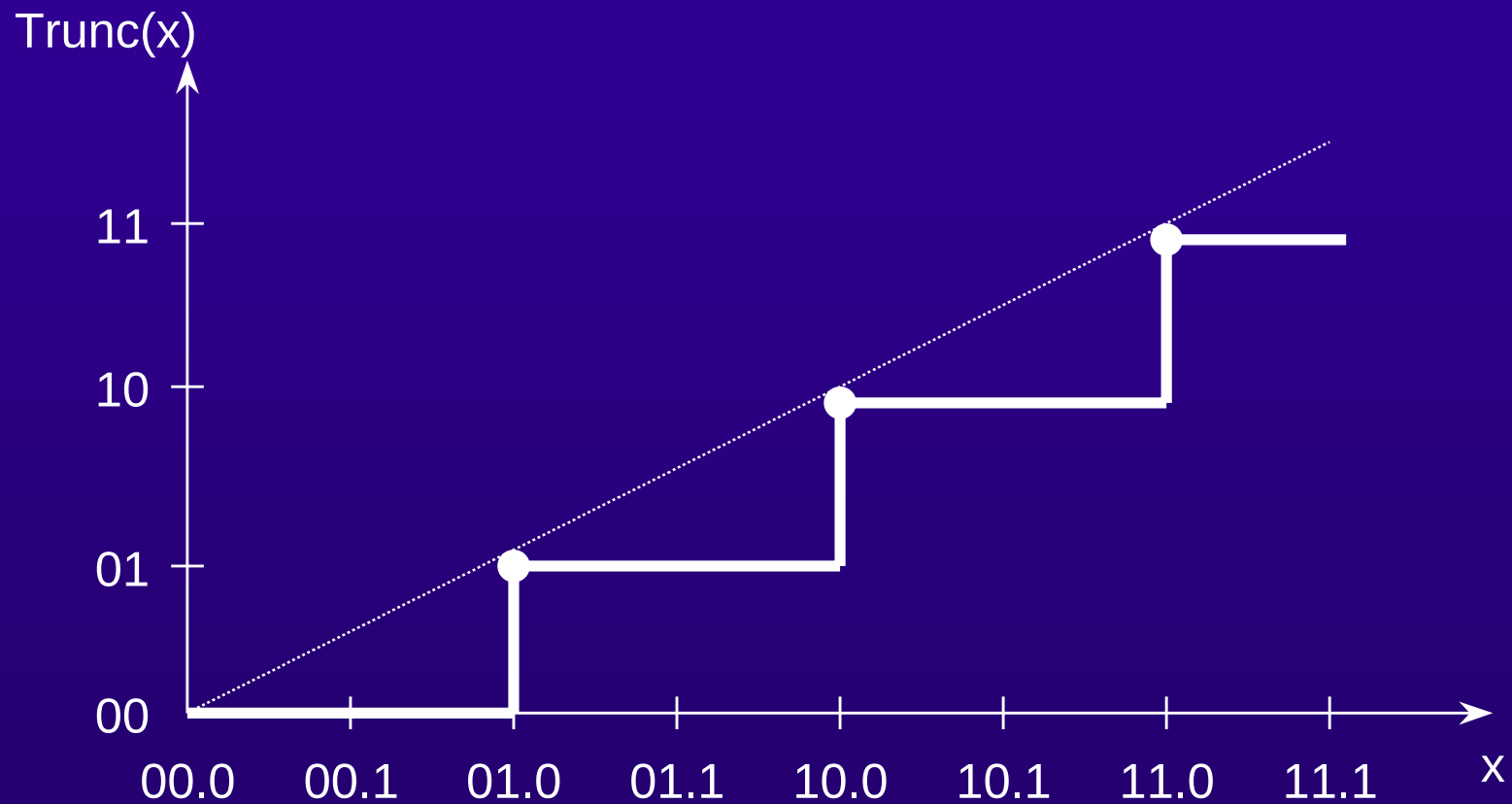
52

S	exponent E	fraction f
---	------------	------------

$$F = (-1)^S \cdot 1.f \cdot 2^{E-1023}$$

■ Round-off schemes

- Truncation



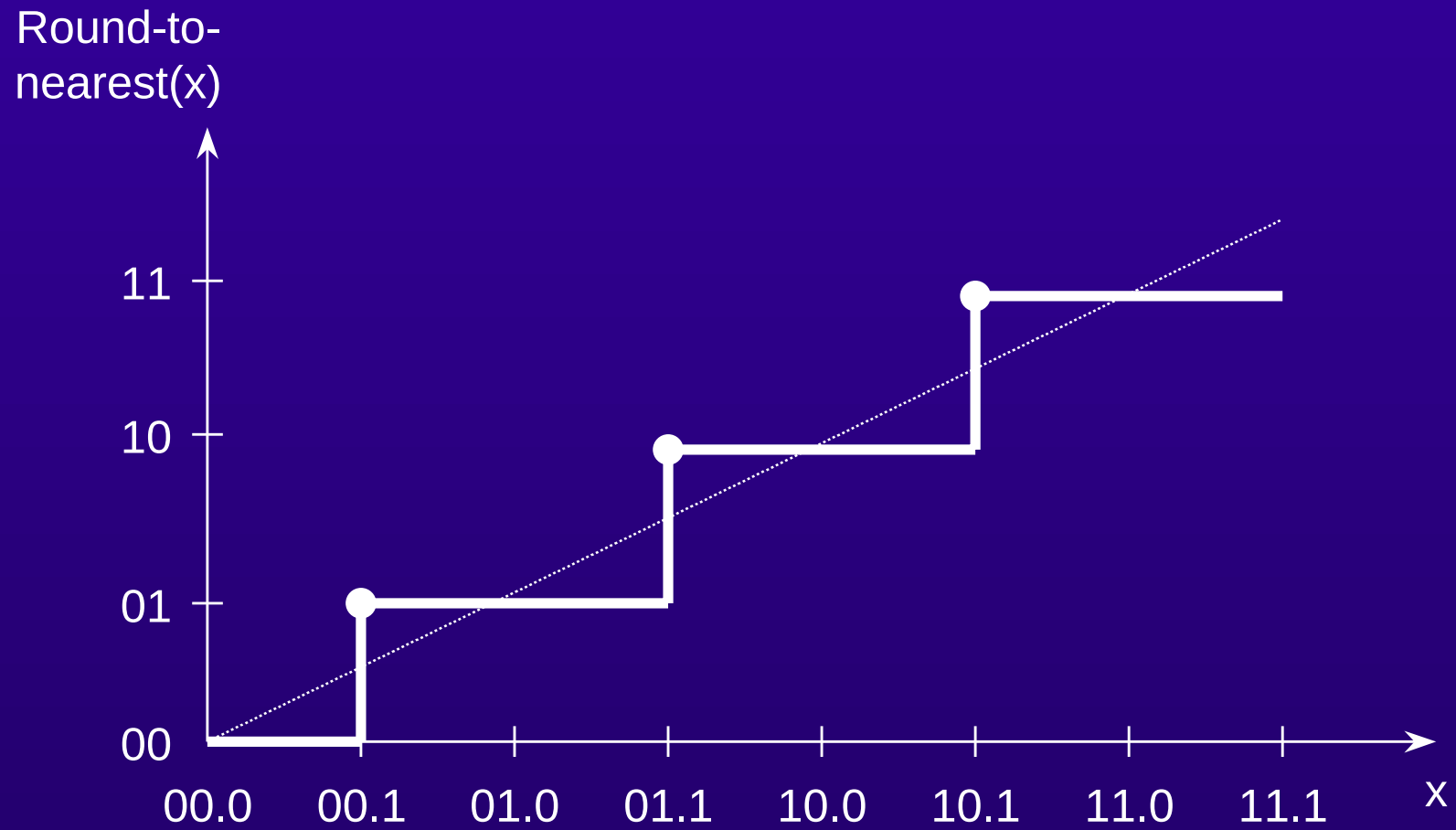
- ▶ **rounding errors for the truncation scheme with $d = 2$ (d : # extra digit)**

Number	Trunc(x)	Error
X.00	X	0
X.01	X	-1/4
X.10	X	-1/2
X.11	X	-3/4

bias = average error

$$= (0 - 1/4 - 1/2 - 3/4)/4 = -3/8$$

- **round-to-nearest**



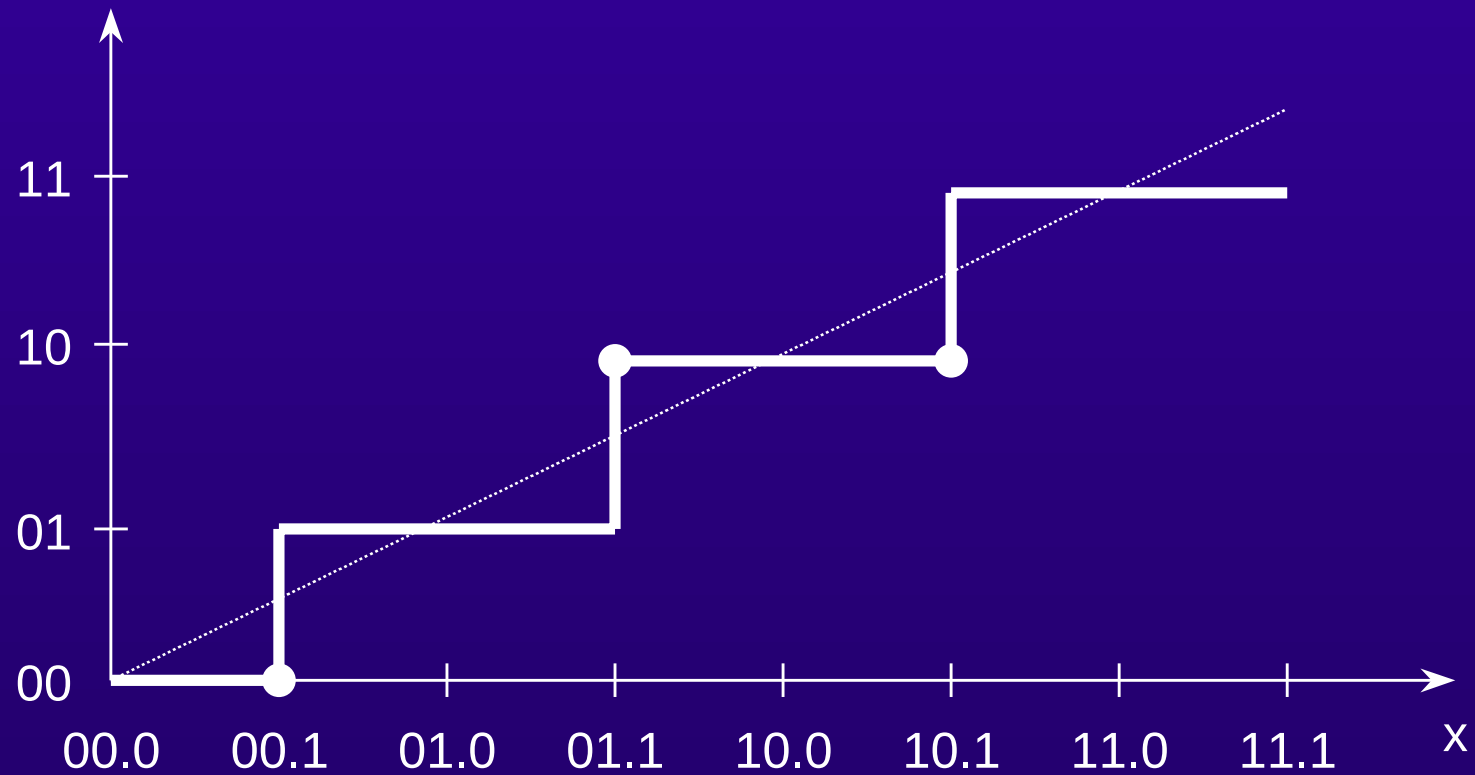
- ▶ rounding errors for the round-to-nearest scheme with $d = 2$

Number	Round-to-nearest(x)	Error
X.00	X	0
X.01	X	-1/4
X.10	X +1	+1/2
X.11	X +1	+1/4

$$\text{bias} = (0 - 1/4 + 1/2 + 1/4)/4 = 1/8$$

- **round-to-nearest-even**

Round-to-
nearest-even(x)

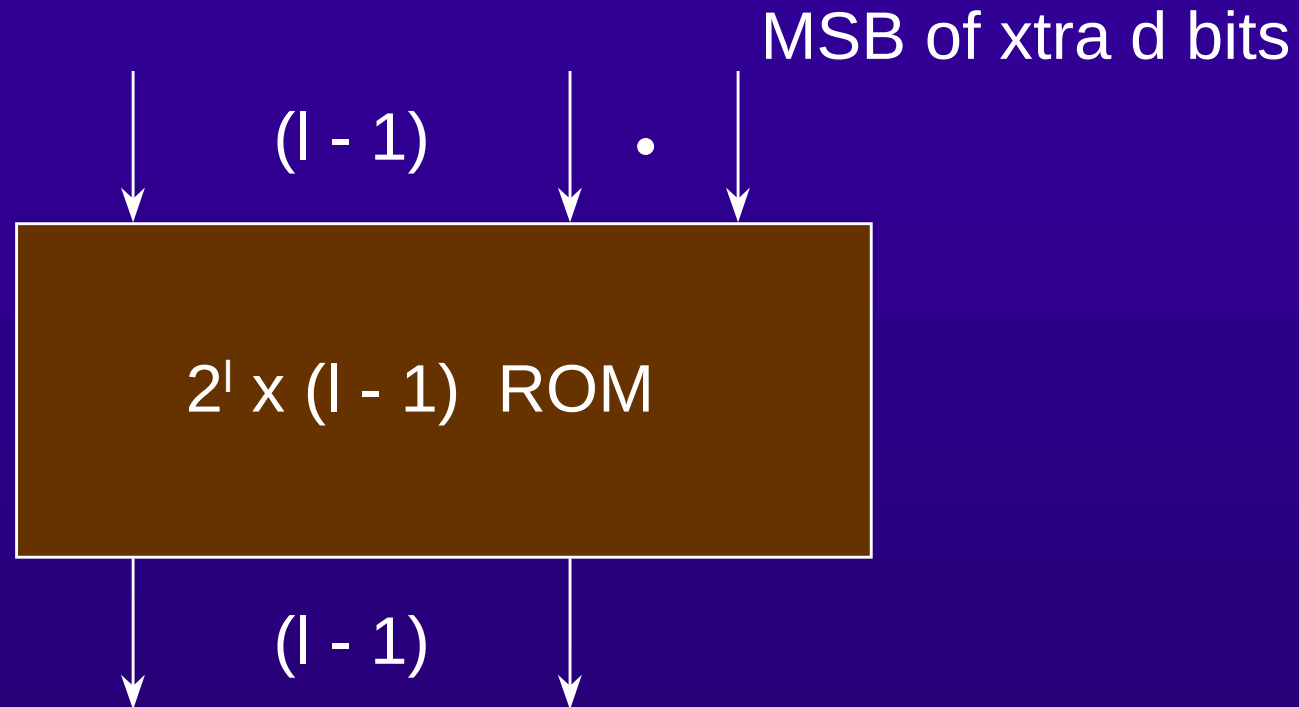


- ▶ rounding errors for the round-to-nearest-even scheme with $d = 2$

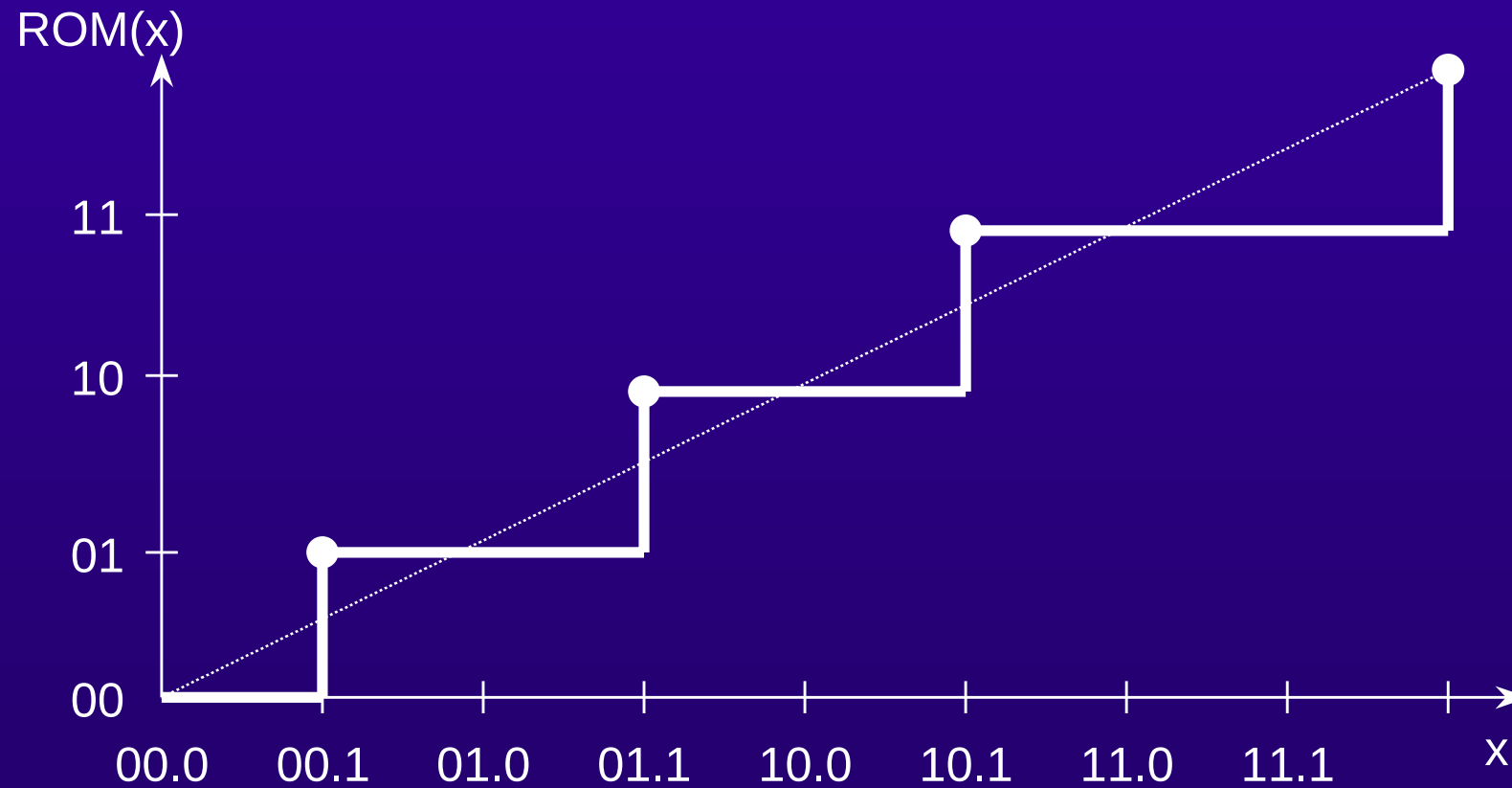
Number	Round-to-nearest-even(x)	Error
X0.00	X0.	0
X0.01	X0.	-1/4
X0.10	X0.	-1/2
X0.11	X1.	+1/4
X1.00	X1.	0
X1.01	X1.	-1/4
X1.10	X1. +1	+1/2
X1.11	X1. +1	+1/4

bias = 0

- ROM rounding
 - avoid add operation



- round-to-nearest using ROM



- ▶ rounding errors for the ROM rounding scheme with $l = 3$ and $d = 1$

Number	Round-to-nearest(x)	Error
X00.0	X00.	0
X00.1	X01.	+1/2
X01.0	X01.	0
X01.1	X10.	+1/2
X10.0	X10.	0
X10.1	X11.	+1/2
X11.0	X11.	0
X11.1	X11.	-1/2

$$\text{bias} = 1/2[(1/2)^d - (1/2)^{l-1}]$$

■ Guard Digits

- Multiplication

- ▲ $1.f \times 1.f$: at most one shift right

- ▲ $0.f \times 0.f$: at most one shift left

- ⇒ one guard digit(G) for postnormalization

- Division

- ▲ $0.f / 0.f$: at most one shift right

- ▲ $1.f / 1.f$: at most one shift left

- ⇒ one guard digit(G) for postnormalization

- Round-to-nearest needs one more digit
R(round)

- **Round-to-nearest-even needs additional digit S(sticky)**
 - ▲ indicates whether all the additional digits are zero

- **Addition/Subtraction**

- ▶ **no shift for the alignment**

A	0.100000101100	
B aligned	0.100000101000	
A - B	0.000000000100	
postnormalization	0.100000000000	

- ▶ **one bit shift for the alignment**

A	0.100000101100	0
B aligned	0.011111111111	1
A - B	0.000000101100	1
postnormalization	0.101100100000	

one guard bit

▶ shift by two or more bits for the alignment:

A	0.100000101100	00
B aligned	0.001100000001	10
A - B	0.010100101010	10
postnormalization	0.101001010101	

⇒ postnormalization needs at most one bit shift
but one guard bit is not sufficient

A	0.100000101100	000000
B aligned	0.000000110000	000110
A - B	0.01111111011	111010
postnormalization	0.11111111011	

▶ use a sticky bit

A	0.1000000101100	000000
B aligned	0.0000000110000	000110
A - B	0.011111111011	111010
postnormalization	0.111111110111	



		G	S
A	0.1000000101100	0	0
B aligned	0.0000000110000	0	1
A - B	0.011111111011	1	1
postnormalization	0.111111110111		

→ good for truncation scheme

- ▶ round-to-nearest(-even) scheme requires an additional bit

A	0.100000101100	000000
B aligned	0.000000110000	010110
A - B	0.011111111011	101010
postnormalization	0.111111110111	0
round-to-nearest	0.111111110111	



		G	S
A	0.100000101100	0	0
B aligned	0.000000110000	0	1
A - B	0.011111111011	1	1
postnormalization	0.111111110111	1	
round-to-nearest	0.111111111000		

▲ use a round bit

A	0.1000000101100	000000
B aligned	0.0000000110000	010110
A - B	0.011111111011	101010
postnormalization	0.111111110111	0
round-to-nearest	0.111111110111	



		G	R	S
A	0.1000000101100	0	0	0
B aligned	0.0000000110000	0	1	1
A - B	0.011111111011	1	0	1
postnormalization	0.111111110111	0		
round-to-nearest	0.111111110111			

round-to-nearest-even scheme can use S bit

▶ if no postnormalization is required (no shift left), then

- G bit serves as an R bit

- $S_{\text{new}} = R + S_{\text{old}}$

▶ round-to-nearest-even : add ulp to the significand if

$$R \cdot S + R \cdot S \cdot L = R \cdot (S + L) = 1$$

		G	R	S
A	0.1000001010100	0	0	0
B aligned	0.0000000110000	0	1	1
A - B	0.1000000100011	1	0	1
		R	S	
before rounding	0.1000000100011	1	1	
after round-to-nearest	0.1000000100100			