Logical Operator | Bitwise operator

OR	||	a||b	|	OR
AND	&&	a&&b	&	
Not	!	!a	~	

inp 0010 0000 outputs
0000 0001

OR | 0010 0000

0000 1

0010
0000

OR

char a = 'a';

printf ("%d", a);

Hex | ASCII

2^8 = 256

91

a b – z, A B – – Z, 0 9 256

8-bits → 2^8 diff. value

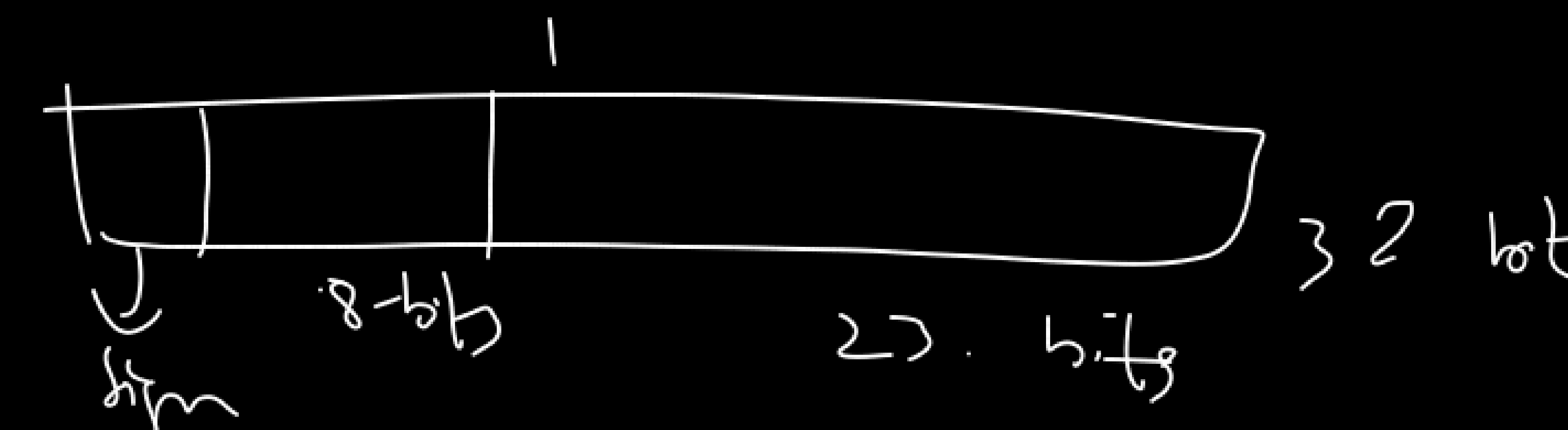
0
1
2
.
.
255

Unicode: 16-bit – 32-bit

Normal form ← $\boxed{1.f \times 2^{exp}}$

f → fraction

$1.f$ → significant.



for Normal form

1.11101×2^{-16}

Subnormal form ← $\boxed{0.111101}$

$0.f \times 2^{exp}$

11 highest

-16 lowest

- 4-bit

32-bit Normal

0 | 0000 0001 | 0 0 0 0 0 0 |

$0.f \times 2^{exp}$

| | 0000 0000 | 00 - - 1 0 |

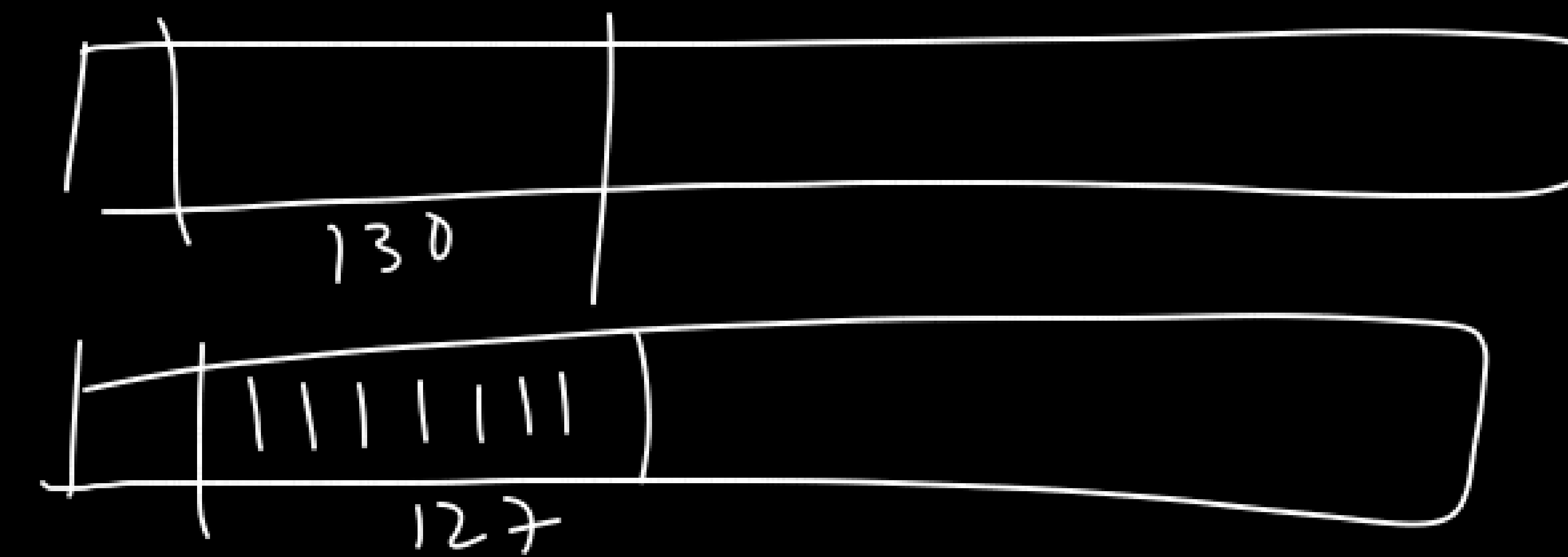
symbol power

0 | 0 - - 0 | 1 1 1 - - 1

if $exp = 0$ & $f \neq 0$, value $\pm 0.f \times 2^{exp}$

if $exp = 0$, $f = 0$, value $= 0$

Addition of floats



$$
\begin{array}{rl}
 & 1.11 \times 2^{3} \\
+ & 0.0011 \times 2^{3} \leftarrow 1.011 \times 2^{1} \\
\hline
\end{array}
$$

Step 1: Make exponents equal, make lower → larger

Calculate diff, $d = |exp1 - exp2|$

Step 2: $sig_2 >> d$

Step 3 → Add $Sig_1 + Sig_2$ (1.1111)

Step 4 → Round off

Step 5 Normalize × power

4.59

4.5

4.6

Round-off

i) Take closest possible.

$\frac{1}{\quad}$

			6
6	$5.56 \rightarrow$	5.6	5
5	$5.39 \rightarrow$	5.4	
	$\underline{5.46} \rightarrow$	5.5	6

In floating point o/ps

<u>Gurd - bits</u>

Floating point Multiplication

$$\begin{array}{r} 0.11 \\ 0.1 \\ \hline 1.1 \\ 1.0 \\ \hline 1.10 \end{array} \qquad 0.1 > 0.011$$

$$\begin{array}{r} 1.f_1 \times 2^{e_1} \\ 1.f_2 \times 2^{e_2} \\ \times \hline \end{array}$$

$127+f_1$

$S f_1$

$127+f_2$

$S f_2$

$f = Sf_1 + Sf_2 - 127$

Step 1 Add exponents $e_1 + e_2 = e$

Step 2: Multiply Sig 1 X Sig. 2

Step 3 → Normalization (Shift$^{\text{left}}$ Result & $e = e + \#\text{shift}$

Circuit for Int Multiplication

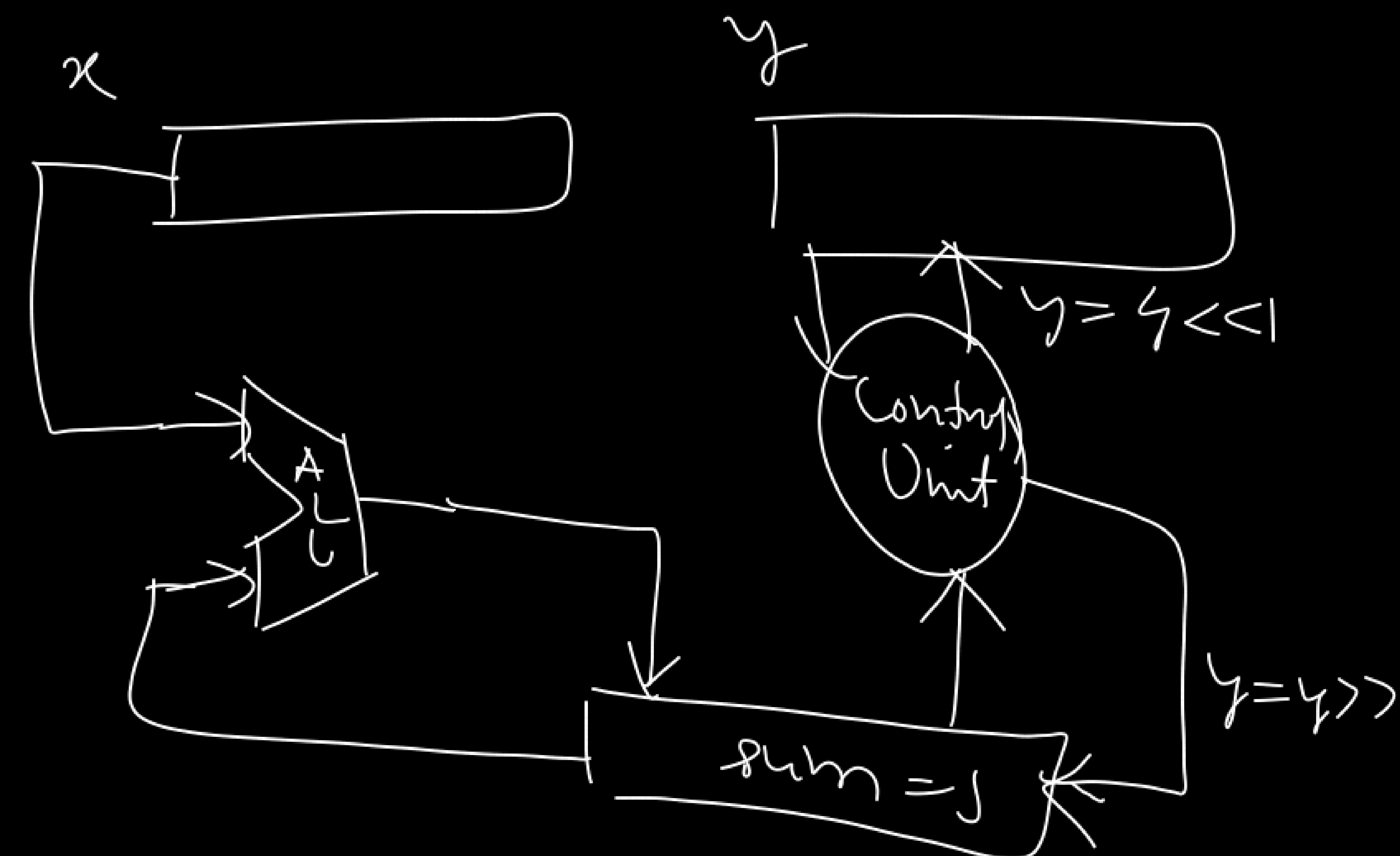
x $\quad$ y

$x_{n-1} \ldots x_0 \quad y_{n-1} \ldots y_0$

x · 0 0 1 0

1 1 0 1

```
s = 0
for i = 1 to 32
    if y[0] = 1
        s = s + x
    y = y >> 1
    x = x << 1
```




ALU