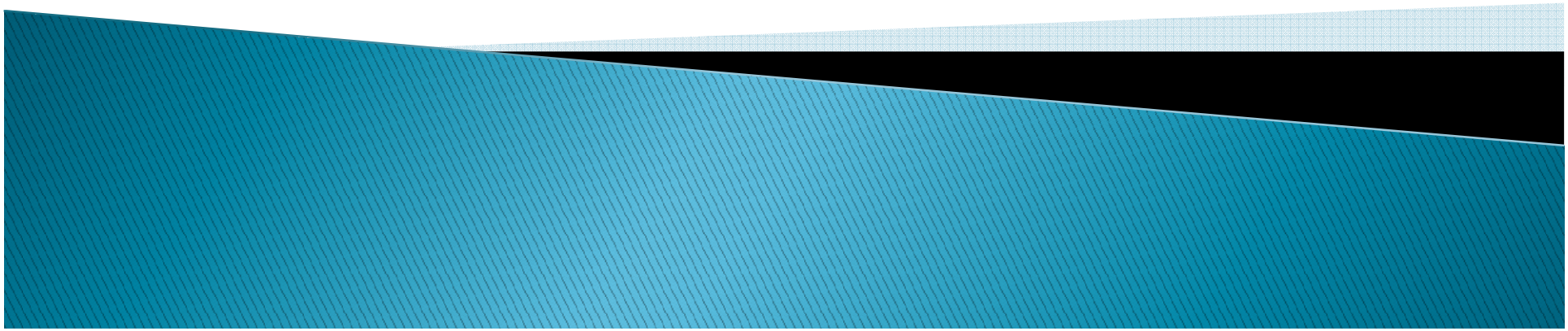
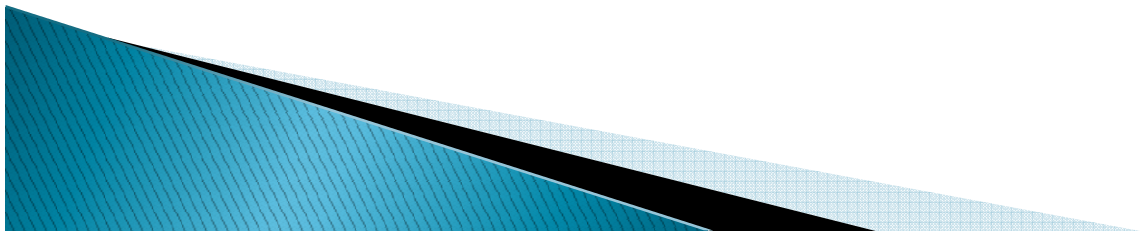
Arithmetic for Computers

Floating Point



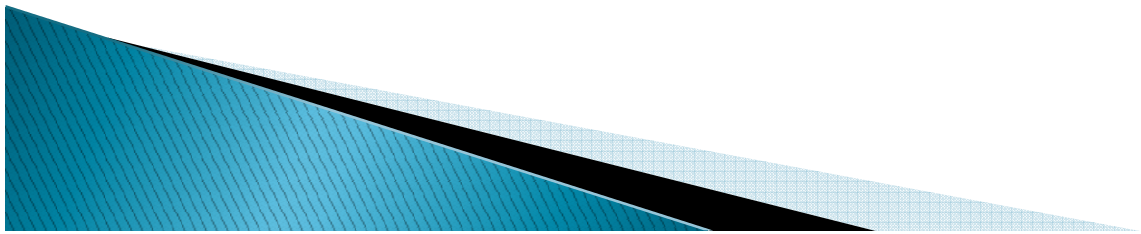
Floating Point

- Representation for non-integral numbers
  - very small
  - very large numbers
- Examples:
  - $4.25 * 10^{1}$
  - 10^{-3}
  - -3.35
  - $-1.42 * 10^{2}$



Floating Point

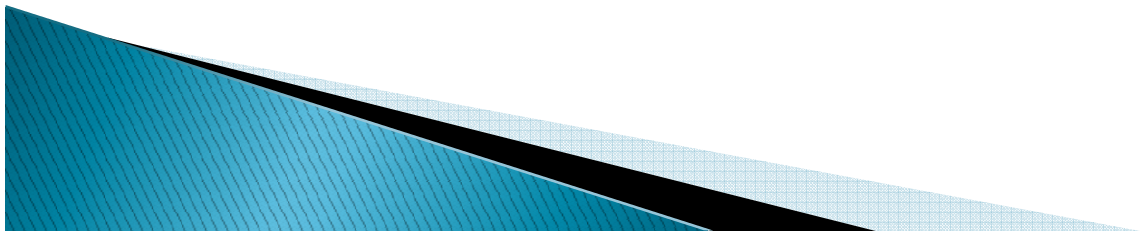
- Like scientific notation
  - Normalized (binary point is placed to the right of the first significant bit)
    - -2.34×10^{56}
  - Not Normalized
    - $+0.002 \times 10^{-4}$
    - $+987.02 \times 10^{9}$
- In binary
  - $\pm 1.xxxxxxx_2 \times 2^{yyyy}$



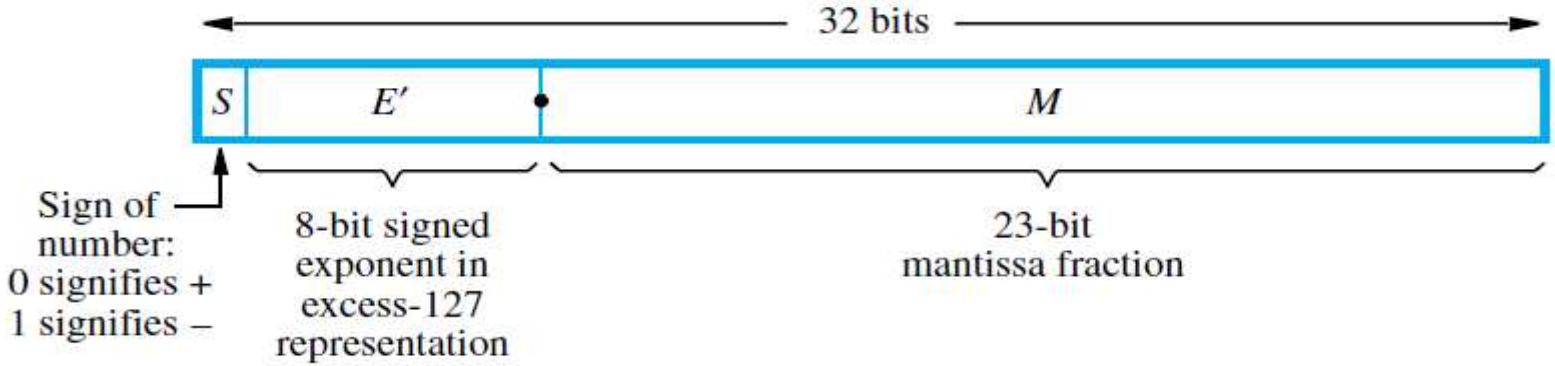
Floating Point Standard

- IEEE (Institute of Electrical and Electronics Engineers) Standard 754, labeled 754-2008

- Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)



Single precision (32-bit)



$$\text{Value represented} = \pm 1.M \times 2^{E'-127}$$

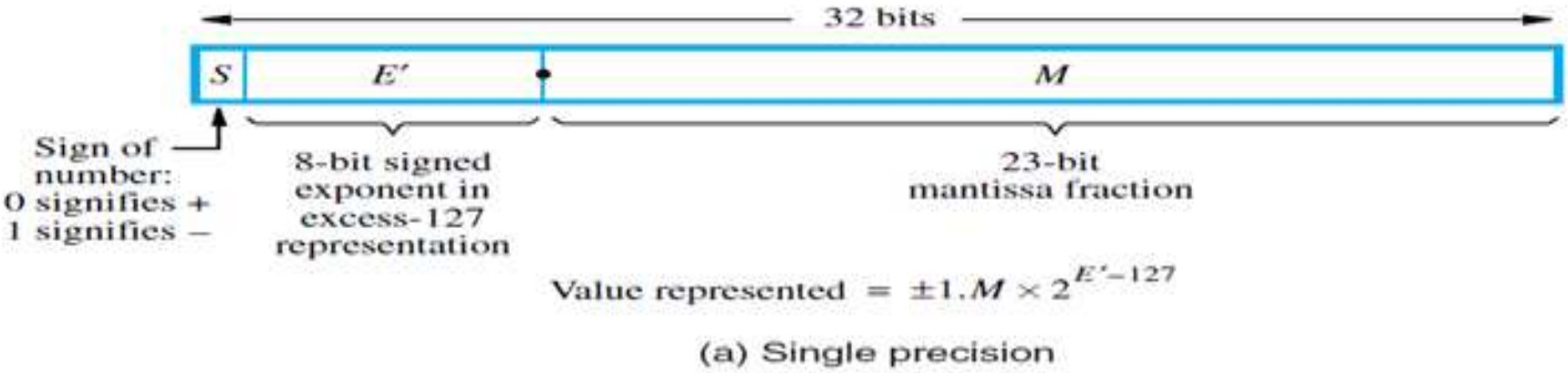
(a) Single precision

$$\text{Value represented} = 1.001010\ldots0 \times 2^{-87}$$

(b) Example of a single-precision number

- E`=E+127 = -87+127= 40
- 40 binary =101000

Single precision (32-bit)

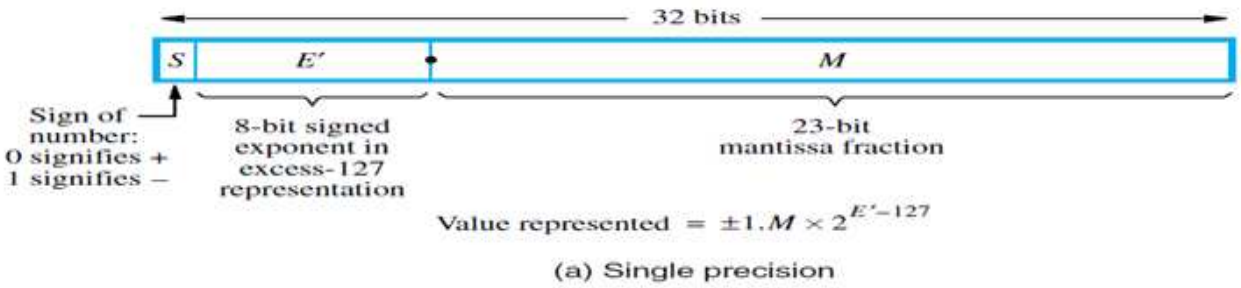


(a) Single precision

$$B = 1.M = 1.b_{-1}b_{-2}\ldots b_{-23}$$

$$V(B) = 1 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-23} \times 2^{-23}$$

Single precision (32-bit)



(a) Single precision

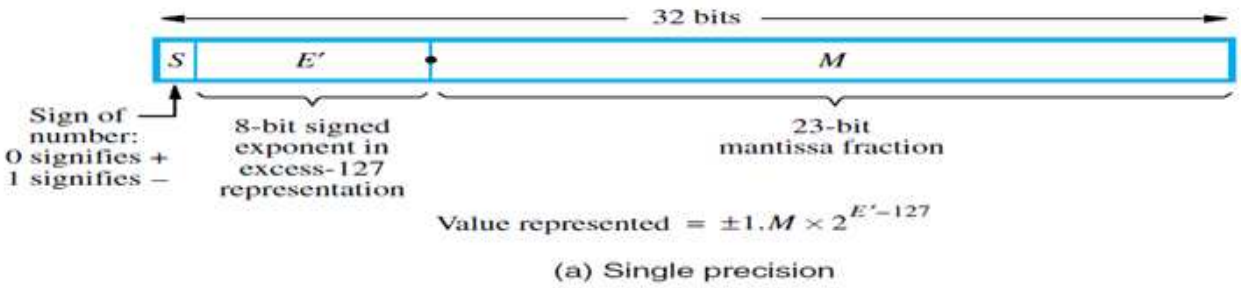
- *E ` = E + 127 Excess-127 format*

- *E` is in the* range 0 ≤ *E* ≤ *255*
- *E is in the range −127 ≤ E ≤ 128*
- *Valid range*
- *E` is in the* range 1 ≤ *E* ≤ *254*
- *E is in the range −126 ≤ E ≤ 127*

- *The end values of this range, 0 and 255, are used to represent special v*alues
- The range of *E` for normal values is 1 ≤ E ≤ 254*

- Less than −126 then underflow
- Greater than +127 then overflow

Single precision (32-bit)



(a) Single precision

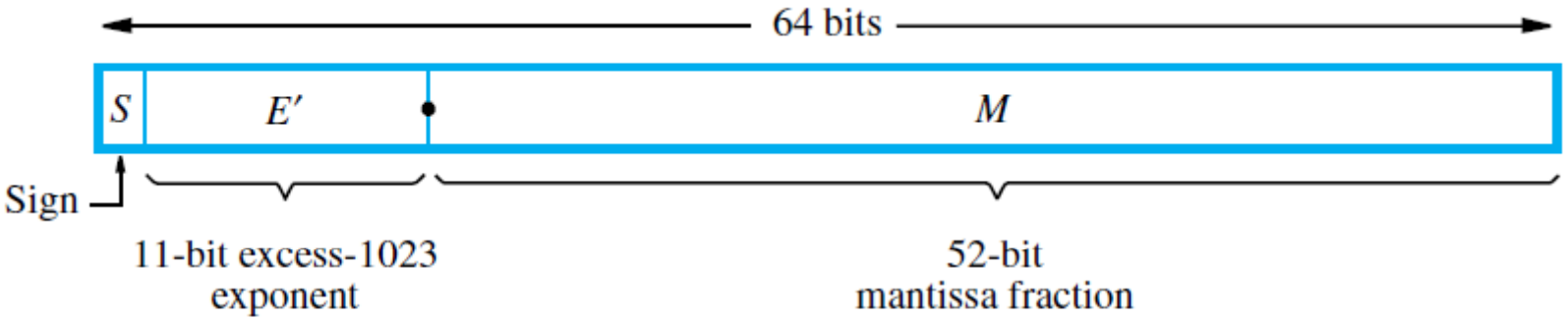
- *E` = 0 and M =0, the value 0 is represented*
- *E `= 255 and M = 0, the value infinite is represented*

- *E` = 0 and M ≠ 0, denormal* $\pm 0.M \times 2^{-126}$
- *E `= 255 and M ≠ 0, Not a Number (NaN) :* performing an invalid operation

<u>Exception</u>

- A processor must set *exception flags if any of* the following conditions arise
  - Underflow
  - overflow
  - divide by zero
  - inexact
  - invalid

Double precision (64-bit)



$$\text{Value represented} = \pm 1.M \times 2^{E'-1023}$$

(c) Double precision

IEEE Floating-Point Format

single: 8 bits excess 127
double: 11 bits excess 1023

single: 23 bits
double: 52 bits

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: sign bit
- Normalize significand: 1.Fraction
- Exponent:
  - E` = E+ Bias
  - Ensures exponent is unsigned

Floating-Point Example

- Represent -0.75
  - Sign
    - S = 1
  - Mantissa
    - 0.75*2 =1.50
    - 0.50*2 =1.00
    - $(0.75)_2 = 0.11 = 1.1 * 2^{-1}$ (normalized)
  - Fraction = $1000\ldots00_2$
  - Exponent = -1 + Bias
    - Single: -1 + 127 = 126 = 01111110_2
    - Double: -1 + 1023 = 1022 = 01111111110_2
- Single: 10111111101000…00
- Double: 10111111111101000…00

Floating-Point Example

- What number is represented by the single-precision float
  1100000001 01000…00
  - S = 1
  - Fxponent
    - E` = 10000001_2 = 129
    - E=E`-127= 129-127 =2
  - Fraction = $01000…00_2$
    - = $0*2^{-1}+1*2^{-2}$
    - = 0+1/4 = 0+0.25= 0.25
  - Mantisaa
    - 1.010000 = $1*2^0$ + 0.25 = 1.25

- $x = (-1)^1 \times 1.25 \times 2^2$
  $= -5.0$

Floating-Point operation

- $2.9400 \times 10^2 + 4.3100 \times 10^4$
- Represent 2.9400×10^2 as 0.0294×10^4
- $0.029400 \times 10^4 + 4.3100 \times 10^4 = 4.3394 \times 10^4$

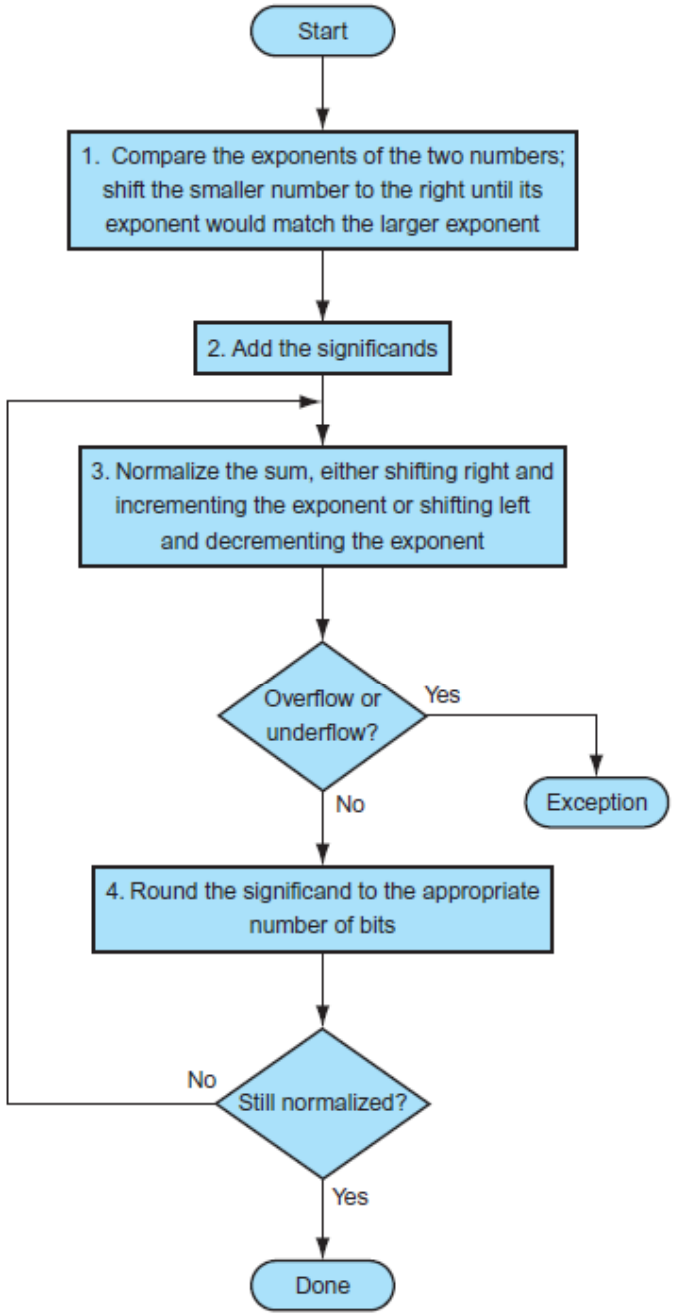
- **Add/Subtract Rule**

  1. Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
  2. Set the exponent of the result equal to the larger exponent.
  3. Perform addition/subtraction on the mantissas and determine the sign of the result.
  4. Normalize the resulting value, if necessary.

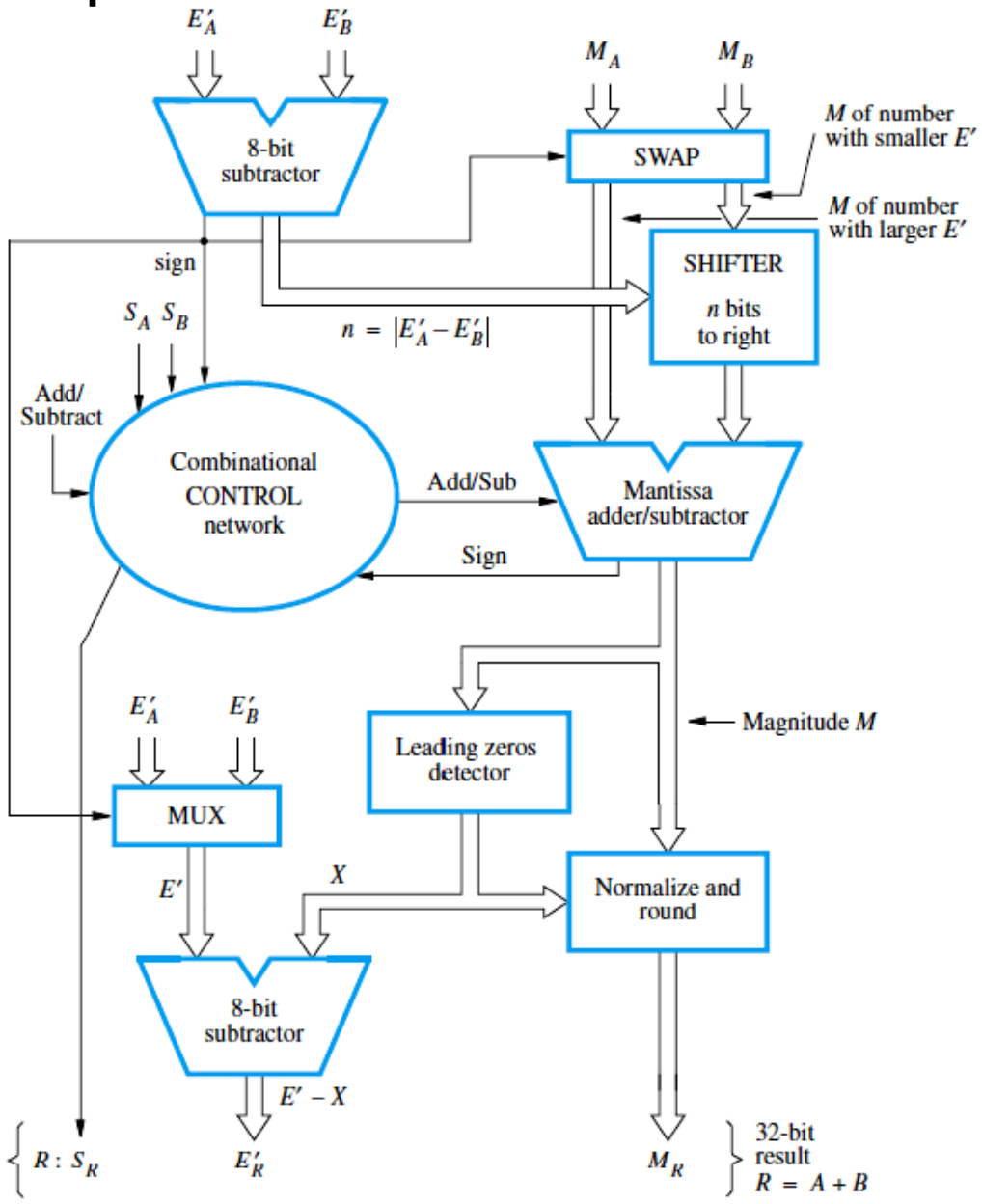
Implementing Floating-Point addition

- adding the numbers 0.5 and 0.4375

- $0.5 = 1.000 * 2^{-1}$
- $-0.4375 = -1.110 * 2^{-2}$
- Step1 : $-0.4375 = -1.110 * 2^{-2}$
  $= -0.111 * 2^{-1}$
- Step2 : $1.000+(-0.111) = 0.001$
- Step3 : Normalize $0.001 * 2^{-1}$
  $= 1.000 * 2^{-4}$



Implementing Floating-Point Operations



Floating-point addition-subtraction unit.

Floating-Point operation

- **Multiply Rule**

1. Add the exponents and subtract 127 to maintain the excess-127 representation.
2. Multiply the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

- **Divide Rule**

1. Subtract the exponents and add 127 to maintain the excess-127 representation.
2. Divide the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

FP Instructions in MIPS

- Separate FP registers
  - 32 single-precision: $f0, $f1, … $f31
  - Paired for double-precision: $f0/$f1, $f2/$f3, …

MIPS floating-point **assembly language**

Category	Instruction	Example	Meaning	Comments
Arithmetic	FP add single	add.s $f2,$f4,$f6	$f2 = $f4 + $f6	FP add (single precision)
	FP subtract single	sub.s $f2,$f4,$f6	$f2 = $f4 - $f6	FP sub (single precision)
	FP multiply single	mul.s $f2,$f4,$f6	$f2 = $f4 × $f6	FP multiply (single precision)
	FP divide single	div.s $f2,$f4,$f6	$f2 = $f4 / $f6	FP divide (single precision)
	FP add double	add.d $f2,$f4,$f6	$f2 = $f4 + $f6	FP add (double precision)
	FP subtract double	sub.d $f2,$f4,$f6	$f2 = $f4 - $f6	FP sub (double precision)
	FP multiply double	mul.d $f2,$f4,$f6	$f2 = $f4 × $f6	FP multiply (double precision)
	FP divide double	div.d $f2,$f4,$f6	$f2 = $f4 / $f6	FP divide (double precision)
Data transfer	load word copr. 1	lwc1 $f1,100($s2)	$f1 = Memory[$s2 + 100]	32-bit data to FP register
	store word copr. 1	swc1 $f1,100($s2)	Memory[$s2 + 100] = $f1	32-bit data to memory
Condi-tional branch	branch on FP true	bc1t 25	if (cond == 1) go to PC + 4 + 100	PC-relative branch if FP cond.
	branch on FP false	bc1f 25	if (cond == 0) go to PC + 4 + 100	PC-relative branch if not cond.
	FP compare single (eq,ne,lt,le,gt,ge)	c.lt.s $f2,$f4	if ($f2 < $f4) cond = 1; else cond = 0	FP compare less than single precision
	FP compare double (eq,ne,lt,le,gt,ge)	c.lt.d $f2,$f4	if ($f2 < $f4) cond = 1; else cond = 0	FP compare less than double precision

FP Example: °F to °C

- C code:

```
float f2c (float fahr) {
  return ((5.0/9.0)*(fahr - 32.0));
}
```

  - fahr in $f12, result in $f0, literals in global memory space

- Compiled MIPS code:

```
f2c: lwc1  $f16, const5($gp)  #f16=5
     lwc2  $f18, const9($gp)   #f18=9
     div.s $f16, $f16, $f18    #f16=f16/f18
     lwc1  $f18, const32($gp) #f18=32
     sub.s $f18, $f12, $f18     #f18=f12-f18
     mul.s $f0,  $f16, $f18     #f0=f16*f18
     jr    $ra
```

Accurate Arithmetic

- IEEE Std 754 specifies additional rounding control
  - Extra bits are maintained during intermediate steps for maximum accuracy in the final results

    : *Guard bits*
  - Guard bits has to be truncated to generate the final result
  - Choice of rounding modes
    - *Chopping :*

      $0.b_{-1}b_{-2}b_{-3}000$ to $0.b_{-1}b_{-2}b_{-3}111$ truncated to $0.b_{-1}b_{-2}b_{-3}$.

    - *Von Neumann rounding*

      $0.b_{-1}b_{-2}b_{-3}001$. to $0.b_{-1}b_{-2}b_{-3}111$ are truncated to $0.b_{-1}b_{-2}1$.

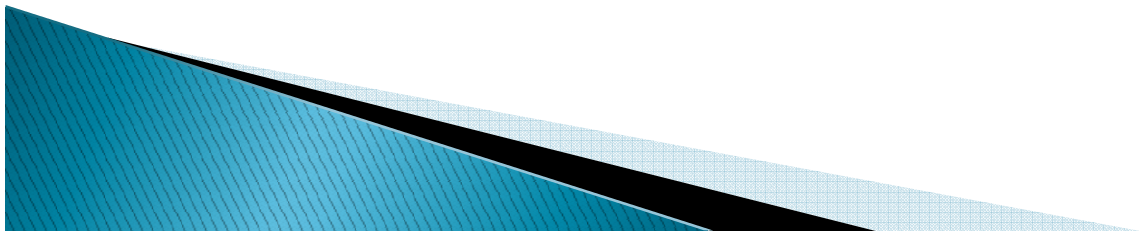
    - *Rounding*

      $0.b_{-1}b_{-2}b_{-3}1\ldots$ is rounded to $0.b_{-1}b_{-2}b_{-3} + \bar{0}.001$

      $0.b_{-1}b_{-2}b_{-3}0\ldots$ is rounded to $0.b_{-1}b_{-2}b_{-3}$

Subword parallelism

- A subword is a lower precision unit of data contained within a word.
- In subword parallelism, we pack multiple subwords into a word and then process whole words.
- With the appropriate subword boundaries, this technique results in parallel processing of subwords.
- Since the same instruction applies to all the subwords within the word, this is a form of small-scale SIMD (single-instruction. multiple-data) processing.'



Subword parallelism

- process data that are of the same size
  - For example, if the word size is 64 bits, some useful subword sizes are
    - 8, 16, and 32 bits
- Subword parallelism is an efficient and flexible solution for media processing,