Recitation 4

Computer Architecture (section 1)

Representing Signed Integers

11000000111001

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

What about -12345?

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

Sign magnitude encoding

Extra bit to denote sign

What about -12345?

111000000111001 $\longrightarrow$ -12345

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

What about -12345?

Sign magnitude encoding

Extra bit to denote sign

111000000111001 $\longrightarrow$ -12345

011000000111001 $\longrightarrow$ 12345

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

Sign magnitude encoding

Extra bit to denote sign

What about -12345?

111000000111001 → -12345

011000000111001 → 12345

100000000000000 → -0

000000000000000 → 0

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

Sign magnitude encoding

Extra bit to denote sign

What about -12345?

111000000111001 $\rightarrow$ -12345

011000000111001 $\rightarrow$ 12345

100000000000000 $\rightarrow$ -0

000000000000000 $\rightarrow$ 0

Can represent 2^n values, but really only 2^n-1 are usable

Representing Signed Integers

11000000111001

$2^{13}+2^{12}+2^{5}+2^{4}+2^{3}+2^{0}$

12345

Sign magnitude encoding

Extra bit to denote sign

What about -12345?

111000000111001 $\rightarrow$ -12345

011000000111001 $\rightarrow$ 12345

100000000000000 $\rightarrow$ -0

000000000000000 $\rightarrow$ 0

Can represent 2^n values, but really only 2^n-1 are usable

Range:

$[-2^{n-1}-1, 2^{n-1}-1]$

Representing Signed Integers

How to avoid the -0?

Representing Signed Integers

How to avoid the -0?

Idea: shift over all negative numbers

Now, -0 will become a -1

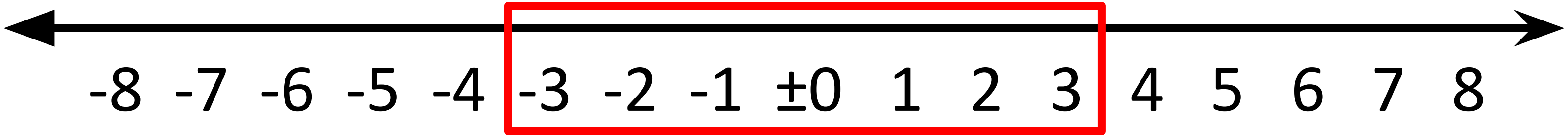
Representing Signed Integers

How to avoid the -0?

Idea: shift over all negative numbers

Now, -0 will become a -1

With 3 bits: *sign magnitude encoding*



Representing Signed Integers

How to avoid the -0?

Idea: shift over all negative numbers

Now, -0 will become a -1

With 3 bits: *sign magnitude encoding*

-8 -7 -6 -5 -4 -3 -2 -1 ±0 1 2 3 4 5 6 7 8

Now

-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8

Representing Signed Integers

Two's complement encoding

$$0110000000111001 \rightarrow 12345$$

Representing Signed Integers

Two's complement encoding

$0110000000111001 \longrightarrow 12345$

To invert sign: flip all bits and add 1

Representing Signed Integers

Two's complement encoding

0110000000111001 $\longrightarrow$ 12345

To invert sign: flip all bits and add 1

Still denotes sign

1001111111000110 + 1

Representing Signed Integers

Two's complement encoding

0011000000111001 → 12345

To invert sign: flip all bits and add 1

Still denotes sign

1100111111000110 + 1

1100111111000111 → -12345

Representing Signed Integers

Two's complement encoding

0011000000111001 ⟶ 12345

To invert sign: flip all bits and add 1

Still denotes sign

1100111111000110 + 1

1100111111000111 ⟶ -12345

Now -0 = 0

0000000000000000 ⟶ 0

1111111111111111 + 1

0000000000000000 ⟶ 0

Representing Signed Integers

Two's complement encoding

0110000000111001 → 12345

To invert sign: flip all bits and add 1

Still denotes sign

1001111111000110 + 1

1001111111000111 → -12345

Now -0 = 0

0000000000000000 → 0

1111111111111111 + 1

0000000000000000 → 0

Range:
$[-2^{n-1}, 2^{n-1}-1]$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits $\quad 0.0 \rightarrow 0/2^1$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \rightarrow 0/2^1$

$0.01 \rightarrow 1/2^2 = 1/4$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$$0.0 \longrightarrow 0/2^1$$

$$0.01 \longrightarrow 1/2^2 = 1/4$$

$$0.0011 \longrightarrow 1/2^3+1/2^4$$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \rightarrow 0/2^1$

$0.01 \rightarrow 1/2^2 = 1/4$

$0.0011 \rightarrow 1/2^3 + 1/2^4 = 1/8 + 1/16 = 3/16$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \longrightarrow 0/2^1$

$0.01 \longrightarrow 1/2^2 = 1/4$

$0.0011 \longrightarrow 1/2^3+1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \longrightarrow 1/2^3+1/2^4\ 1/2^6$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \rightarrow 0/2^1$

$0.01 \rightarrow 1/2^2 = 1/4$

$0.0011 \rightarrow 1/2^3+1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \rightarrow 1/2^3+1/2^4\ 1/2^6 = 1/8 + 1/16 + 1/64 = 13/64$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \longrightarrow 0/2^1$

$0.01 \longrightarrow 1/2^2 = 1/4$

$0.0011 \longrightarrow 1/2^3+1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \longrightarrow 1/2^3+1/2^4\ 1/2^6 = 1/8 + 1/16 + 1/64 = 13/64$

$101.011 \longrightarrow$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \rightarrow 0/2^1$

$0.01 \rightarrow 1/2^2 = 1/4$

$0.0011 \rightarrow 1/2^3 + 1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \rightarrow 1/2^3 + 1/2^4\ 1/2^6 = 1/8 + 1/16 + 1/64 = 13/64$

$101.011 \rightarrow 2^2 + 2^0$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \longrightarrow 0/2^1$

$0.01 \longrightarrow 1/2^2 = 1/4$

$0.0011 \longrightarrow 1/2^3+1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \longrightarrow 1/2^3+1/2^4\ 1/2^6 = 1/8 + 1/16 + 1/64 = 13/64$

$101.011 \longrightarrow 2^2+2^0+1/2^2+1/2^3$

Fractions in Binary

Binary to decimal: Each *set* bit behind the radix represents $1/2^i$

No set bits

$0.0 \longrightarrow 0/2^1$

$0.01 \longrightarrow 1/2^2 = 1/4$

$0.0011 \longrightarrow 1/2^3+1/2^4 = 1/8 + 1/16 = 3/16$

$0.0011010 \longrightarrow 1/2^3+1/2^4\ 1/2^6 = 1/8 + 1/16 + 1/64 = 13/64$

$101.011 \longrightarrow 2^2+2^0+1/2^2+1/2^3 = 4 + 1 + 1/4 + 1/8 = 5 + 3/8$

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

$2 \times 0.45 = 0.9$

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

$2 \times 0.45 = 0.9$

$2 \times 0.9 = 1.8$

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

2 x 0.6 = 1.2

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

2 x 0.6 = 1.2

2 x 0.2 = 0.4

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

2 x 0.6 = 1.2

2 x 0.2 = 0.4

2 x 0.4 = 0.8

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

$2 \times 0.45 = 0.9$

$2 \times 0.9 = 1.8$

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

$2 \times 0.2 = 0.4$

$2 \times 0.4 = 0.8$

$2 \times 0.8 = 1.6$

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

$2 \times 0.45 = 0.9$

$2 \times 0.9 = 1.8$

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

$2 \times 0.2 = 0.4$

$2 \times 0.4 = 0.8$

$2 \times 0.8 = 1.6$

Repeating pattern

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

2 x 0.6 = 1.2

2 x 0.2 = 0.4

2 x 0.4 = 0.8 **LSB**

2 x 0.8 = 1.6

Repeating pattern

Fractions in Binary

Decimal to binary: Keep multiplying fractional portion by 2

0.45

2 x 0.45 = 0.9

2 x 0.9 = 1.8

2 x 0.8 = 1.6

2 x 0.6 = 1.2

2 x 0.2 = 0.4

2 x 0.4 = 0.8

2 x 0.8 = 1.6

LSB

0.01(1100)

Repeating pattern

Fixed Point Representation

Radix always at same index

1101.1001

Fixed Point Representation

Radix always at same index

1101.1001

Problem: What if we need more or less precision?

1101.0000

0000.0101

Fixed Point Representation

Radix always at same index

1101.1001

Problem: What if we need more or less precision?

1101.0000 — Wasted

0000.0101 — Wasted

IEEE 754 Floating Point

A standard for floating point representation - unlike fixed point, the radix can now “float”.

IEEE 754 Floating Point

A standard for floating point representation - unlike fixed point, the radix can now “float”.

Single Precision: 32-bit

Double Precision: 64-bit

IEEE 754 Floating Point

A standard for floating point representation - unlike fixed point, the radix can now “float”.

Single Precision: 32-bit

Double Precision: 64-bit

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

Sign bit (S) **Exponent bits (E)** **Mantissa bits (M)**

IEEE 754 Floating Point

A standard for floating point representation - unlike fixed point, the radix can now "float".

Single Precision: 32-bit

Double Precision: 64-bit

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

Sign bit (S) **Exponent bits (E)** **Mantissa bits (M)**

Form: $(-1)^S \times M \times 2^E$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23	**0.8**
23 / 2 = 11 **R** $\boxed{1}$	$2 \times 0.8 = 1.6$
11 / 2 = 5 **R** 1	$2 \times 0.6 = 1.2$
5 / 2 = 2 **R** 1	$2 \times 0.2 = 0.4$
2 / 2 = 1 **R** 0	
1 / 2 = 0 **R** 1 **MSB**	

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

$2 \times 0.2 = 0.4$

$2 \times 0.4 = 0.8$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

$2 \times 0.2 = 0.4$

$2 \times 0.4 = 0.8$

$2 \times 0.8 = 1.6$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1
11 / 2 = 5 **R** 1
5 / 2 = 2 **R** 1
2 / 2 = 1 **R** 0
1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$
$2 \times 0.6 = 1.2$
$2 \times 0.2 = 0.4$
$2 \times 0.4 = 0.8$ **LSB**
$2 \times 0.8 = 1.6$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

$2 \times 0.8 = 1.6$

$2 \times 0.6 = 1.2$

$2 \times 0.2 = 0.4$

$2 \times 0.4 = 0.8$ **LSB**

$2 \times 0.8 = 1.6$

Fractional binary:

$10111.(1100) = 1.0111(1100) \times 2^4$

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1
11 / 2 = 5 **R** 1
5 / 2 = 2 **R** 1
2 / 2 = 1 **R** 0
1 / 2 = 0 **R** 1 **MSB**

0.8

2 ×0.8 = 1.6
2 ×0.6 = 1.2
2 ×0.2 = 0.4
2 ×0.4 = 0.8 **LSB**
2 ×0.8 = 1.6

Fractional binary:

$10111.(1100) = 1.0111(1100) \times 2^4$

Exponent: 4 + 7 (bias: $2^{k-1}-1$)

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** $\boxed{1}$

11 / 2 = 5 **R** $\boxed{1}$

5 / 2 = 2 **R** $\boxed{1}$

2 / 2 = 1 **R** $\boxed{0}$

1 / 2 = 0 **R** $\boxed{1}$ **MSB**

0.8

2 ×0.8 = $\boxed{1}$.6

2 ×0.6 = $\boxed{1}$.2

2 ×0.2 = $\boxed{0}$.4

2 ×0.4 = $\boxed{0}$.8 **LSB**

2 ×0.8 = 1.6

Fractional binary:

10111.(1100) = 1.0111(1100)×2^4

Exponent: 4 + 7 (bias: 2^{k-1}-1)

11 ⟶ 1011

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1
11 / 2 = 5 **R** 1
5 / 2 = 2 **R** 1
2 / 2 = 1 **R** 0
1 / 2 = 0 **R** 1 **MSB**

0.8

2 ×0.8 = 1 6
2 ×0.6 = 1 2
2 ×0.2 = 0 4
2 ×0.4 = 0 8 **LSB**
2 ×0.8 = 1.6

Fractional binary:

$10111.(1100) = 1.0111(1100) \times 2^4$

Exponent: 4 + 7 (bias: $2^{k-1}-1$)

$11 \longrightarrow 1011$

0 1011 011111001

Floating Point - Example

Spec: 9 bit FP with 4 bit exponent and 4 bit mantissa. Convert 23.8 to FP

23

23 / 2 = 11 **R** 1

11 / 2 = 5 **R** 1

5 / 2 = 2 **R** 1

2 / 2 = 1 **R** 0

1 / 2 = 0 **R** 1 **MSB**

0.8

2 ×0.8 = 1 6

2 ×0.6 = 1 2

2 ×0.2 = 0 4

2 ×0.4 = 0 8 **LSB**

2 ×0.8 = 1.6

Fractional binary:

10111.(1100) = 1.0111(1100)×2^4

Exponent: 4 + 7 (bias: 2^{k-1}-1)

11 ⟶ 1011

0 1011 011111001

0 1011 1000

Round to nearest, ties to even

Floating Point - Modes

- Normal: For most fractions

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0's or 1's
    - E = e + bias (2^{k-1}-1)

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0’s or 1’s
    - E = e + bias (2^{k-1}-1)
  - Mantissa is a fractional binary with a “1” prefix in decimal form

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0's or 1's
    - $E = e + \text{bias } (2^{k-1}-1)$
  - Mantissa is a fractional binary with a "1" prefix in decimal form
- Denormal values: Very small values near 0

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0's or 1's
    - E = e + bias ($2^{k-1}-1$)
  - Mantissa is a fractional binary with a "1" prefix in decimal form
- Denormal values: Very small values near 0
  - Exponent field is all 0's
    - E = 1 - bias

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0's or 1's
    - E = e + bias ($2^{k-1}-1$)
  - Mantissa is a fractional binary with a "1" prefix in decimal form
- Denormal values: Very small values near 0
  - Exponent field is all 0's
    - E = 1 - bias
  - Mantissa is a fractional binary, but without "1" prefix
    - Possible to represent 0, but also -0

Floating Point - Modes

- Normal: For most fractions
  - Exponent field not all 0’s or 1’s
    - E = e + bias (2^{k-1}-1)
  - Mantissa is a fractional binary with a “1” prefix in decimal form
- Denormal values: Very small values near 0
  - Exponent field is all 0’s
    - E = 1 - bias
  - Mantissa is a fractional binary, but without “1” prefix
    - Possible to represent 0, but also -0
- Special values
  - Exponent field all 1’s
  - Mantissa: All 0 $\longrightarrow$ $\pm\infty$, non-zero $\longrightarrow$ NaN