ECE 486/586
Computer Architecture

Prof. Mark G. Faust

Maseeh College of Engineering
and Computer Science

PORTLAND STATE
UNIVERSITY

Outline

- Topics
  - Binary Arithmetic (Unsigned binary operands)
  - Signed number representations
    - Signed Magnitude
    - Diminished Radix Complement (Ones Complement)
    - Radix Complement (Twos Complement)
  - Binary Arithmetic Revisited and Overflow
  - Sign Extension
  - IEEE 754 Floating Point

Binary Arithmetic

Rules for "carry" same as in decimal

$$\begin{array}{r} 0 \\ +\ \underline{0} \\ 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ \underline{1} \\ 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ \underline{0} \\ 1 \end{array} \qquad \begin{array}{r} {}^{1}\ \ \ \\ 1 \\ +\ \underline{\ \ 1} \\ 10 \end{array}$$

carry

Unsigned Binary (UB) Addition

- Examples (4-bit word)

2 + 6

$$
\begin{array}{rr}
 & {\scriptstyle 1\,1} \\
 & 0010 \\
+ & 0110 \\
\hline
 & 1000
\end{array}
\qquad
\begin{array}{r}
2 \\
+\ 6 \\
\hline
8
\end{array}
$$

11 + 6

$$
\begin{array}{rr}
 & {\scriptstyle 1\ \ 1\,1} \\
 & 1011 \\
+ & 0110 \\
\hline
 & 10001
\end{array}
\qquad
\begin{array}{r}
11 \\
+\ \ 6 \\
\hline
17
\end{array}
$$

Overflow

$17 > 2^4-1$

Overflow (Underflow)

Definition:

Result of an arithmetic operation is too large (or small) to be represented in number of bits available

Detection:

Varies with the representation. For unsigned binary, it’s determined by a carry out of the MSB

$$\begin{array}{rr} & 1\ \ 1\ 1\quad\ 1\ 1\ \ \ \ \ \\ & 10101101 \\ + & \underline{\ 01101100} \\ & 100011001 \end{array}$$

8 bit operands

Signed Numbers

- Have been assuming non-negative numbers
  - Unsigned Binary (UB)
- Several representations for signed numbers
  - Sign Magnitude (SM)
  - Diminished Radix Complement (DRC)
    - 1s Complement
  - Radix Complement (RC)
    - 2s Complement

Sign Magnitude

MSB functions as sign bit

0 = positive

1 = negative

Range of numbers:

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

n = 6

Two representations for 0

Negative formed by complementing sign bit

$15_{10} = 001111_{SM}$

$-15_{10} = 101111_{SM}$

$010100_{SM} = 20_{10}$

$110100_{SM} = -20_{10}$

SM number						Decimal number
d_5	d_4	d_3	d_2	d_1	d_0	
0	1	1	1	1	1	$+31 = +(2^5 - 1)$
0	1	1	1	1	0	+30
0	1	1	1	0	1	+29
0	1	1	1	0	0	+28
		⋮				
0	0	0	0	1	0	+2
0	0	0	0	0	1	+1
0	0	0	0	0	0	+0
1	0	0	0	0	0	−0
1	0	0	0	0	1	−1
1	0	0	0	1	0	−2
		⋮				
1	1	1	1	0	0	−28
1	1	1	1	0	1	−29
1	1	1	1	1	0	−30
1	1	1	1	1	1	$-31 = -(2^5 - 1)$

Diminished Radix Complement (DRC)

Called "Ones Complement"

MSB indicates sign
0 = positive
1 = negative
(but not a sign bit!)

Range of numbers:
$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

Two representations for 0

Negative formed by complementing entire word
(called "taking the ones complement")

$15_{10} = 001111_{DRC}$
$-15_{10} = 110000_{DRC}$

$010100_{DRC} = 20_{10}$
$101011_{DRC} = -20_{10}$

DRC number						Decimal number
d_5	d_4	d_3	d_2	d_1	d_0	
0	1	1	1	1	1	$+31 = +(2^5 - 1)$
0	1	1	1	1	0	+30
0	1	1	1	0	1	+29
0	1	1	1	0	0	+28
		⋮				
0	0	0	0	1	0	+2
0	0	0	0	0	1	+1
0	0	0	0	0	0	+0
1	1	1	1	1	1	–0
1	1	1	1	1	0	–1
1	1	1	1	0	1	–2
		⋮				
1	0	0	0	1	1	–28
1	0	0	0	1	0	–29
1	0	0	0	0	1	–30
1	0	0	0	0	0	$-31 = -(2^5 - 1)$

Radix Complement (RC)

Called “Twos Complement”

MSB indicates sign
0 = positive
1 = negative
(but not a sign bit)

Range of numbers:
$-(2^{n-1})$ to $+(2^{n-1} - 1)$

Only one representation for 0

Negative formed by complementing entire word and adding 1 (called “taking the twos complement”)

$15_{10} = 001111_{RC}$
$-15_{10} = 110001_{RC}$

$010100_{RC} = 20_{10}$
$101100_{RC} = -20_{10}$

RC number						Decimal number
d_5	d_4	d_3	d_2	d_1	d_0	
0	1	1	1	1	1	$+31 = +(2^5 - 1)$
0	1	1	1	1	0	+30
0	1	1	1	0	1	+29
0	1	1	1	0	0	+28
		⋮				
0	0	0	0	1	0	+2
0	0	0	0	0	1	+1
0	0	0	0	0	0	+0
0	0	0	0	0	0	–0
1	1	1	1	1	1	–1
1	1	1	1	1	0	–2
		⋮				
1	0	0	1	0	0	–28
1	0	0	0	1	1	–29
1	0	0	0	1	0	–30
1	0	0	0	0	1	–31
1	0	0	0	0	0	$-32 = -2^5$

Twos Complement – Special Cases

Example: Take Twos Complement (RC) of 0000 (0_{10})

$$
\begin{array}{r}
0000 \\
{\scriptstyle 1\;1\;1\;\;} \\
1111 \\
+\quad 1 \\
\hline
0000
\end{array}
$$

Complement

Add One

Ignore carry out of MSB

Twos Complement – Special Cases

Example: Take Twos Complement (RC) of 1000 (-8_{10}) (most negative)

$$
\begin{array}{r}
1000 \\
\scriptstyle 1\ 1\ 1 \\
0111 \\
\underline{+\quad 1} \\
1000
\end{array}
$$

Complement

Add One

Can’t represent abs(most negative number)

Conversions of Signed Representations

- From decimal
  - Represent the absolute value of the number in UB
  - Use the correct number of bits (add leading 0s)
  - If the decimal number is negative, use the appropriate rule to negate the representation
    - S/M – complement the sign bit
    - DRC (ones complement) – complement every bit
    - RC (twos complement) – complement every bit, add 1
- To decimal
  - If number is +, convert from UB to decimal (done!)
  - If number is - use appropriate rule to negate it (obtain its absolute value)
    - S/M – complement the sign bit
    - DRC (ones complement) – complement every bit
    - RC (twos complement) – complement every bit, add 1
  - Convert this (positive) number as though UB to decimal
  - Add a negative sign

Why do we use RC (Twos Complement)?

- Only one representation for zero
- Simplified Addition
  - Sign Magnitude Addition
    - Must consider two operands without sign bits
    - If sign bits same: perform add, check overflow
    - If sign bits different: subtract (two cases)
      - +A and – B → A – B
      - – A and + B→ B – A
    - Generate correct sign bit for sum
  - Radix Complement (Twos Complement) Addition
    - Just add!
- Simple Subtraction
  - Done via addition
    - A + B
    - A – B = A + (-B)
    - Caution: Can't take negative of most negative number

Binary Arithmetic

- Unsigned Binary (UB)
- Signed Binary (SB)
- Diminished Radix Complement (DRC) ← Rarely used
  - 1s complement
- Radix Complement (RC)
  - 2s complement

Radix Complement (RC) or Twos Complement Addition

- Examples (4-bit word)

7 + (-2)

```
  1 1 1
    0111        7
+   1110     + (-2)
   -----     ------
   10101        5
```

5 + (-7)

```
      1
    0101        5
+   1001     + (-7)
    ----     ------
    1110      (-2)
```

Ignore carry out of MSB

Radix Complement (RC) or Twos Complement Addition

- Examples (4-bit word)

-7 + (-2)

```
    1
    1001      -7
+   1110   + (-2)
   -----   ------
   10111      -9
```

Ignore carry out of MSB

We added two negative numbers and got a positive result!

Overflow!!

$-9 < -(2^3)$

Carry and Overflow

Is this an overflow condition?

```
      1 1  1 0 1 1  0 0
         10101101
+        01101100
        ---------
        100011001
```

If this is an unsigned binary (UB) number – Yes!

Carry and Overflow

Is this an overflow condition?

```
   1 1 1 0 1 1 0 0
    10101101
+   01101100
  ----------
  100011001
```

If this is a radix complement (RC) number – No!

A negative number plus a positive number cannot produce an overflow

Carry and Overflow
Radix Complement (2s Complement)

- Consider:
  - Positive + Negative → Never overflow
  - Negative + Negative → Overflow possible
  - Positive + Positive → Overflow possible
- Detecting Overflow:
  - If signs of addends are same and sign of sum is different
  - Equivalent to $C_{SBP} \neq C_{SBP+1}$

Carry and Overflow

C_{SBP+1} C_{SBP}

C – Carry
SBP – Sign Bit Position

$$
\begin{array}{r}
1\,1\,1\,0\,1\,1\,0\,0 \\
10101101 \\
+\quad \underline{01101100} \\
100011001
\end{array}
$$

Radix Complement (RC) Addition

- Examples (4-bit word)

$C_{SBP+1} \neq C_{SBP}$

-7 + (-2)

C_{SBP+1} C_{SBP}

$$
\begin{array}{rrr}
 & 1\,0\,0\,0 & \\
 & 1001 & -7 \\
+ & \underline{\ 1110} & \underline{+\,(-2)} \\
 & 10111 & -9
\end{array}
$$

We added two negative numbers and got a positive result!

Overflow!

$-9 < -(2^3)$

Sign Extension

What happens when you move a number from a smaller word size to a larger one?

UB	S/M	RC (2s complement)
00000011 (3)	**00000011** (3)	**00000011** (3)
0000000000000011 (3)	**0000000000000011** (3)	**0000000000000011** (3)
11111101 (253)	**11111101** (-125)	**11111101** (-3)
0000000011111101 (253)	**1000000001111101** (-125)	**0000000011111101** (+253!)
		11111101 (-3)
		1111111111111101 (-3)

“extend” the sign bit left through the new MSB

Floating Point

- Need to represent "real" numbers
- Fixed point too restrictive for precision and range
- Not unlike familiar "scientific notation"

$$+6.022 \times 10^{23}$$

Sign Mantissa Exponent

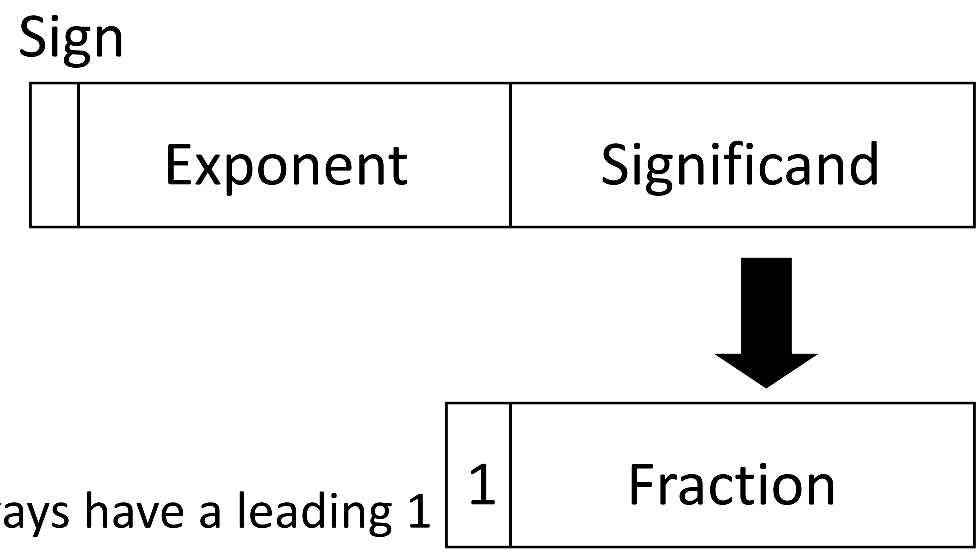
Normalized mantissa – single digit to left of decimal point

IEEE 754 Floating Point Standard

Binary

$+1.101011 \times 2^{12}$

Sign Mantissa Exponent



A normalized (binary) mantissa will always have a leading 1 so we can assume it and get an extra bit of precision instead

Exponents are stored with a bias which is added to the exponent before being stored (allows fast magnitude compare)

Universally used on virtually all computers

Several levels of precision/range: Single, Double, Double-Extended

IEEE 754 Floating Point Standard

- Single Precision – 32 bits, Bias = 127

$$F = -1^{\text{Sign}} \text{ x } (1+\text{Significand}) \text{ x } 2^{(\text{Exponent-Bias})}$$

IEEE 754 Floating Point Standard

- Example $F = -1^{Sign} \times (1+\text{Significand}) \times 2^{(\text{Exponent-Bias})}$

1	8	23
1	10000001	01000000000000000000000
31	30 … 23	22 … 0
-	129	$= 0.25_{10}$

$= -\ 1.01_2 \times 2^{(129-127)}$

$= -\ 1.25 \times 2^{(129-127)}$

$= -\ 1.25 \times 2^2$

$= -\ 1.25 \times 4$

$= -\ 5.0$

$= -\ 1.01_2 \times 2^{(129-127)}$

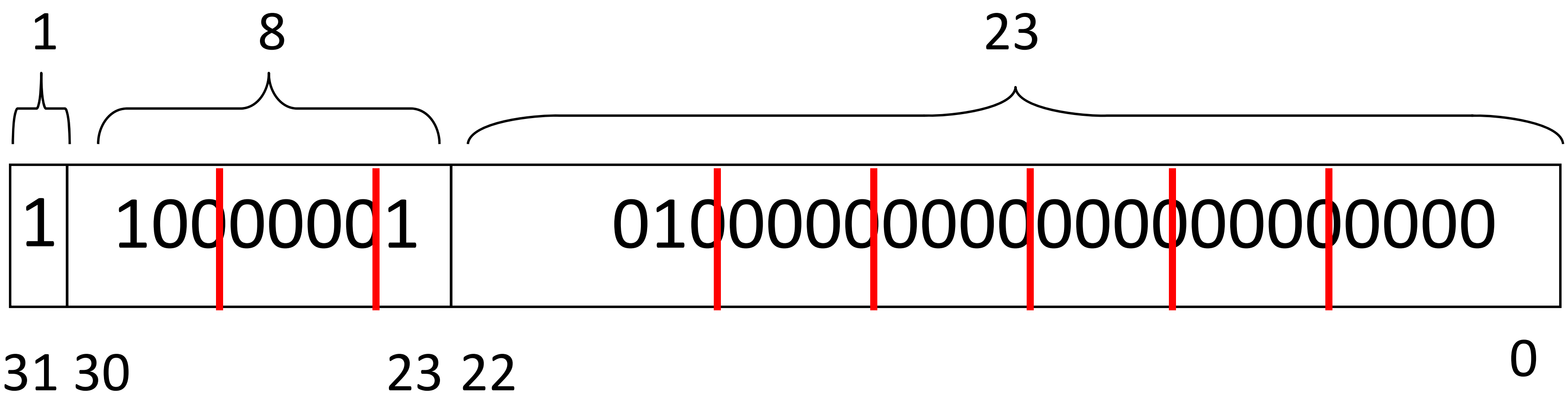
$= -\ 1.01_2 \times 2^2$

$= -\ 101_2$

$= -\ 5.0$

IEEE 754 Floating Point Standard

- Will commonly see these expressed as hex



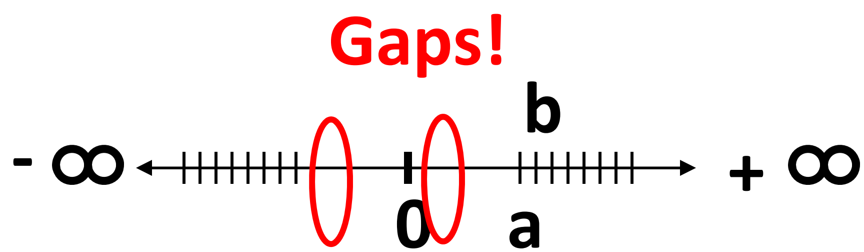
C0A00000

IEEE 754 Floating Point Standard

- Some Special Cases
  - Zero (no assumed leading 1)
    - Exponent = 0, Significand = 0
  - NaN (Not a Number), e.g. ∞
    - Exponent = 255
  - Denormalized numbers
    - Exponent = 0, Significand $\neq$ 0

IEEE 754 Floating Point Standard

- Why do we need denormalized numbers?
  - Addresses gap caused by implicit leading 1
  - Smallest positive number (a) is $1.000\ldots000 \times 2^{-126}$
  - Next number (b) is $1.000\ldots001 \times 2^{-126} = (2^{-126} + 2^{-149})$



S	Exponent	Significand
1	8	23

0	**00000000**	**00000000000000000000000**	**= 0**
0	**00000000**	**00000000000000000000001**	**= $1.00000000000000000000001 \times 2^{-127}$**
0	**00000000**	**00000000000000000000010**	**= $1.00000000000000000000010 \times 2^{-127}$**
0	**00000000**	**00000000000000000000011**	**= $1.00000000000000000000011 \times 2^{-127}$**
0	**00000000**	**00000000000000000000100**	**= $1.00000000000000000000100 \times 2^{-127}$**

Distance from zero to smallest positive number is $1.00000000000000000000001 \times 2^{-127} \sim= 2^{-127}$

Distance to next number is $0.00000000000000000000001 \times 2^{-127} = 2^{-23} \times 2^{-127} = 2^{-150}$

IEEE 754 Floating Point Standard

- Denormalized Numbers
  - Solution – Non-normalized form
  - Exponent = 0, Significand $\neq$ 0
  - Implicit Exponent of -126
  - $F = -1^{Sign} \times (\text{Significand}) \times 2^{(-126)}$
  - Smallest positive number (a) = $0.000\ldots001 \times 2^{-126} = 2^{-149}$
  - Next smallest number (b) = $0.000\ldots010 \times 2^{-126} = 2^{-148}$