ICS 233
COMPUTER ARCHITECTURE

Computer Arithmetic

Lecture 15

Lecture Outline

- Real Number Representation
- Floating-point Arithmetic

Real Number Representation

- **Fixed-point Representation**
- **Floating-point Representation**

Real Number Representation

- **Fixed-point Representation**

- In this representation, a fixed binary point is assumed
- This format allows the representation of numbers with a fractional part
- Limitations
  - Very large number cannot be represented, nor can very small fractions
  - The fractional part of the quotient in a division of two large numbers could be lost

Real Number Representation

- **Floating-point Representation**

- Similar to scientific notation
- The binary point is dynamically moved to a convenient location and the exponent is used to keep track of that binary point
- This allows a range of very large and very small numbers to be represented with only a few bits.

Example : 1000.11001×2^6

Exponent

Mantissa
(Significand)

IEEE Standard for Floating point Representation

- **IEEE Standard 754**

- Floating point representation is defined in IEEE Standard 754
- IEEE Standard defines

  - a 32-bit Single precision format

  - a 64-bit Double precision format

IEEE 754 Single Precision format

- Single precision representation uses a 32-bit format
- The format consists of three fields
  - **Sign bit field – 1 bit Most significant bit used to indicate the sign of the mantissa**
  - **Biased Exponent field – next 8 bits used to store the biased exponent (true exponent + bias of value 127)**
  - **Mantissa field – last 23 bits used to store the fractional mantissa**
    - **Mantissa is expressed in Normalized form**
    - **The first bit of the mantissa is always 1 and need not be stored in the mantissa field**

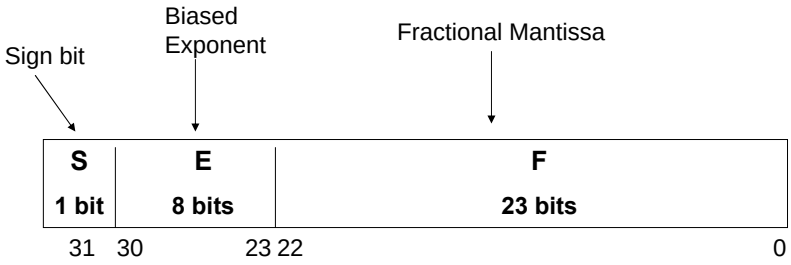
Example : $1000.11001 \times 2^6 = 1.00011001 \times 2^9$

after Normalization

Fractional mantissa

IEEE 754 Single Precision format

- Single precision representation uses a 32-bit format



Sign Bit = 1, negative mantissa

= 0, positive mantissa

$$N = (-1)^S \ 1.F \times 2^{E-B}$$

Where B - bias

IEEE 754 Single Precision format

- Single precision format parameters

Word width	32 bits
Exponent width	8bits
Mantissa width	23 bits
Exponent Bias	127
Maximum Exponent	127
Minimum Exponent	-126
Number of exponents	254
Number of fractions	2^{23}
Number range	10^{-38} to 10^{+38}

IEEE 754 Single Precision format

- Express the following in Single precision format :

- N = **$1100.110110 \times 2^{10}$**

After Normalization,

$N = 1.100110110 \times 2^{13}$

S =0, F = 100110110

True exponent, e = 13

E = e + B = 13 + 127 = 140 = 10001100

In single precision form :

N = 0 10001100 10011011000000000000000

= 464D8000 H (packed form)

IEEE 754 Single Precision format

❑ Express the following in Single precision format :

➢ **N = - 0.000100110110 x 2^{-22}**

After Normalization,

N = - 1.00110110 x 2^{-26}

S =1, F = 00110110

True exponent, e = -26

E = e + B = -26 + 127 = 101 = 01100101

In single precision form :

N = 1 01100101 00110110000000000000000

= B29B0000 H (packed form)

IEEE 754 Double Precision format

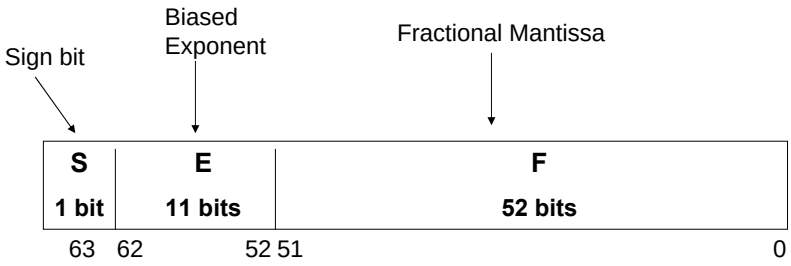
➢ Double precision representation uses a 64-bit format

➢ The format consists of three fields

- **Sign bit field – 1 bit Most significant bit used to indicate the sign of the mantissa**
- **Biased Exponent field – next 11 bits used to store the biased exponent (true exponent + bias of value 1023)**
- **Mantissa field – last 52 bits used to store the fractional mantissa**
  - **Mantissa is expressed in Normalized form**
  - **The first bit of the mantissa is always 1 and need not be stored in the mantissa field**

IEEE 754 Double Precision format

- Double precision representation uses a 64-bit format



Sign Bit = 1, negative mantissa

= 0, positive mantissa

$$N = (-1)^S \ 1.F \times 2^{E-B}$$

Where B - bias

IEEE 754 Double Precision format

- Double precision format parameters

Word width	64 bits
Exponent width	11 bits
Mantissa width	52 bits
Exponent Bias	1023
Maximum Exponent	1023
Minimum Exponent	-1022
Number of exponents	2046
Number of fractions	2^{52}
Number range	10^{-308} to 10^{+308}

IEEE Standard for Floating point Representation

- **ZERO**

- An exponent of zero together with a fraction of zero represents positive or negative zero, depending on the sign bit

  0 00000000 00000000000000000000000 = +0.0
  1 00000000 00000000000000000000000 = -0.0

- **INFINITY**

- An exponent of all ones together with a fraction of zero represents positive or negative infinity, depending on the sign bit.

  0 11111111 00000000000000000000000 = + ∞
  1 11111111 00000000000000000000000 = - ∞

IEEE Standard for Floating point Representation

- **Denormalized Number**

- An exponent of zero together with a nonzero fraction represents denormalized number. In this case, the bit to the left of the binary point is zero (0.F) and the bias is 126 (single precision) or 1022 (double precision) i.e., true exponent is E-126 or E-1022. The number is positive or negative depending on the sign bit

  0 00000000 10000000000000000000000 = $+0.1 \times 2^{-126}$
  1 00000000 10000000000000000000000 = -0.1×2^{-126}

- **Not A Number (NaN)**

- An exponent of all ones together with a nonzero fraction NaN, which means not a number and is used to signal various exception conditions. .

  0 11111111 10000000000000000000000 = NaN
  1 11111111 10000000000000000000000 = NaN

Floating Point Arithmetic

- **Addition**
- **Subtraction**
- **Multiplication**
- **Division**

- **Exponent Overflow**
  - Occurs when a positive exponent greater than the maximum possible positive exponent value is obtained

- **Exponent Underflow**
  - Occurs when a negative exponent smaller than the minimum possible negative exponent value is obtained

ICS 233
COMPUTER ARCHITECTURE

Floating point Addition

Floating Point Addition

- **Given two Floating point number N1 and N2(in packed form). Perform N1 + N2 and store the result in packed floating point packed form**
- **Step 1 :**
  Check whether one of the numbers is zero,
  if yes then output nonzero number as the result else goto step 2
- **Step 2 :**
  Unpack the floating point packed numbers N1 and N2 to retrieve sign of mantissas(S1, S2) , biased exponents(E1, E2) and mantissas (m1, m2)
- **Step 3 :**
  Make the exponents E1 and E2 equal.
  i.e., Shift the smaller number mantissa to the right by | E1 – E2 | bit positions and update the exponent of smaller number to make it equal to that of the bigger number.
- **Step 4:**
  Perform addition of mantissas, i.e., m = m1 + m2
- **Step 5 :**
  Determine the sign of the result (S), fractional mantissa (F) and biased exponent (E) of the result
- **Step 6 :**
  Express the result in floating point packed form.

Floating Point Addition

Question :

a) Convert the following decimal numbers into the IEEE format for single precision floating point numbers and express each in packed form.

N1 = 23.895 N2 = 39.525

b) Perform N1 + N2 using the IEEE format binary floating point numbers obtained in part a) of the question. You should begin the calculation with the packed floating point representations of these numbers and end with the packed result.

Floating Point Addition

Solution :

a) N1 = 23.895
= 10111.11100101000111101 01

Normalizing,
N1 = 1.01111110010100011110101 x 2^4

S1 = 0, F1 = 01111110010100011110101, e1 = 4
E1 = e1 + 127 = 4 + 127 = 131 = 10000011

In packed form,
N1 = 01000001101111110010100011110101
= 41BF28F5 H

Floating Point Addition

Solution :

a) N2 = 39.525
= 100111.100001100110011001

Normalizing,
N2 = 1.00111100001100110011001 x 2^5

S2 = 0, F2 = 00111100001100110011001 , e2 = 5
E2 = e2 + 127 = 5 + 127 = 132 = 10000100

In packed form,
N2 = 01000010000111100001100110011001
= 421E1999 H

Floating Point Addition

b) Perform N1 + N2

- **Step 2 :**
  Unpack the floating point packed numbers N1 and N2 to retrieve sign of mantissas(S1, S2) , biased exponents(E1, E2) and mantissas (m1, m2)

- N1 = 41BF28F5 H
  S1 = 0, F1 = 01111110010100011110101
  m1 = 1. 01111110010100011110101
  E1 = 131

  N2 = 421E1999 H
  S2 = 0, F2 = 00111100001100110011001
  m2 = 1. 00111100001100110011001
  E2 = 132

Floating Point Addition

b) Perform N1 + N2

- **Step 3 :**
  Make the exponents E1 and E2 equal.
  Since E2 > E1, Shift m1 to the right by | E1 – E2 |= 1 bit position and update the exponent E1.
  m1 = 0.1 01111110010100011110101
  E1 = 132

- **Step 4:**
  Perform addition of mantissas, i.e., m = m1 + m2

  m1 = 0.10111111100101000111101 01
  m2 = 1.00111100001100110011001 0
  ---
  m= 1.11111011010111000010011 1

Floating Point Addition

b) Perform N1 + N2

- **Step 5 :**

  Determine the sign of the result (S), fractional mantissa (F) and biased exponent (E)

  S = 0, F1 = 11111011010111000010011

  E = E1 = E2 = 132 = 10000100

- **Step 6 :**

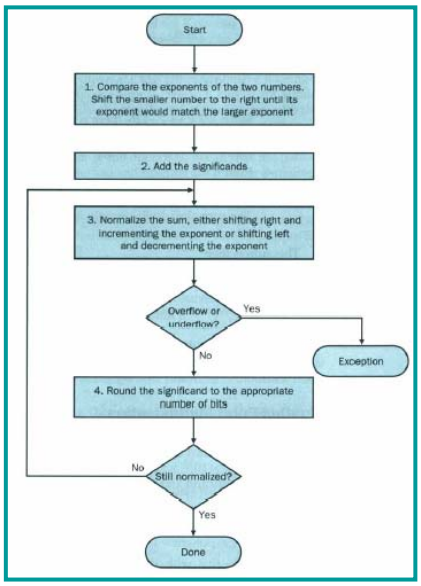
  Express the result in floating point packed form.

  In packed form,

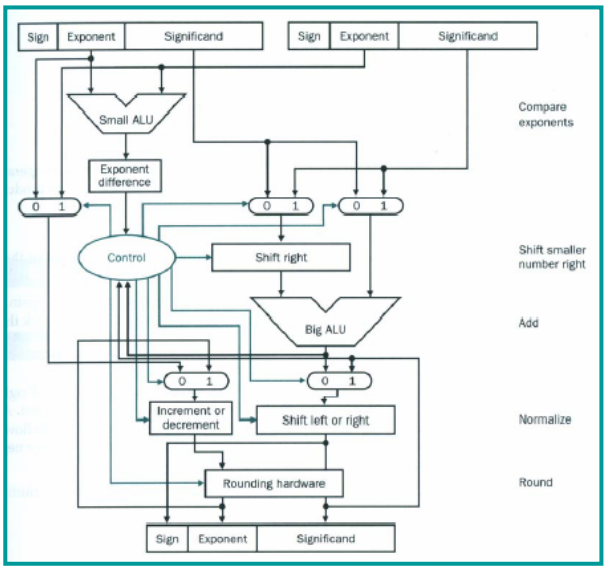
  N = 01000010011111011010111000010011

  = 427DAE13 H

Floating Point Addition



Floating Point Addition



ICS 233
COMPUTER ARCHITECTURE

Floating point Subtraction

Floating Point Subtraction

- **Given two Floating point number N1 and N2(in packed form). Perform N1 - N2 and store the result in packed floating point packed form**

- **Step 1 :**
  Check whether N1 is zero, then output (– N2) as the result
  else check whether N2 is zero then output N1 as the result
  else goto step 2
- **Step 2 :**
  Unpack the floating point packed numbers N1 and N2 to retrieve sign of mantissas(S1, S2) , biased exponents(E1, E2) and mantissas (m1, m2)
- **Step 3 :**
  Make the exponents E1 and E2 equal.
  i.e., Shift the smaller number mantissa to the right by | E1 – E2 | bit positions and update the exponent of smaller number to make it equal to that of the bigger number.
- **Step 4:**
  Perform subtraction of mantissas, i.e., m = m1 - m2
- **Step 5 :**
  Determine the sign of the result (S), fractional mantissa (F) and biased exponent (E) of the result
- **Step 6 :**
  Express the result in floating point packed form.

Floating Point Subtraction

Question :

a) Convert the following decimal numbers into the IEEE format for single precision floating point numbers and express each in packed form.

N1 = 23.895 N2 = 39.525

b) Perform N1 - N2 using the IEEE format binary floating point numbers obtained in part a) of the question. You should begin the calculation with the packed floating point representations of these numbers and end with the packed result.

Floating Point Subtraction

Solution :

a) N1 = 23.895
= 10111.11100101000111101 01

Normalizing,
N1 = 1.01111110010100011110101 x 2^4

S1 = 0, F1 = 01111110010100011110101, e1 = 4
E1 = e1 + 127 = 4 + 127 = 131 = 10000011

In packed form,
N1 = 01000001101111110010100011110101
= 41BF28F5 H

Floating Point Subtraction

Solution :

a) N2 = 39.525
= 100111.100001100110011001

Normalizing,
N2 = 1.00111100001100110011001 x 2^5

S2 = 0, F2 = 00111100001100110011001 , e2 = 5
E2 = e2 + 127 = 5 + 127 = 132 = 10000100

In packed form,
N2 = 01000010000111100001100110011001
= 421E1999 H

Floating Point Subtraction

b) Perform N1 - N2

- **Step 2 :**
  Unpack the floating point packed numbers N1 and N2 to retrieve sign of mantissas(S1, S2) , biased exponents(E1, E2) and mantissas (m1, m2)

- N1 = 41BF28F5 H
  S1 = 0, F1 = 01111110010100011110101
  m1 = 1. 01111110010100011110101
  E1 = 131

  N2 = 421E1999 H
  S2 = 0, F2 = 00111100001100110011001
  m2 = 1. 00111100001100110011001
  E2 = 132

Floating Point Subtraction

b) Perform N1 - N2

- **Step 3 :**
  Make the exponents E1 and E2 equal.
  Since E2 > E1, Shift m1 to the right by | E1 – E2 |= 1 bit position and update the exponent E1.
  m1 = 0.1 01111110010100011110101
  E1=132

Floating Point Subtraction

b) Perform N1 - N2

- **Step 4:**
  Perform subtraction of mantissas, i.e., m = m1 - m2

m1 = 0.10111111001010001111010 1 -
m2 = 1.00111100001100110011001 0

Sign bit
m1 = **0** 0.10111111001010001111010 1 -
m2 = **0** 1.00111100001100110011001 0

Take 2's complement of m2 and then perform addition
m1 = **0** 0.10111111001010001111010 1 **+**
m2 = 1 0.11000001111001100110011 0
---
m = 1 1.10000010111101011110000 11
S = 1, Take 2's complement of m
m= 0.01111101000010100011110 1
Normalize m and update E
m= 1.11110100001010001111010 1
E = 132 -2 = 130

Floating Point Subtraction

b) Perform N1 - N2

- **Step 5 :**
  Determine the sign of the result (S), fractional mantissa (F) and biased exponent (E)
  S = 1, F1 = 11110100001010001111010
  E = 130 = 10000010

- **Step 6 :**
  Express the result in floating point packed form.
  In packed form,
  N = 1100 0001 0111 1010 0001 0100 0111 1010
  = C17A147A H