Announcements

- Assignment 9
  - Writing a simple shell
  - Due 11:59 p.m. tonight!
- Final Exam
  - 12:00 – 2:00 p.m., Thursday, December 11
  - Comprehensive
    - Lectures 1-28 (70%)
    - Lectures 29-35 (30%)
- Review Session
  - 2:00 – 4:00 p.m., Wednesday, December 10
  - MECC 127

Lecture 35

Floating-Point Representations

CPSC 275
Introduction to Computer Systems

Floating Point

- ***Floating point*** is the representation used by most computer systems to approximate real numbers.
- Until the mid-80's, companies each developed their own standard for floating point, resulting in a lack of portability for data.
- In 1985, IEEE 754 was published and became the industry standard.
- The floating-point standard can be used to represent values of the form:

$$V = x * 2^y$$

Fractional Decimal

In decimal, we represent a value d as follows:

$$d_m d_{m-1} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-n}$$

↑ *decimal point*

$$d = \sum_{k=-n}^{m} d_k * 10^k$$

Fractional Binary

Binary works the same way; to represent a value b:

$$b_m b_{m-1} \dots b_1 b_0 . b_{-1} b_{-2} \dots b_{-n}$$

$\uparrow$ *binary point*

$$b = \sum_{k=-n}^{m} b_k * 2^k$$

Fractional Binary

Binary 101.01 = $1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$ = 5.25

In decimal, what happens when we shift the decimal point one place to the right? How about to the left?

It works the same in binary; shift to the right:
101.01 $\rightarrow$ 1010.1 = $1 * 2^3 + 1 * 2^1 + 1 * 2^{-1}$ $=$ 8 + 2 + ½ = 10.5

And a shift to the left:
101.01 $\rightarrow$ 10.101 = $1 * 2^1 + 1 * 2^{-1} + 1 * 2^{-3}$ $=$ 2 + ½ + $^1/_8$ = 2.625

Limitations

- In decimal, there are some numbers that cannot be precisely represented in a finite number of decimal places,
  - e.g. 1/3 = 0.3333333….
- Likewise in binary. We are limited to values that can be expressed as $x * 2^y$, where *x* must be able to be represented in a finite number of bits.
- For example, 1/5 can be represented exactly in decimal as 0.2, but not so in binary.

Representation	Value	Decimal
0.0_2	$\frac{0}{2}$	0.0_{10}
0.01_2	$\frac{1}{4}$	0.25_{10}
0.010_2	$\frac{2}{8}$	0.25_{10}
0.0011_2	$\frac{3}{16}$	0.1875_{10}
0.00110_2	$\frac{6}{32}$	0.1875_{10}
0.001101_2	$\frac{13}{64}$	0.203125_{10}
0.0011010_2	$\frac{26}{128}$	0.203125_{10}
0.00110011_2	$\frac{51}{256}$	0.19921875_{10}

IEEE Floating Point Standard

The standard has three components:

1. A sign bit ***S***, where *S* is 0 for (+), 1 for (-).
2. The *mantissa* ***M*** (or *significand*),
3. The exponent ***E***.

It is interpreted as:

$$V = (-1)^S * M * 2^E$$

IEEE Floating Point Standard

To determine the values of *S*, *M*, and *E* for a decimal number, say 10.5:

1. *S* is pretty easy.
2. Write the binary version of the magnitude. We saw before that 10.5 is `1010.1` in binary.
3. Move the binary point to the left or right until there is a single 1 to its left.
    - For this example, that gives us `1.0101` The number of positions moved is *E* = +3. (Moving left gives us a positive value of *E*, moving right gives us a negative value.)
4. The bits to the right of the binary point are the fractional part of *M*, `0101.`

Q: But what happens to the leading 1?

*A *normalized number* is a number where the most significant digit of the mantissa is non-zero.

IEEE Floating Point Standard

We can verify that the values $S = 0, M = 0101, E = 3$ have the value `10.5` by applying:

$$V = (-1)^S * M * 2^E$$

We obtain:

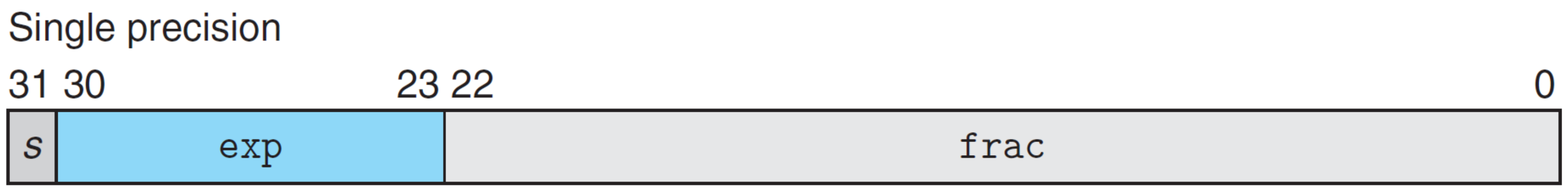
$$V = (-1)^0 * 1.0101 * 2^3 = 1 * \left(1 + \frac{1}{2^2} + \frac{1}{2^4}\right) * 2^3$$

$$= \left(1 + \frac{1}{4} + \frac{1}{16}\right) * 8$$

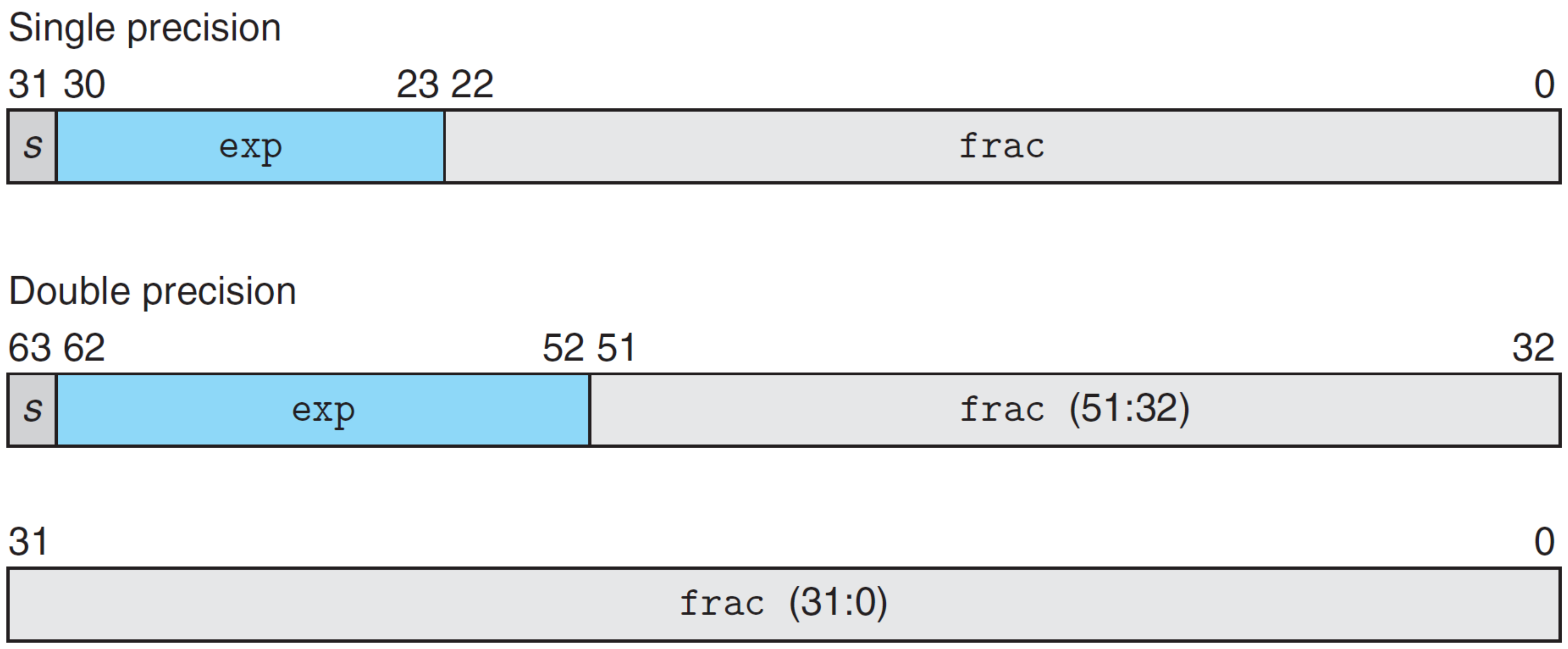
$$= \frac{21}{16} * 8$$

$$= 10.5$$

IEEE Floating Point Standard



IEEE Floating Point Standard



Q: But how about negative exponents?

IEEE Floating Point Standard

- To handle negative exponents, a *bias* is introduced.
- If there are *k* exponent bits, we calculate the bias as $2^{k-1} - 1$.
  - For example, if single precision format has 8 bits for exponent,
    $bias = 2^7 - 1 = 127$
- We add the bias to the actual exponent and store the result.
- The stored result is interpreted by subtracting the bias.
- Why introduce bias? (Homework)

Bias for the Exponent

- If we have k = 4 bits for the exponent, then the bias will be $2^{k-1} - 1 = 2^3 - 1 = 7$
- To determine (biased) bit representation from true exponent, we add 7 (that will ensure a non-negative value).
- To determine true exponent from (biased) bit representation, we subtract 7 (that will allow negative values).

Binary	E + 7	E
1111	15	8
1110	14	7
1101	13	6
1100	12	5
1011	11	4
1010	10	3
1001	9	2
1000	8	1
0111	7	0
0110	6	-1
0101	5	-2
0100	4	-3
0011	3	-4
0010	2	-5
0001	1	-6
0000	0	-7

IEEE Floating Point Standard – 8-bit model

S (1 bit)	**E** (4 bits)	**M** (3 bits)

Consider the value 6.5. In binary, that's 110.1

So S = 0, E = 2, and M = 1.101

Our bias is $2^{4-1} - 1 = 2^3 - 1 = 7$, so E = 2 + 7 = 9, or binary 1001.

We drop the leading 1 from M, leaving the fractional part 101.

The result is:

```
0 1 0 0 1 1 0 1
```

THANK YOU!