

## MODULE 2: Data Representation

# Lecture 2.2

# Signed Data Representation

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering  
Virginia Tech

# Lecture 2.2 Objectives

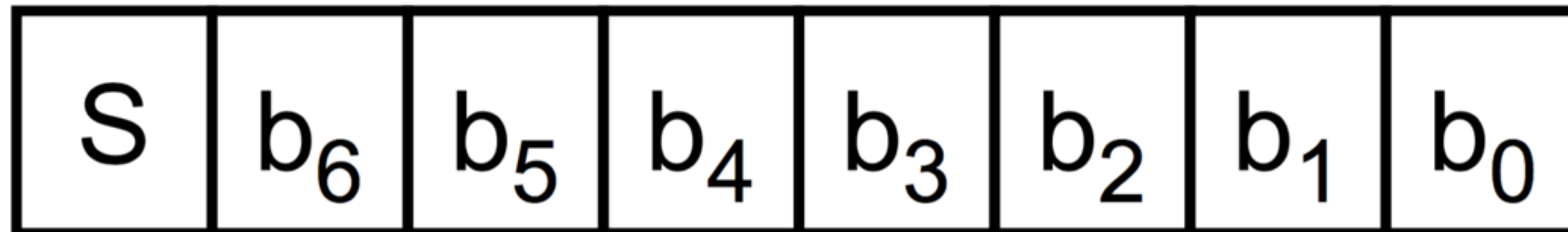
- Explain the difference between signed and unsigned representation
- Convert between signed decimal numbers and signed binary numbers using two's complement representation and using signed magnitude representation
- Explain the difference between the two's complement operation and two's complement representation
- Calculate the negative of a signed binary number

# Signed Numbers

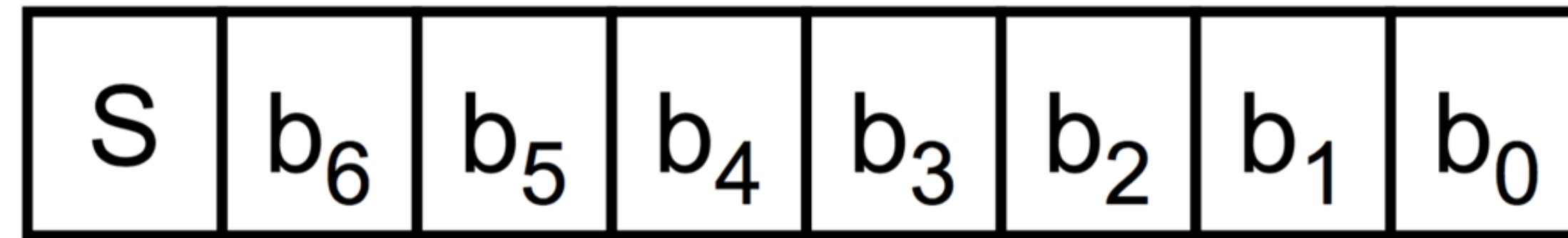
- Computers need to be able to represent signed (positive and negative) values as well as unsigned values
- Unsigned values are treated implicitly as positive, but there is no explicit information stored indicating that they are positive
- Signed numbers require that information is explicitly provided indicating if the value is positive or negative

# Signed Magnitude

- A simple way to indicate if a value is positive or negative is to add one bit, usually to the left of the most significant bit (MSB), indicating the sign
  - 0  $\Rightarrow$  positive
  - 1  $\Rightarrow$  negative
- For an 8-bit integer representation, there would be one sign bit (S) and seven magnitude bits ( $b_6 - b_0$ )



# Signed Magnitude Example



- For this scheme, the range is +127 to -127
  - 01111111 (+127) to 11111111 (-127)
- Conversion done by converting magnitude to or from binary and adding the sign bit
- Examples
  - $(+27)_{10} = (00011011)_2$
  - $(-27)_{10} = (10011011)_2$
  - $(0)_{10} = (00000000)_2$  or  $(10000000)_2$

# CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Using 8 bits, compute the signed magnitude representation of  $(-25)_{10}$

# CHECK POINT

Answer:

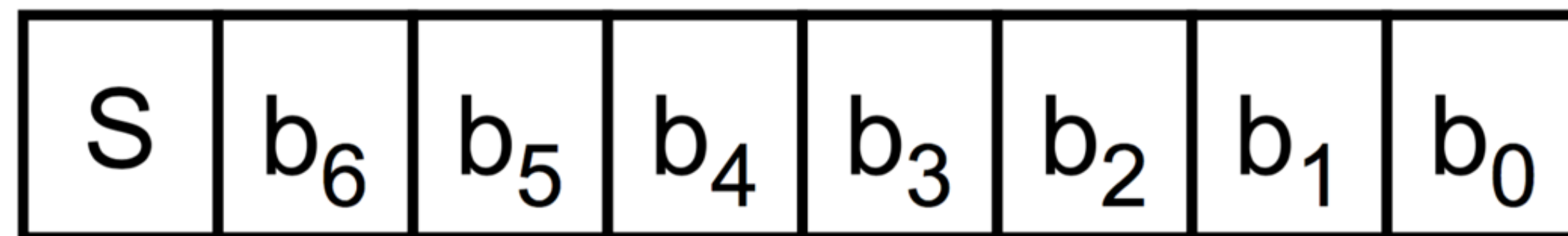
- $(-25)_{10}$  in (8 bits) signed magnitude representation: 10011001

If you have any difficulties, please review the lecture video before continuing.



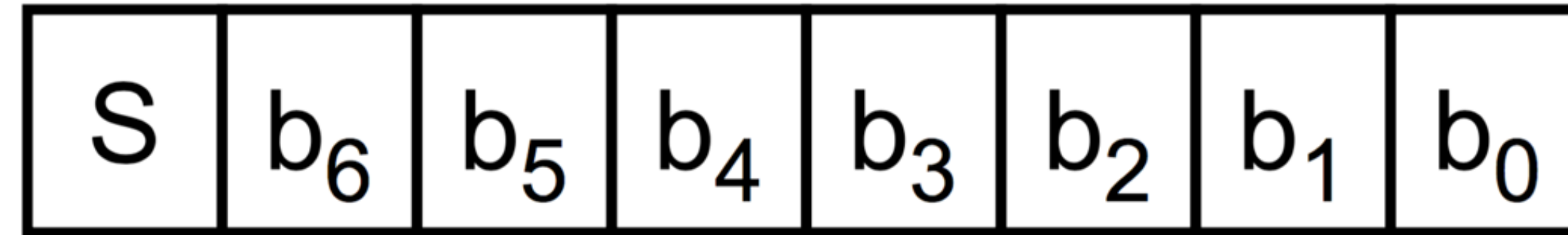
# Two's Complement Representation

- Two's complement representation is the most widely used signed representation for fixed point values
  - Single representation of zero
  - Arithmetic operations are easily implemented
- For an 8-bit integer representation, there would be one sign bit (S) and seven magnitude bits ( $b_6 - b_0$ ), but the magnitude bits are not simply the magnitude





# Two's Complement Example



- For this scheme, the range is +127 to -128
  - 01111111 (+127) to 10000000 (-128)
- Conversion done by:
  - Forming positive value of number (same as positive signed magnitude representation)
  - Performing two's complement *operation* on the positive number if the value is negative

# Two's Complement Operation (1)

- Process:
  - Invert (complement) every bit by changing each 1 to a 0 and each 0 to a 1
  - Add 1 to the value (using binary addition) and ignore overflow
- For binary addition
  - $0 + 0 = 0$  (no carry)
  - $0 + 1 = 1$  (no carry)
  - $1 + 0 = 1$  (no carry)
  - $1 + 1 = 0$  (carry 1 to the next digit)

# Two's Complement Operation (2)

- Example: find the two's complement of the 8-bit value 01001100
  - First, invert every bit: 10110011
  - Then, add 1 to this value

$$\begin{array}{r} \begin{array}{|c|} \hline 000011 \\ \hline \end{array} \leftarrow \text{carry bits} \\ 10110011 \\ +1 \\ \hline 10110100 \leftarrow \text{sum} \end{array}$$

- Two's complement of 01001100 is 10110100

# Two's Complement Conversion

- Example (using 8 bits):  $(+27)_{10}$ 
  - $(+27)_{10} = (00011011)_2$
  - Since it is a positive number, this is the same as the signed magnitude representation
- Example (using 8 bits):  $(-27)_{10}$ 
  - $(+27)_{10} = (00011011)_2$
  - Since we want the negative value, perform the two's complement operation on the representation of +27
  - $(-27)_{10} = (11100101)_2$
  - This is different than if we were using signed magnitude

# Negation with Two's Complement

- To negate a number, just perform the two's complement operation
- Example: find the negative of  $(11100101)_2$ 
  - Invert every bit: 00011010
  - Add 1: 00011011
  - We started with  $(-27)_{10}$  and then negated to get the representation for  $(+27)_{10}$

# CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Using 8 bits, compute the two's complement negative of  $(+27)_{10}$



# CHECK POINT

Answer:

The two's complement of  $(+27)_{10}$  is 00011011. To negate this, we perform the two's complement operation on 00011011:

- 00011011  $\leftarrow$  original
- 11100100  $\leftarrow$  after bit inversion
- 11100101  $\leftarrow$  after +1
- Therefore, the two's complement negation of  $+27_{10}$  is 11100101.

If you have any difficulties, please review the lecture video before continuing.



# Summary

- Computer systems may store values as unsigned or signed
- Note that the meaning of a sequence of bits is usually implicit, e.g., software or hardware must know whether to treat as signed or unsigned
- Signed magnitude is a simple representation, but requires relatively complex hardware or software for arithmetic
- Two's complement is a more complex representation (to us), but uses simpler logic for arithmetic

## MODULE 2: Data Representation

# Lecture 2.2

# Signed Data Representation

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering  
Virginia Tech