2’s and 1’s Complement

Lecture 2 Exercise

9/2/15

2’s Complement

ℓ = 8 bits of memory (one byte)

Can represent numbers from $0 \rightarrow 255$ ($00000000 \rightarrow 11111111$)

Or can represent signed numbers from $-128 \rightarrow +127$

Example: To represent -10, compute $2^{\ell} - 10 = 256 - 10 = 100000000 - 1010 = 11110110$

Represent 72 in binary:

Represent -35 in 2’s complement:

What is a quick way to tell whether a number is positive or negative?

Use 2’s complement to compute 72-35 by computing 72 + (-35):

What happens to the highest order carry?

1’s Complement

ℓ = 8 bits of memory (one byte)

Can represent numbers from $0 \rightarrow 255$ ($00000000 \rightarrow 11111111$)

Or can represent signed numbers from $-127 \rightarrow +127$ (there are 2 ways to represent 0—what are they?)

Example: To represent -10, compute 10 in binary: 1010. To compute -10, flip the bits: 11110101

Represent 72 in binary:

Represent -35 in 1’s complement:

What is a quick way to tell whether a number is positive or negative?

Use 1’s complement to compute 72-35 by computing 72 + (-35) (note there is an extra step that must be done during subtraction—what is it?):

Is an extra step during subtraction necessary when computing 35 + (-72)?

2’s and 1’s Complement

Lecture 2 Exercise

9/2/15

Discussion

- Which is easier—computing the 2’s complement or 1’s complement?
- Which is easier—subtraction using 2’s complement or 1’s complement?
- How can we use the 1’s complement to compute the 2’s complement?
- Which do you think should be the preferred choice?