Class 3: Representing Information – Signed and Unsigned Numbers

- Signed and Unsigned Numbers
  - Sign Magnitude
  - One's complement
  - Two's complement
  - Excess 2^{m-1}
- Addition and Subtraction
  - Overflow
  - Binary arithmetic
  - Hexadecimal arithmetic

Signed and Unsigned Numbers

- Most of the numbers we have been looking at have been unsigned.
- For signed numbers there are several different formats:
  - Sign-Magnitude – the left-most bit provides the sign
  - One's Complement – numbers are inverted
  - Two's Complement – numbers are inverted and one is added

Sign-Magnitude

- The simplest format – the left-most bit provides the sign (0 for positive, 1 for negative)
- Examples (for 8 bit numbers):

  +5 =

  -5 =
- Two representations for zero!

  00000000

  10000000

One's Complement

- Again, the sign is indicated by the left-most bit.
- Numbers are converted to negative by inverting all the bits.
- Example (8-bit numbers):

  +5 =

  -5 =
- Still two representations for zero:

Try it Out!

- What's the eight-bit 1's complement representation of:

  -93 (+93 = 01011101)

  -68 (+68 = 01000100)

  -7340h

Two's Complement

- Two's complement conversion is a two step process:
  1. Invert all the bits (the same as for one's complement)
  2. Add one. If a carry is generated, discard it.
- Examples (eight-bit numbers):

  +5 = 00000101

  -5 =

Two's Complement (Continued)

- Conversion:

  -00000110 (-6)

- One representation for zero:

More examples

- What's the 2's complement representation of:

  -01011101

  -01000100

  -7340h

  -22_{10}

What do these representations have in common?

- For negative numbers the left-most bit is always 1!
- This bit is commonly known as the sign bit.

What about positive numbers?

- What's the sign magnitude representation of 26?
- What's the 1's complement representation of 26?
- What's the 2's complement representation of 26?

Common Mistakes

- Trying to combine sign-magnitude with the other number formats.
- Taking the complement of a positive number.
- Not using the correct number of bits! Make sure your number is the correct number of bits before converting.

Excess-2^{m-1}

- For m-bit numbers, the number is represented by storing it as the sum of itself and 2^{m-1}.
- 8-bit numbers – excess 128 – number is stored as its true value plus 128
- Example:
  - -3 becomes –3 + 128 = 125
    - = 01111101 (binary representation of 125)
  - +3 becomes 3 + 128 = 131
    - = 10000011 (binary representation of 131)
- Identical to 2's complement with the sign bit reversed!

Trying Excess-128

- What's the Excess-128 representation of:

14

-8

Reversing it

- What are the decimal values of the following excess-128 numbers?

01001000

10001001

Comparison of Methods

- Table from p. 639, Tanenbaum

Sign and Zero Extension

- What if you need to convert an eight-bit two's complement number to sixteen bits?
- Extend the sign bit (left-most bit) by adding 8 more ones (for negative) or zeros (for positive) in front.
- Binary:

  10000001 -> 1111111110000001

  00101000 -> 0000000000101000
- Hexadecimal (same numbers):

  81h -> FF81h

  28h -> 0028h

Binary Arithmetic

- Start at the right-most bit and add the corresponding bits in the two numbers.
- If a carry is generated, it is carried one position to the left, just as in decimal arithmetic.
- To subtract, add the negative value of the subtracted number:

  10 – 3 = 10 + (-3) = +7

Binary Arithmetic

- Figure A-8 (Tanenbaum)
- Figure A-9 (Tanenbaum)

Binary Arithmetic Examples

- Two's complement numbers:

```
  00101001
+ 00101110
```

```
  10011101
- 00010011
```

Overflow

- If the two numbers (addend and augend) are of opposite signs, overflow can not occur
- If they are the same sign and the result is the opposite sign, overflow has occurred

→ if the carry into the sign bit is different from the carry out of the sign bit then overflow has occurred

Hexadecimal Addition and Subtraction

- Remember – it's not decimal, even when it looks like it!
  1h + 9h = Ah (not 10h!)
- Add from right to left like you would decimal.
- If the sum of two digits is greater than 15, a carry is generated:
  Ah + 9h = 19_{10}
  $19_{10}/16_{10}$ = quotient 1, remainder 3
  Put the remainder in the lowest digit position and carry the quotient to the next highest position
  so, Ah + 7h = 13h

Hexadecimal Addition and Subtraction

- Addition example (unsigned numbers):

```
  3 B A 8
+ 0 2 B 5
  3 E 5 D
```

- Subtraction example:

```
  3 B A 8
- 0 0 0 9
  3 B 9 F
```

More examples

- Again, two's complement numbers:

```
  7F23h
+ 034Ch


  2B80h
- C1A4h
```