Integer Representation
Error Detection

Error Detection

- Errors are possible as the number representation uses a fixed length
- Certain operations will cause the result to exceed
  - the maximum positive value
  - the maximum negative value
- Why do we care?
  - The result of such an operation cannot be expressed in the fixed length

When Can Error Occur

- Unsigned Numbers
  - Adding two positive numbers
  - Subtracting a larger number from a smaller number
- Complement Numbers
  - Adding two numbers of the same sign (either +ve or -ve)
  - Subtracting a negative number from a positive number
  - Subtracting a positive number from a negative number
- Only the one of the above (orange text) is a guaranteed error → why?

Error Detection

- There is no way to avoid such errors
  - Why does widening the data size not work?
- There are way(s) to detect when each of the errors occurs
- What the error is and how to detect it are **different**

Carry Error

- When the result of a math operation exceeds the allowable range, when the values are interpreted as unsigned values
  - Only applies to unsigned values
  - Does not specify a particular math operation
  - Detection is different for different operations
- Textbook **incorrectly** refers to this as an unsigned overflow

Carry Error - Detection

- During addition, a carry error occurs
  if and only if a carry out of the MSBit occurs
- During subtraction a carry error occurs
  if and only if a borrow into the MSBit is required

Carry Error - Examples

Assuming 4-bit unsigned numbers are being used

```
          0 111        there is NO carry error
  5         0101       since the carry out is 0
+ 3       + 0011
  8         1000

          1            there IS a carry error
   9  1001        since the carry out is 1
+ 10+ 1010
  19  0011
```

Carry Error - Examples

Assuming 4-bit unsigned numbers are being used

```
                112
            1 2202        there IS a carry error
    2         0010        since a borrow TO the
  - 3       - 0011        MSB was required
   15         1111
```

Overflow Error

- When the result of a math operation exceeds the allowable range, when the values are interpreted as signed values
  - Only applies to signed values
  - Does not specify a particular math operation

Overflow Error

- When the result of a math operation exceeds the allowable range, when the values are interpreted as signed values
  - Only applies to signed values
  - Does not specify a particular math operation

Overflow Error - Detection

- During addition, an an overflow error occurs if and only if carry in to the MSB != carry out from the MSB
  - it is possible to check the signs of the three items but this is far more convoluted and difficult
  - so is not an acceptable method.
- Why is only addition specified and not subtraction?

Overflow Error - Examples

Assume that 4-bit 2's comp signed numbers are being used

```
          0 01     there is no overflow error
   2        0010      since carry in = carry out
 + 3      + 0011
   5        0101

          0 111       there is an overflow error
   5        0101      since carry in (1) != carry out (0)
 + 3      + 0011
  -8        1000
```

Error Detection - Final Note

IMPORTANT!

- For a particular operation it is possible to have both errors, one of the errors or neither error
- 2's complement selected since it has the same addition circuit as unsigned

∴ Only the error detection circuitry is different

- i.e. the CPU does not care if the number being added are signed or unsigned, the add operation uses the add circuit and **both** error detection circuits, and the user determines which error to look for