One's Complement Notation

Text 2.2 pages 27- top of 31
pages 33-34

Odometer Numbers

- **Completely unrelated to signed magnitude notation**
- Look at a 3 digit odometer
  - smallest digit is 000 and largest digit is 999
  - 1000 different values represented
  - moving forward from 000 gives 001, 002, 003
  - moving backward from 000 gives 999, 998, 997
    - can use these as negatives

One's Complement

- Use the odometer idea, and split the unsigned range
  - In half by convention
- Advancing from 0 gives positive
- Reversing from 0 gives negative
- Same overall number of values able to represented
- Same range as Signed Magnitude
  - Assignment of bit patterns to negative numbers is different
- Next slide shows bit pattern assignment for 4-bit One's Complement Binary

One's Complement

Assume a 4-bit fixed length

Decimal Number	One's Complement Representation	Hex
7	0111	7
6	0110	6
5	0101	5
4	0100	4
3	0011	3
2	0010	2
1	0001	1
0	0000	0
-0	1111	F
-1	1110	E
-2	1101	D
-3	1100	C
-4	1011	B
-5	1010	A
-6	1001	9
-7	1000	8

One's Complement

Notice:

- All positives start with a zero (0), all negatives start with a (1)
- Positives same as regular binary, but only uses n-1 bits
  - Bit n-1 (MSBit) must be zero for the value to be valid
  - ex. $13_{10} = 1101_2$ bit 3 is a 1 therefore 13 not a valid 4-bit number

One’s Complement

- One’s Complement no longer sets the MSBit to indicate the sign
- Use the NEG_{rep} of the positive number to represent the negative number
- This changes the MSBit from an explicit sign bit, to an implicit sign bit
- An MSBit of 0 is still positive
- An MSBit of 1 is still negative
- While the MSBit indicates a sign it is not a sign bit
    - It is part of the number

One's Complement

Add any positive and negative 4-bit value (e.g. 5 and -5)

```
  0101
+ 1010
  1111 (maximum 4-bit value)
```

- positive n-bit + negative n-bit = maximum n-bit value
  - $NEG_{1's\ comp}(N)$ = max value - N **or**
  - $NEG_{1's\ comp}(N)$ = flip each bit

Decimal to One's Complement Conversion

Convert Decimal to N-bit One's Complement Binary

1. result = abs(num) converted to binary
2. **IF** MSBit (result) != 0 OR length(result) > N bits **THEN**
    - "*num* cannot be represented in N bits"
3. **ELSE**
    - **IF** num < 0 **THEN**

        result = $NEG_{1's\ Comp}$(result)

NEG_{rep} is the negation operation **NOT** make negative
Thus, the NEG operation can take **ANY** value

One's Complement

Convert 115_{10} to 8-bit one's complement binary, expressed in hex

1. abs(115) = 115
2. 115_{10} converted to 8-bit binary is 01110011_2
3. Since the bit 7 is 0 (and no bit 8 or more was required) 115_{10} is valid
4. Since the number is positive nothing further is done
5. In hex => 73_{16}

One's Complement

Convert -67_{10} to 8-bit one's complement binary, expressed in hex

1. abs(-67) = 67
2. 67_{10} converted to 8-bit binary is 01000011_2
3. Since bit 7 is 0 (and no bit 8 or more was required) -67_{10} is valid
4. Since the number is negative apply $NEG_{1's\ comp}(01000011_2) = 10111100_2$
5. In hex => BC_{16}

One’s Complement to Decimal Conversion

Convert One’s Complement Binary to Decimal

1. **IF** MSBit(binary) == 1 **THEN**
        temp = $NEG_{1's\ comp}$(binary)
   ELSE
        temp = binary
2. result = temp converted to decimal
3. **IF** MSBit(binary) == 1 **THEN**
        result = -1 * result

One's Complement

Convert 00011011_2 1's comp to base 10

1. The MSBit is 0 so this is a positive number

   No changes made to the number

2. 00011011_2 converted to base-10:

   `result = 16 + 8 + 2 + 1 =` 27_{10}

3. Since the MSBit = 0 nothing further needs to be done

   `result =` 27_{10}

One's Complement

Convert 10100110_2 1's comp to base 10

1. The MSBit is 1 so this is a negative number

   $NEG_{1\text{'s comp}}(10100110_2) = 01011001_2$

2. 01011001_2 converted to base-10

   $result = 64 + 16 + 8 + 1 = 89_{10}$

3. The MSBit is 1

   $result = -1 * result = -89_{10}$

One's Complement

Try the following:

```
               1
3 + 2 =      0011
           + 0010
             ----
             0101

                   1 111
3 + -2 =            0011
                  + 1101
                    ----
                    0000        WHAT?
```

One's Complement - End Around Carry

To make addition work, the carry out bit **must** be added to the result of the addition

```
           0  1                    1 111
3 + 2 =      0011    3 + -2 =        0011
           + 0010                  + 1101
             0101                    0000
           +    0                  +    1
             0101                    0001
```

Therefore 1's complement addition requires two binary adds

One's Complement - Eliminating Subtraction

Unlike for Signed Magnitude subtraction can be eliminated

```
3 - 2            =    3 + (-2)

x - y            =    x + (-y) (generalized form)

  0011           ⇔      0011
- 0010                + 1101
  0001                  0000
                      +    1
                        0001
```

One's Complement

- A fixed-length number using one's complement representation has range:
  - **$-(2^{(N-1)} - 1)$ to $2^{(N-1)} - 1$** (same as SM)
- Half the values are positive and half the values are negative
- Zero can be represented in two ways (same as SM)
  - 00000000 (+0)
  - 11111111 (-0) (negative 0 differs between SM and 1's complement)

One's Complement

Pros:

✓ Easy (one step) negation

✓ Subtraction is just addition of negation (no extra circuit required)

✓ All numbers have inverses

✓ MSB indicates sign (negative circuit test is simple)

One's Complement

Cons:

- × two zeros, so extra logic circuit
- × different circuit required for signed/unsigned addition (end around carry for signed addition)
- × negative numbers not directly identifiable