Number Systems II

Signed Binary Number Encoding

- The leftmost bit (**high-order/most significant bit**) indicates whether a number is NEGATIVE (1) or NON-NEGATIVE(0)

- Two potential signed binary encodings -
  - Signed Magnitude
  - Two's Complement

Signed Magnitude

- Treats the high-order bit exclusively as a sign bit
- If the high-order bit is 1 then it is negative, and 0 when non-negative
- The high-order bit does not affect the absolute value of the number

Converting Signed Binary to Decimal

- Compute a decimal value for the digits indexed from 0 to (N-2)
- Check the $(N-1)^{th}$ index (most-significant bit). If it's 1, the value is negative, otherwise it is non-negative

$$(1101)_2 = -(1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)$$

$$= -(1 \times 4 + 0 \times 2 + 1 \times 1)$$

$$= -5$$

Two's Complement

- Treats the high-order bit both as a sign bit and also affects the value of the number

- If the high-order bit is 1 for an N-bit binary number then $(-1x\ 2^{N-1})$ is added to the total sum of all the remaining bits and the number becomes negative

- If the high-order bit is 0 for an N-bit binary number then $(0\ x\ 2^{N-1})$ is added to the total sum of all the remaining bits and the number becomes positive

Converting Two's Complement to Decimal

- Compute a decimal value for the digits indexed from 0 to (N-2)
- Check the $(N-1)^{th}$ index (most-significant bit). If it's 1, the value is negative, otherwise it is non-negative

$$(1101)_2 = (-1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)$$

$$= (-1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1)$$

$$= -8 + 4 + 0 + 1$$

$$= -3$$

Converting Negative Decimal to Two's Complement

- Start with decimal in binary
- Flip all the bits
- Add 1
- Example: to get -13

```
0000 1101  (decimal 13)

1111 0010  (flip all the bits)
0000 0001  (add 1)

1111 0011  (decimal -13)
```

Addition of two unsigned binary numbers

- In case of Binary numbers, each digit holds only 0 or 1
- When adding two bits that are both 1, the result carries out to the next digit
- That means there are also carry ins during the addition of two digits
- When summing two binary numbers A and B, there are eight possible outcomes depending on the values of $Digit_A$, $Digit_B$, and a $Carry_{in}$ from the previous digit

Inputs			Outputs	
$Digit_A$	$Digit_B$	$Carry_{in}$	Result (Sum)	$Carry_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Examples

1.
```
  0 0 1 0
+ 1 0 1 1
---------
Result: 1 1 0 1
```

```
  1 ← Carry the 1 from digit 1
0 0 1 0       into digit 2
+ 1 0 1 1
```

2.
```
  1 1 0 0
+ 0 1 1 1
---------
Result:   0 0 1 1
Carry out: 1
```

```
1 1 ← Carry a 1 from: digit 2 into
1 1 0 0 digit 3, and digit 3 out of
+ 0 1 1 1 the overall
```

Bitwise Operators

- Bitwise AND
- Bitwise OR
- Bitwise XOR
- Bitwise NOT
- Bitwise Shifting
  - Shifting Left
  - Shifting Right

Bitwise Operations

Bitwise operations follow the Truth table for each type of operator

Table 1. The Results of Bitwise ANDing Two Values (A AND B)

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

Table 2. The Results of Bitwise ORing Two Values (A OR B)

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise Operations

Bitwise operations follow the Truth table for each type of operator

Table 3. The Results of Bitwise XORing Two Values (A XOR B)

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

Table 4. The Results of Bitwise NOTing a Value (A)

A	~ A
0	1
1	0

Bit Shifting

- Two types of Bit Shifting
  - Shifting Left: "<<" is used as the operand
  - Shifting Right: ">>" is used as the operand

Shifting Left

- Shifting a sequence to the left by N places moves each of its bits to the left N times

- Appends new zeros to the right side of the sequence

- Example:

  Shifting 0b00101101 to the left by 2 places

  Result: 0b10110100

Shifting Right

- Two variants:
  - Logical Right Shift
  - Arithmetic Right Shift

Logical Right Shift

- Prepends new zeros to the left side (high-order bits) of the sequence
- Example:

  Shifting 0b10110011 to the right by 2 places

  Result: 0b**00**101100

Shifting Right

Arithmetic Right Shift

- Prepends a copy of the shifted value's most significant bit into each of the new bit positions

- Example:

  Shifting 0b10110011 to the right by 2 places

  Result: 0b**11**101100