Topic 2 : Building a Computer

Fundamentals 1 : Information Representation

Reference : D & L: Ch 2. + Ch 3 pages 53-69

Why Study Binary

- In a digital computer all information is represented using the binary number system.
- Reason: because it is easy to build electronic circuits which operate on the basis of two states, either on or off.

Number Systems

- Labelling the columns.

Hundreds	Tens	Units
7	1	1

Number Systems

- Labelling the columns.

Hundreds	Tens	Units
7	1	1
10^2	10^1	10^0
7	1	1

Number Systems (1)

- Labelling the columns.

Hundreds	Tens	Units
7	1	1
10^2	10^1	10^0
7	1	1

711 = 7 * 100 + 1 * 10 + 1 * 1

Number Systems (2)

- Arithmetic

Hundreds	Tens	Units	
7	1	1	
	9	9	+

Number Systems (3)

- Arithmetic

Hundreds	Tens	Units	
7	1	1	
	9	9	+
8	1	0	

Number Systems (4)

- Range

Hundreds	Tens	Units
	9	9

In general: Range is $10^N - 1$

The Basics of Binary

- Let's consider an 8 digit binary number
- Label the columns starting at the right in increasing powers of 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128's	64's	32's	16's	8's	4's	2's	1's

The Basics of Binary

- Let's consider an 8 digit binary number
- Label the columns starting at the right in increasing powers of 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128s	64s	32s	16s	8s	4s	2s	1s
1	0	1	0	1	0	1	1

= 128 + 32 + 8 + 2 + 1 = 171

Some Binary Bits and Pieces

- Terminology
  - Each BInary digiT is called a bit.
  - Eight binary digits are called a byte.
  - The left most bit is called the most significant bit (or MSB).
  - The right most bit is called the least significant bit (or LSB).
- Range
  - Normally, range is 0 .. $2^N - 1$

Binary Arithmetic

- Rules for binary arithmetic are the same as for decimal arithmetic.

0 0 1 0 1 0 1 0 = 42

0 0 0 1 1 1 1 1 = 31

Binary Arithmetic

- Rules for binary arithmetic are the same as for decimal arithmetic.

```
  0 0 1 0 1 0 1 0     =  42
  0 0 0 1 1 1 1 1     =  31
-----------------------------
  0 1 0 0 1 0 0 1     =  73
-----------------------------
```

Overflow

- Binary numbers often have to be held in a fixed number of bits.
- This introduces the problem of overflow.

Binary Arithmetic (1)

```
  1 0 0 0 1 1 1 0     =  142
  1 0 0 0 0 0 1 1     =  131
-----------------------------

-----------------------------
```

Binary Arithmetic (2)

```
    1 0 0 0 1 1 1 0     =  142
    1 0 0 0 0 0 1 1     =  131
-------------------------------
 1  0 0 0 1 0 0 0 1     ≠  273
-------------------------------
```

- Overflow has occurred and the answer is wrong !!

Negative Numbers

- The previous section assumes only positive numbers are represented.
- Three main methods for representing negative numbers (all use the same method for positive numbers)
  - (a) Sign and Magnitude
  - (b) Diminished Radix Complement (One's Complement)
  - (c) Radix Complement (Two's Complement)

(a) Sign and Magnitude

- Use the left most bit (the sign bit) to signal whether the number is positive or negative.

e.g. 0 1 0 0 1 0 0 1 = +73

1 1 0 0 1 0 0 1 = -73

- Range: $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$

(a) Sign and Magnitude

- Use the left most bit (the sign bit) to signal whether the number is positive or negative.

e.g. 0 1 0 0 1 0 0 7 = +73

1 1 0 0 1 0 0 7 = -73

- Range: - (2^{N-1} - 1) to +(2^{N-1} - 1)
- Two distinct representations of zero: +0, -0

(b) One's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

(b) One's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

↓

1

(b) One's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

↓

0 1

(b) One's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

↓

0 0 1

(b) One's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

↓

1 0 0 1

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

⬇

0 1 0 0 1

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

⬇

0 0 1 0 0 1

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

⬇

1 0 0 1 0 0 1

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

⬇

0 1 0 0 1 0 0 1

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

0 1 0 0 1 0 0 1 = 73

(b) One’s Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1.

e.g 1 0 1 1 0 1 1 0 is a negative number

0 1 0 0 1 0 0 1 = 73

Therefore original number = -73

(b) One's Complement ... contd.

- Range : Same as sign and magnitude.
- Notes:
  - Also has double representation of zero.
    - +0 (00000000), -0 (10000000)
  - But, is easier to design circuits which perform arithmetic on numbers represented in this form.

(c) Two's Complement

- Any pattern whose sign bit is 1 represents the negative of the value obtained by replacing each 1 with a 0 and each 0 with a 1, and then adding 1 to the final value.
- So, as with one's complement except you add an extra one (hence two's complement!).

(c) Two's Complement ... contd.

- e.g. 1 0 1 1 0 1 1 1 is a negative number.

(c) Two's Complement ... contd.

- e.g. 1 0 1 1 0 1 1 1 is a negative number.

```
 = 0 1 0 0 1 0 0 0 +
                 1
 _________________

 _________________
```

(c) Two's Complement ... contd.

- e.g. 1 0 1 1 0 1 1 1 is a negative number.

```
 = 0 1 0 0 1 0 0 0 +
                 1
 _________________
   0 1 0 0 1 0 0 1     = 73
 _________________
```

Therefore original number was -73

(c) Two's Complement ... contd.

- Note: given 8 bits, the negative of 00000000 (-0) is effectively 00000000…

```
              0 0 0 0 0 0 0 0
complemented  1 1 1 1 1 1 1 1
                            1  +
             __________________
             ① 0 0 0 0 0 0 0 0
```

(c) Two's Complement ... contd.

- Only one representation of zero because of overflow.

```
             0 0 0 0 0 0 0 0
complemented 1 1 1 1 1 1 1 1
                           1 +
           -----------------
           1 0 0 0 0 0 0 0 0
```

- However, the negative value 10000000 doesn't have a positive equivalent.
- Range, therefore: - (2^{N-1}) to +(2^{N-1}- 1)

(c) Two's Complement ... contd.

- You can perform arithmetic using two's complement numbers.

```
0 0 0 1 1 1 1 1 =    31 +
1 1 0 1 0 1 1 1 =  - 41
-----------------------

-----------------------
```

(c) Two's Complement ... contd.

- You can perform arithmetic using two's complement numbers.

```
0 0 0 1 1 1 1 1 =    31
1 1 0 1 0 1 1 1 =  - 41
-----------------------
1 1 1 1 0 1 1 0 =  - 10
-----------------------
```

(c) Two's Complement ... contd.

- Note that overflow is important again.

```
0 1 0 0 0 0 1 0 =    66
0 1 0 0 0 0 1 0 =    66
-----------------------

-----------------------
```

(c) Two's Complement ... contd.

- Note that overflow is important again.

```
0 1 0 0 0 0 1 0 =    66
0 1 0 0 0 0 1 0 =    66
-----------------------
1 0 0 0 0 1 0 0 ≠   132
-----------------------
```

Floating Point Numbers (1)

- All of the representations described so far support only integers.
- We can support floating point numbers by storing both a fraction (f) and an exponent (e).

$$f * base^{e}$$

Floating Point Numbers (2)

- Using such a system numbers can be stored using different representations (e.g. sign and magnitude, two's complement etc..).
- Generally, floating point numbers are normalised when they are stored such that the most significant bit is non-zero.

Converting to Binary

- Three main methods:
  - Keep dividing by two and the remainder forms the number.
  - Keep subtracting decreasing powers of 2.
  - Look and guess (subtraction).

Successive Division

- Convert 135_{10} to base 2

Successive Division

- Convert 135_{10} to base 2

LSB

135 / 2	=	67 remainder	1
67 / 2	=	33 remainder	1
33/2	=	16 remainder	1
16/2	=	8 remainder	0
8/2	=	4 remainder	0
4/2	=	2 remainder	0
2/2	=	1 remainder	0
1/2	=	0 remainder	1

MSB

Subtraction

- Convert 135 to base 2

Subtraction

- Convert 135 to base 2

MSB

135	we can	1	subtract 128 (r7)
7	we cant	0	subtract 64
7	we cant	0	subtract 32
7	we cant	0	subtract 16
7	we cant	0	subtract 8
7	we can	1	subtract 4 (r3)
3	we can	1	subtract 2 (r1)
1	we can	1	subtract 1 (r0)

LSB

Look and Guess (subtraction)

- Convert 135 to base 2

Look and Guess (subtraction)

- Convert 135 to base 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128's	64's	32's	16's	8's	4's	2's	1's
1	0	0	0	0	1	1	1

Conversion of Negative Numbers

- Convert the positive equivalent.
- Then change it to a negative number depending on the representation used.
- If using two's complement remember to add the 1 after complementing.

Representation of Characters

- Characters such as 'A' must also be represented.
- Standard representation (ASCII code).

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

◆ Also have to represent numbers as characters, e.g. '7' but note these aren't the same as 7.

Hexadecimal Notation

- Binary is rather inconvenient to read and write for humans.
- Often use base 16 (hexadecimal)
  - i.e. 0 .. 9, A. B, C, D, E and F
- 8 binary digits can be represented using 2 hex digits.
  - e.g. $01001001_2 = 49_{16}$
  - $11111111_2 = FF_{16}$

Octal Notation

- An alternative to both binary and hexadecimal is octal notation.
- Octal = base 8.
- 3 binary digits = 1 octal digit.
  - e.g. $01001001_2 = 111_8$
  - $11111111_2 = 377_8$

Summary

- Numbers and characters may be represented using binary.
- Variety of techniques for representing negative numbers.
- Can perform arithmetic on binary numbers.
- Can easily convert between more readable hexadecimal and binary notation.

Coming Next Week…

- Fundamentals 2 : Computer Logic
- Reference : D & L: Chapter 4.