Positional notation – Ch. 2.2 …/continued

Conversions between Decimal and Binary

Binary to Decimal

Technique

- use the definition of a number in a positional number system with base 2
- evaluate the definition formula using decimal arithmetic

Example

```
543210 ← position = corresponding power of 2
||||||
```

$$101011 = 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 = 43 \text{ (decimal)}$$

$d_5d_4d_3d_2d_1d_0$

Decimal to Binary

Technique

- repeatedly divide by 2
- remainder is the next digit
- binary number is developed right to left

Example

173 ÷ 2	86	1	1
86 ÷ 2	43	0	01
43 ÷ 2	21	1	101
21 ÷ 2	10	1	1101
10 ÷ 2	5	0	01101
5 ÷ 2	2	1	101101
2 ÷ 2	1	0	0101101
1 ÷ 2	0	1	10101101

What is going on in the repeated division by 2?

Example

```
43 ÷ 2      21   1          1
21 ÷ 2      10   1         11
10 ÷ 2       5   0        011
 5 ÷ 2       2   1       1011
 2 ÷ 2       1   0      01011
 1 ÷ 2       0   1     101011
```

43 (decimal) $= 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

43 ÷ 2 produces a quotient of 21, with a remainder of 1

$43 \div 2 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1}$

21 (decimal) $= 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

21 ÷ 2 produces a quotient of 10, with a remainder of 1

$21 \div 2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1}$

10 (decimal) $= 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

10 ÷ 2 produces a quotient of 5, with a remainder of 0

$10 \div 2 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1}$

5 (decimal) $= 1\times2^2 + 0\times2^1 + 1\times2^0$

5 ÷ 2 produces a quotient of 2, with a remainder of 1

$5 \div 2 = 1\times2^1 + 0\times2^0 + 1\times2^{-1}$

2 (decimal) $= 1\times2^1 + 0\times2^0$

2 ÷ 2 produces a quotient of 1, with a remainder of 0

$2 \div 2 = 1\times2^0 + 0\times2^{-1}$

1 (decimal) $= 1\times2^0$

1 ÷ 2 produces a quotient of 0, with a remainder of 1

$1 \div 2 = 0\times2^0 + 1\times2^{-1}$

Generalization: Conversions between Decimal and base *b*

<u>base *b* to Decimal</u>

Technique

- use the definition of a number in a positional number system with base *b*
- evaluate the definition formula using decimal arithmetic

<u>Decimal to base *b*</u>

Technique

- repeatedly divide by *b*
- remainder is the next digit
- base *b* number is developed right to left

Conversions between Binary and Octal/Hexadecimal

<u>Binary to Octal</u>

Technique

- group bits into threes, right to left
- convert each such group to an octal digit

Example

`1011010111 = 001 011 010 111 = 1327` (octal)

<u>Binary to Hexadecimal</u>

Technique

- group bits into fours, right to left
- convert each such group to a hexadecimal digit

Example

`1011001011 = 0010 1100 1011 = 2CB` (hexadecimal)

Octal to Binary

Technique

- convert each octal digit to a three-bit binary representation

Example

`705 = 111 000 101 = 111000101` (binary)

Hexadecimal to Binary

Technique

- convert each octal digit to a four-bit binary representation

Example

`10AF = 0001 0000 1010 1111 = 1000010101111` (binary)

You need the following tables in the above conversion processes:
(These can easily be reconstructed in the margins of a test paper when needed)

Binary ←→ Octal

Octal Digits	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Binary ←→ Hexadecimal

Hex Digits	Binary			
0	0000		8	1000
1	0001		9	1001
2	0010	(10)	A	1010
3	0011	(11)	B	1011
4	0100	(12)	C	1100
5	0101	(13)	D	1101
6	0110	(14)	E	1110
7	0111	(15)	F	1111

What about converting between Octal and Hexadecimal?

Octal to Hexadecimal:

Convert Octal to Binary and then convert Binary to Hexadecimal.

Hexadecimal to Octal:

Convert Hexadecimal to Binary and then convert Binary to Octal.

Chapter 3
Data Representation

Data and computers – Ch. 3.1

- everything inside a computer is stored as patterns of 0s and 1s.
- numbers, text, audio, video, images, graphics, etc.
- how do you convert it to 0s and 1s.
- how do you store the 0s and 1s efficiently.

Representing Numeric data – Ch. 3.2

Representing Natural numbers with a finite number of digits

General Property

Number of digits	Min	Max
n	0	$b^n - 1$

Examples:

b=10, n=3	000 to 999	
b=2, n=3	000 to 111	(equivalent to 0 to 7, in decimal)
b=8, n=3	000 to 777	(equivalent to 0 to 511, in decimal)
b=16, n=3	000 to FFF	(equivalent to 0 to , in decimal)

Representing Negative Number (Integers)

Basic Definition

- an integer is a number which has no fractional part

Examples: `+123` `-67`

Integers (Signed, natural numbers)

- previously: unsigned numbers only, i.e. natural numbers
- need mechanism to represent both positive and negative numbers
- two schemes: 1) sign-magnitude, 2) complementary representation

Sign-Magnitude

in binary:

- sign: left-most bit (0 = positive, 1 = negative)
- magnitude: remaining bits

Example: (using 6-bit sign-magnitude representation)

```
+5 = 000101        -5 = 100101
```

Ranges

	Binary			
	Unsigned (Natural Number)		Sign-Magnitude (Integer)	
Number of bits	Min	Max	Min	Max
1	0	1		
2	0	3	–1	+1
3	0	7	–3	+3
4	0	15	–7	+7
5	0	31	–15	+15
6	0	63	–31	+31
etc.				
n	0	$2^n - 1$	$-\left(2^{n-1} - 1\right)$	$+\left(2^{n-1} - 1\right)$

Difficulties with Sign-Magnitude

- two representations for zero
  `+0 = 000000        -0 = 100000`
- arithmetic is awkward, especially subtraction

Complementary Representation

- positive numbers are represented by their corresponding natural numbers
- negative numbers are represented as (very) large natural numbers
- subtraction reduces to addition

negative numbers: $-x \equiv b^n - x$

Example:

Let $b = 10$ and $n = 2$

Positive numbers	Negative numbers
(+0) 00	
(+1) 01	99 (-1)
(+2) 02	98 (-2)
(+3) 03	97 (-3)
...	...
(+49) 49	51 (-49)
	50 (-50)

Visualization: