Number system and conversions (Section 1.4 of textbook)

Vikas Dhiman for ECE275

May 8, 2024

1 Place value number system

- What is a place value number system?
- What are some examples?
- What are some non-examples?
- What is a radix (or base)?
- How to convert between different radix in place value system?
- What are some commonly used number systems in computer engineering?

1.1 Decimal number system

1.2 Binary numbers

1.3 Conversion between different radix

Problem 1 *Convert the following binary numbers to decimal:* $(11110)_2$, $(100111)_2$.

Conversion from decimal to binary The value is in decimal because we find it easy to do calculations in decimal numbers. Decimal values can be converted back to Binary representation by *repeated division* by 2 while noting down the remainder. Allow me to use / sign to denote both quotient and remainder after division. Let's convert $(22)_{10}$ back to binary:

$$\begin{aligned}
22/2 &= (11, 0) && \text{11 is the quotient and 0 is the remainder} \\
11/2 &= (5, 1) && \text{5 is the quotient and 1 is the remainder} \\
5/2 &= (2, 1) \\
2/2 &= (1, 0) \\
1/2 &= (0, 1)
\end{aligned}$$

Read the remainders from bottom to top and right them as left to right, to form the resultant binary number $(22)_{10} = (10110)_2$.

Problem 2 *Find the binary representation for decimal numbers:* 123 *and* 89*. Show your work.*

2 Hexadecimal numbers

Numbers with base 16 are called Hexadecimal numbers. From 0 to 9 the symbols are same as decimal numbers. From 10 to 15, Hexadecimal numbers use A to F.

$$A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$$

. Example, $(10AD)_{16} = 1 \times 16^3 + 10 \times 16^1 + 13 = 4096 + 160 + 13 = 4269$.

3 Octal numbers

Numbers with base 8 are called octal numbers. Example, $(354)_8 = 3 \times 8^2 + 5 \times 8 + 4 = 192 + 40 + 4 = 236$.

4 Hexadecimal/octal to binary and vice-versa

Normally, if you have to convert between a number of base r_1 to a number of base r_2, we will have to convert it via decimal numbers. Convert from base r_1 to decimal and then from decimal to r_2.

Since Hexadecimal base 16 is an exact power of 2 ($16 = 2^4$). Conversion between Hexadecimal to binary is easy. You can group 4 binary digits from right to left and convert each group of 4 binary digits to a single Hexadecimal digit and back. Example, $(10110)_2 = (0001\_0110)_2 = (16)_{16}$. To convert back. Take example, $(10AD)_{16} = (0001\_0000\_1010\_1101)_2 = (1\_0000\_1010\_1101)_2$.

Problem 3 *Find the binary and decimal values of the following Hexadecimal numbers* $(A25F)_{16}$*,* $(F0F0)_{16}$*.*

Similarly octal to binary can proceed by grouping 3-binary digits at a time. Example, $(354)_8 = (011\_101\_100)_2$.

Problem 4 *Find the binary and decimal values of the following Octal numbers* $(3751)_8$ *and* $(722)_8$*.*

5 Signed binary numbers

Signed numbers include both negative and positive numbers. There three common signed number representations

1. Sign magnitude representation
2. One's complement
3. Two's complement

5.1 Sign-magnitude representation

The Most significant (left most) *bit* (binary digit) represents sign ($0 = +$ and $1 = -$), the rest represent the magnitude. Example, a 5-bit number $(11010)_2$ in signed magnitude representation has the value of $(-1010)_2 = -10$. Note that $+10$ has to be represented by a leading 0 at the most significant bit (MSB) $+10 = (01010)_2$. Hence, the number of bits have to be specified.

Problem 5
- *Write down all possible 4-digit binary numbers and corresponding decimal values if they are in signed magnitude format? What is the minimum and maximum value?*
- *What is the minimum and maximum value of n-digit signed binary number in sign-magnitude format?*

5.2 One's complement negation

You can convert a positive number (say +10) to negative number by applying a negative sign in front of it (-(+10) = -10). It is more evident from taking negative of a negative number (-(-10) = +10). In case of sign-magnitude representation, the "negative operator" flips the sign bit. The next two signed number representations (1's complement and 2's complement) are designed around specific negative operator definitions.
Negate $13_{10} = 01101_2$ using 5-bit one's complement.

Negate -13_{10} using 5-bit one's complement.

5.3 One's complement binary numbers

In one's complement representation, the negative operation is obtained by flipping all the bits of the binary number. Example, a 5-bit one's complement of $+10 = (01010)_2$ is $(10101)_2 = -10$. Note that flipping bits is equivalent to subtracting the number from $(11111)_2$, hence the name. You can also confirm that double negative operator yields back the same number.

Problem 6

- *Write down all possible 4-digit binary numbers and corresponding decimal values if they are in sign magnitude format? What is the minimum and maximum value?*
- *What is the minimum and maximum value of n-digit signed binary number in one's complement?*

Problem 7 *Determine the decimal values of the following 1's complement 6-digit binary numbers :*

1. *01101110*
2. *10101101*

Problem 8 *Convert the decimal numbers -17 and +23 into the 6-digit one's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.*

5.4 Two's complement negation

In two's complement representation, the n-digit negative number is obtained by subtracting the positive number from 2^n. Example, two's complement of 5-digit binary number $+10 = (01010)_2$ is $2^5 - 10 = 22 = (11000)_2$. An easier algorithm to get two's complement goes via one's complement. Note that $(11111)_2 = 2^5 - 1$. We can get two's complement by adding 1 to one's complement. To get two's complement:

1. Flip all the bits. (Same as taking one's complement).
2. Add 1 to the number.

Negate $13_{10} = 01101_2$ using 5-bit two's complement.

Negate -13_{10} using 5-bit two's complement.

How to convert one's complement number representation into sign-magnitude numbers?

1. Check if the number is positive or negative. Even for one's complement representation, or two's complement representation, if the MSB (Most-significant bit) is 1, then the number is negative, otherwise positive.
2. If positive: For positive numbers, two's complement, one's complement and sign magnitude are the same. No conversion between different representation is needed. 2.b If negative: For negative numbers. Flip the bits of 1's complement. Once you flip the 1's complement bits of a negative number, you get the corresponding positive number.
3. We still want to represent the original negative number. So we set the MSB of sign-magnitude representation to 1. Since the range (min and max) for both n-bit 1's complement and sign-magnitude are the same (between $-(2^{n-1} - 1)$ and $2^{n-1} - 1$), you can always represent 8-bit 1's complement numbers with needing to extend the 8-bit number to 9-bits.

Example: Convert 8-bit one's complement 10101010 to 8-bit sign-magnitude Let number n = 10101010

1. Is the number +ve or -ve: It is negative because it starts with 1.
2. The number is not positive.
3. Take the 1's complement of the negative number to get the positive part. i.e. Flip the bits:
   -n = 01010101 or n = -(01010101)
4. We got the positive part of the number, but we want to represent the original negative number, so we set the MSB bit one. Hence, the equivalent sign-magnitude representation is:
   n = 11010101

5.5 Two's complement representation

Problem 9 *Determine the decimal values of the following 2's complement 6-digit numbers :*

1. *01011110*
2. *10010111*

Problem 10 *Convert the decimal numbers -17 and +23 into the 6-digit two's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.*

Problem 11 *Convert the decimal numbers 73, 23, -17, and -163 into signed 8-bit numbers in the following representations:*

1. *Sign and magnitude*
2. *1's complement*
3. *2's complement*

5.6 Arithmetic overflow

Problem 12 *Consider addition of 4-digit two's complement binary numbers*

1. $1010_2 + 1101_2$
2. $1011_2 + 1100_2$

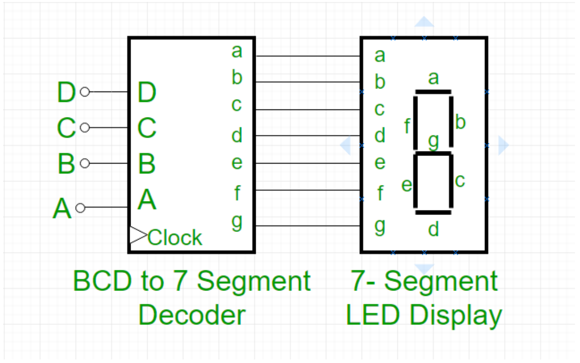
In which of the two case overflow happens? Can you come up with a rule to "easily" detect overflow?

5.6.1 Rules for detecting arithmetic overflow:

1. Adding numbers of different signs never produces an overflow.
2. Adding numbers of the same sign may produce an overflow
   (a) Wrong approach: Adding two negative 2's complement numbers always produces an additional carry-over 1, but that in itself isn't an overflow. An example, the range of 4-bit 2's complement numbers is between -8 to +7. Adding -3 to -4 in 2's complement is 1101 + 1100 produces an additional carry over 1. You can ignore the additional carry-over 1 to get the correct answer 1001 = -7 which is within range -8 to 7.
   (b) Approach 1: The easiest way for now to detect overflow is if adding two -ve numbers results in a +ve number, or adding +ve numbers results in a -ve number.
   (c) Approach 2: You can also do a range test in decimal based range test. The range of n-bit 2's complement numbers is between -2^{n-1} and $2^{n-1}-1$. For 5-bit 2's complement numbers, it is between -16 and 15. For 6-bit 2's complement numbers, it is between -32 and 31.
   (d) Approach 3: You can also check the carry-overs of the most significant two bits. If they match, i.e. 0 and 0, or 1 and 1, then there is no overflow. If they do not match, i.e. 0 and 1 or 1 and 0, then there is an overflow.

6 Binary coded decimal

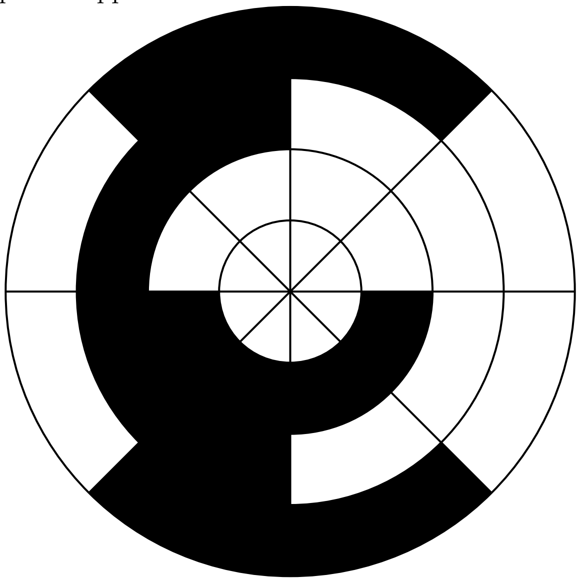
In Binary coded decimal (BCD), each decimal digit is represented by 4 bits. For example, 1047 = $(0001\_0000\_0100\_0111)_{\text{BCD}}$. It is useful in input-output applications where the number has to be either displayed as decimal or received as decimal.



Problem 13 *Convert 11, 23, 35, 57 and 103897 to BCD?*

7 Gray code

A sequence of binary numbers where only one bit changes when the number increases by 1. It is helpful in applications like wheel encoders



Problem 14 *Write all possible 3-bit binary numbers in gray-code*