Data Representation in Digital Systems

VI Signed Integers

Facts to consider

- Number of bits, n, must always be defined
  – 2^n different numbers
- Backward compatibility in representations
  – Non negative numbers must have the same representation as the unsigned case
- No need of extra hardware
  – The same adder/subtractor (using two's complement addition) is used
  – Represent A and –A, in such a way to guarantee that A+(-A)=0 using existing hardware for unsigned numbers.

Generating the set (1): a) Backward compatibility with nonnegative representations

General : n-bits

- 2^n numbers :
  - "0" : 000….00
  - 2^n-1 words to be assigned.
  - Half ([2^(n-1)-1]) to positive numbers and a half to respective negative ones:

  (+/-)1, (+/-)2, …. (+/-)(2^{n-1}-1)

  - positves: 000..01, 000..10, … up to 011111…11
- One word is yet to be assigned:

  Either +2^{n-1} or -2^{n-1}

Example: 4 bits

- 16 numbers:
  - "0": 0000
  - Still 15 words to be assigned:
  - Seven positive: 1, 2, … 7 and seven negative: -1, -2, … -7
  - +1: 0001 -1: ?
  - +2: 0010 -2: ?
  - +3: 0011 -3: ?
  - +4: 0100 -4: ?
  - +5: 0101 -5: ?
  - +6: 0110 -6: ?
  - +7: 0111 -7: ?
- One word left, to either +8 or -8

Generating the set (2):
a) Backward compatibility with hardware for unsigned numbers

1. In existing hardware, X-W = Z is equivalent to X+ (2's complement of W) = Z (disregarding carry/borrow)

2. In the particular case of X= 0, this yields -W = Z and also (2's complement of W) = Z

3. **Therefore, backward compatibility is guaranteed if representations of W and of -W are two's complements of each other**

4. This principle also has as a consequence that automatically **X – W = X + (-W)**

5. 1000….00 is its self two's complement. We assign it to -2^{n-1}. 2^{n-1} is not existent in the set

0: 0000	
1: 0001	-1: 1111
2: 0010	-2: 1110
3: 0011	-3: 1101
4: 0100	-4: 1100
5: 0101	-5: 1011
6: 0110	-6: 1010
7: 0111	-7: 1001
	-8: 1000

Power expansion for signed numbers in two's complement representation

1. Non negative numbers have the same normal binary meaning, hence:

$$0 b_{n-2} b_{n-3} \cdots b_1 b_0 \rightarrow 2^{n-2} b_{n-2} + 2^{n-3} b_{n-3} + \cdots + 2^1 b_1 + 2^0 b_0$$

2. The n-bit word b_{n-1}0000…000 represents

$$b_{n-1} 00 \cdots 00 \rightarrow \begin{cases} 0 & if\ b_{n-1} = 0 \\ -2^{n-1} & if\ b_{n-1} = 1 \end{cases} = -2^{n-1} b_{n-1}$$

3. Therefore, if $b_{n-1} b_{n-2} \ldots\ b_1\ b_0$ represents the signed decimal A, then

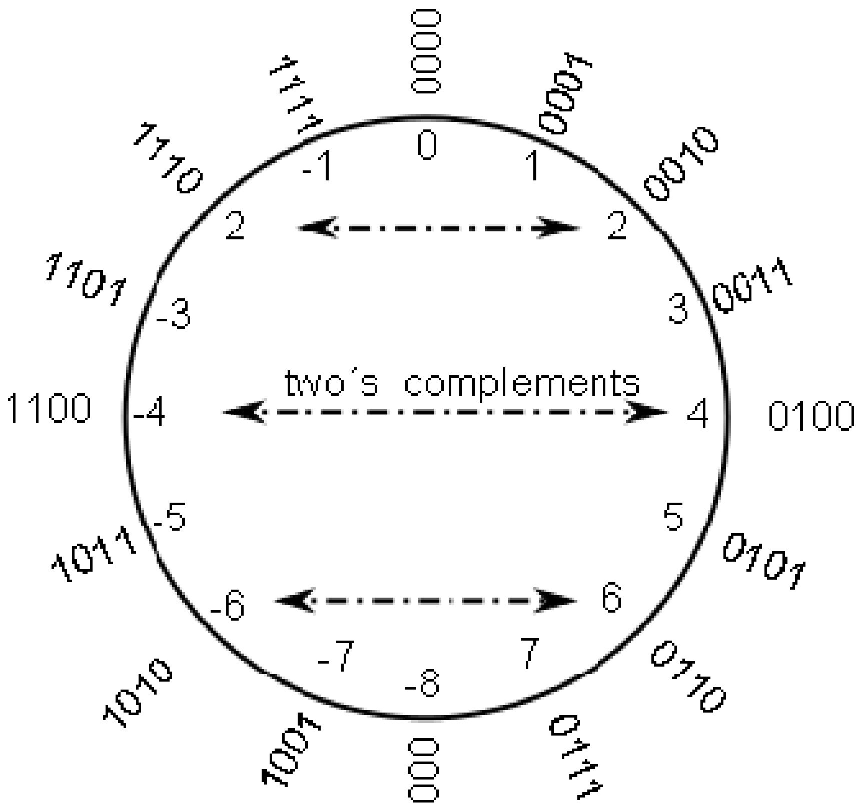
$$\boxed{A = -2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + 2^{n-3} b_{n-3} + \cdots + 2^1 b_1 + 2^0 b_0}$$

Two's complement convention for signed numbers representation

- With n bits, the interval that can be represented is
  $[-2^{n-1}, 2^{n-1}-1]$
- Representation for numbers A and –A are 2's complement of each other
- The msb of negative numbers is 1 and of nonnegative numbers is 0
- The signed decimal equivalent for
  $b_{n-1}b_{n-2} \dots b_1b_0$ is

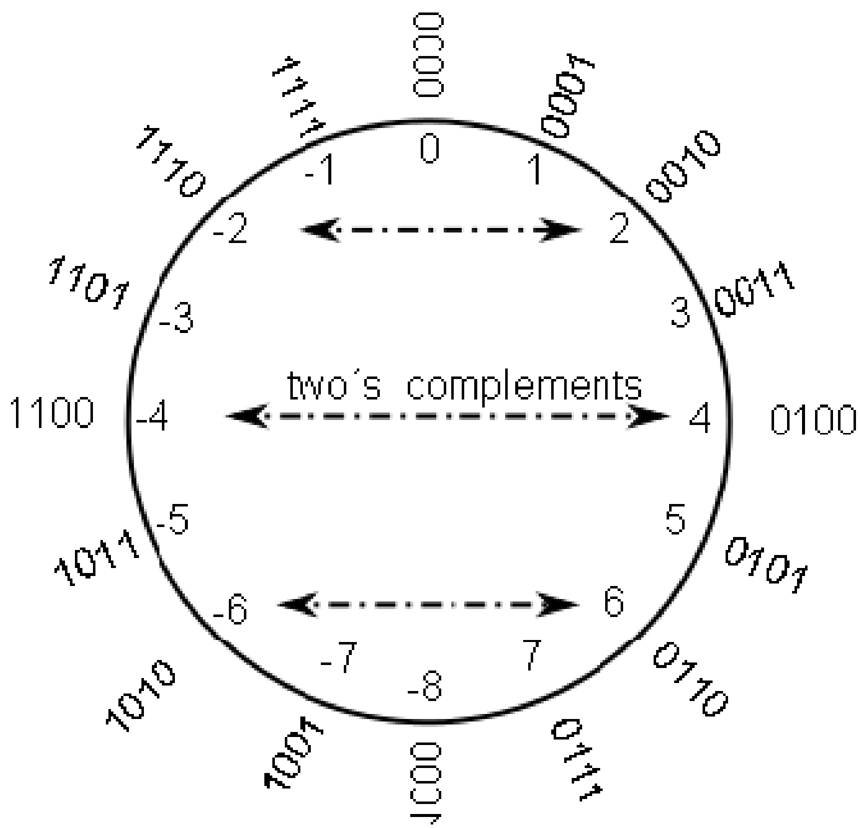
$$\boxed{A = -2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + 2^{n-3}b_{n-3} + \cdots + 2^1 b_1 + 2^0 b_0}$$

Numerical Circle: Signed representations



- Digital words are the same in same order
  - Interpretation changes
  - All nonnegative numbers start with 0, negatives with 1.
- <u>Definition</u>: **Sign Bit** is the most significant bit
  - **0 if nonnegative**
  - **1 if negative**

Signed Numerical Circle
Additions and subtractions (1)



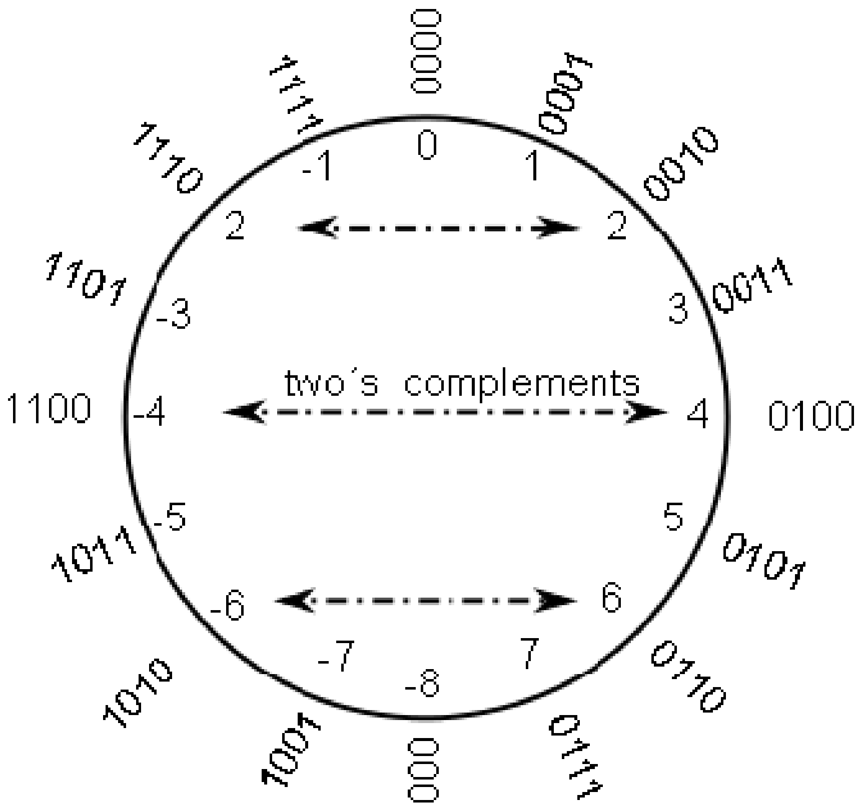
+2 +	0010 +	-2 +	1110 +
+5 =	0101 =	-5 =	1011 =
+7	0111	-7	1 1001

+ 3 +	0011 +	- 3 +	1101 +
- 6 =	1010 =	+ 6 =	0110 =
- 3	1101	+ 3	1 0011

- 1 -	1111 -		1111 +
+ 4 =	0100 =	➔	1100 =
- 5	1 011		1 1011

+ 7 -	0111 -		0111 +
+ 5 =	0101 =	➔	1011 =
+ 2	0010		1 0010

Signed Numerical Circle
Overflow in Additions and subtractions



+2 +	0010 +	- 4 +	1100 +
+6 =	0110 =	- 5 =	1011 =
- 8	1000	+7	1 0111

- 5 -	1011 -		1011 +
+ 4 =	0100 =	➔	1100 =
+ 7	0 111		1 0111

+ 3 -	0011 -		0011 +
- 6 =	1010 =	➔	0110=
- 7	1 1001		1001

Definition: overflow

- **Overflow:** In addition and subtraction operations, overflow occurs if in the n-bit result
  - Addition of two numbers with equal sign yields a result with different sign: (0100 + 0100 = 1000)
  - Subtraction where numbers have different sign, result has the sign of the minuend.
- Notes:
  - Numbers with different signs in addition cannot cause overflow
  - Numbers with similar signs in subtraction cannot cause overflow
  - The operation is invalid if limited to th n-bits, but it is generally valid if carry or borrow are included.

Definition: Bit Extension

- **To extend an n-bit signed representation of A to m>n bits, just add to the left the necessary number of bits, all equal to the sign bit**
  - Example: + 7 → 0111 then +7 → 00000111
  - Example: - 7 → 1001, then -7 → 11111001

Bit extension: demonstration

- We basically need to show that if A is represented with n bits as $b_{n-1}b_{n-2} \ldots. b_1b_0$ then the n+1-bit word $b_nb_{n-1}b_{n-2} \ldots. b_1b_0$ represents A if $b_n=b_{n-1}$.
  - <u>Solution:</u> We use the power expansion. If the bits are 0, equality follows. If both bits are 1, then

$$-2^n + 2^{n-1} + 2^{n-2}b_{n-2} + 2^{n-3}b_{n-3} \ldots + 2^1 b_1 + b_0 =$$

$$(-2+1)2^{n-1} + 2^{n-2}b_{n-2} + 2^{n-3}b_{n-3} \ldots + 2^1 b_1 + b_0 =$$

$$-2^{n-1} + 2^{n-2}b_{n-2} + 2^{n-3}b_{n-3} \ldots + 2^1 b_1 + b_0 = A$$

Two Bit extension exercises (1)

- Exercise 1: Working in hex notation, express -9, with 5 bits, 6 bits, 8 bits, 11 bits and 16 bits:
  - Solution: 17h, 37h, F7h, 7F7h, FFF7h
- Exercise 2: Certain system can only work with either bytes or 16-bit words. A programmer needs to add more than 2 (and less then 256) bytes. What steps should be taken before addition and why?

Two Bit extension exercises (2)

- Exercise 2: Certain system can only work with either bytes or 16-bit words. A programmer needs to add more than 2 (and less then 256) bytes. What steps should be taken before addition and why?
  - Solution: Before addition, each byte must be converted to a 16-bit representation of the same number, because result requires more than 8-bits. Therefore, if the numbers are unsigned, byte X1X0 should be extended to 00X1X0. But for signed numbers, the byte must be signed-extended to 16 bits

Data Representation in Digital Systems

VII Real numbers: Fixed Point

Definition: Format F(p,q)

- If an n-bit word is used to represent real numbers in **fixed-point format F(p,q)**, p+q=n,
  - Most significant p bits → integer part
  - Least significant q bits → fractional part.
- This is valid for both signed and unsigned numbers
- Example 1: the 9-bit word 110110101 ←→ 1B5h, <u>unsigned</u>
  - In format F(5.4) means 11011.0101b = 27.3125
  - In format F(3.6) means 110.110101b = 6.828125
  - In format F(4.5) means 1101.10101b = 13.65625

Format F(p.q): Power expansion

$$\overbrace{b_{(p+q-1)}b_{(p+q-2)}\cdots b_{(q+1)}b_q}^{\text{Integer part: p bits}}\ \overbrace{b_{(q-1)}\cdots b_1 b_0}^{\text{Fractional: q bits}}$$

$$A = \pm 2^{(p+q-1)} b_{(p+q-1)} + 2^{(p+q-2)} b_{(p+q-2)} + \cdots + 2b_{(q+1)} + b_q + \cdots$$
$$+ 2^{-1} b_{q-1} + 2^{-2} b_{q-2} + \cdots + 2^{-q} b_0$$

Sign + for unsigned numbers

Sign – for signed numbers

, *<u>Hint for easy reading</u> :*
eliminate q from sub indices
and exponents in the sum,
(integer part only) and
it will look like a normal
expansion for a positional number

Example: 110101

Format F(3.3)

*** Note: 6.625 + 1.375 = 8 = 2^3

Unsigned	Weights:	4	2	1	0.5	0.25	0.125	→	**6.625**
		1	**1**	**0**	**1**	**0**	**1**		
Signed	Weights:	-4	2	1	0.5	0.25	0.125	→	**-1.375**

Format F(2.4)

*** Note: 3.3125 + 0.6875= 4 = 2^2

Unsigned	Weights:	2	1	0.5	0.25	0.125	0.0625	→	**3.3125**
		1	**1**	**0**	**1**	**0**	**1**		
Signed	Weights:	-2	1	0.5	0.25	0.125	0.0625	→	**-0.6875**

Theorem and consequence:

If A is the decimal (signed or unsigned) for an n-bit word in format F(p.q), and D is the "normal binary" integer (signed or unsigned) for the same word, then

$$A = D/2^{q}$$

PROOF: Multiply the power expansion of A by 2q (Do it!!)

Interval for n-bit words in format F(p.q):

<u>Unsigned case:</u> 0 to $(2^{n}-1)/2^{q}$

<u>Signed case:</u> $-2^{(n-1)/}2^{q}$ to $(2\wedge^{n-1}-1)/2^{q}$

Definition: STEP

- STEP: In format F(p.q), the step is 2^{-q}
  - In F(2.3) the step is $2^{-3} = 0.125$
  - In F(1.4) the step is $2^{-4} = 0.0625$
- The step is the difference between two consecutive numbers
- The larger q, more dense is the interval.

Examples with three bits:

Unsigned Intervals:

Integers: 0 to 7 (Step 1)
F(2.1): 0 to 7/2 = 3.5, Step 0.5
F(1.2): 0 to 7/4 = 1.75, Step 0.25
F(0.3) : 0 to 7/8 = 0.875, Step 0.125

3-bit word			F(2.1)	F(1.2)	F(0.3)
0	0	0	0.0	0.00	0.000
0	0	1	0.5	0.25	0.125
0	1	0	1.0	0.50	0.250
0	1	1	1.5	0.75	0.375
1	0	0	2.0	1.00	0.500
1	0	1	2.5	1.25	0.625
1	1	0	3.0	1.50	0.750
1	1	1	3.5	1.75	0.875

3-bit word			F(2.1)	F(1.2)	F(0.3)
1	0	0	-2.0	-1.00	-0.500
1	0	1	-1.5	-0.75	-0.375
1	1	0	-1.0	-0.50	-0.250
1	1	1	-0.5	-0.25	-0.125
0	0	0	0.0	0.00	0.000
0	0	1	0.5	0.25	0.125
0	1	0	1.0	0.50	0.250
0	1	1	1.5	0.75	0.375

Signed Intervals:

Integers: -4 to 3 (Step 1)
F(2.1): -2 to 3/2 = 1.5, Step 0.5
F(1.2): -1 to 3/4 = 0.75, Step 0.25
F(0.3) : -0.5 to 3/8 = 0.375, Step 0.125

Advantages and Disadvantages of Fixed Point representation

- Advantages
  - Same hardware for addition (operate like integers)
  - Multiplication can be adjusted easily
  - Signed numbers follow same principles as integers
- Disadvantages
  - Only small intervals
  - Not good for “very small” or “very large” numbers

Biasing representations

- The concept of bias (offset) can be applied to any situation:
  - If there are 2n elements in an interval [a,b] and there is a unsigned representation –normal binary or Fixed point F(p.q) - in an interval [0,M] of the same length and step size between elements, bias or offset can be applied such that if N is in [a,b], then N+a = D, with D in [0,M] and the word for D is used for N
  - Similarly for signed representations.

Data Representation in Digital Systems

VIII Continuous (Analog) Intervals