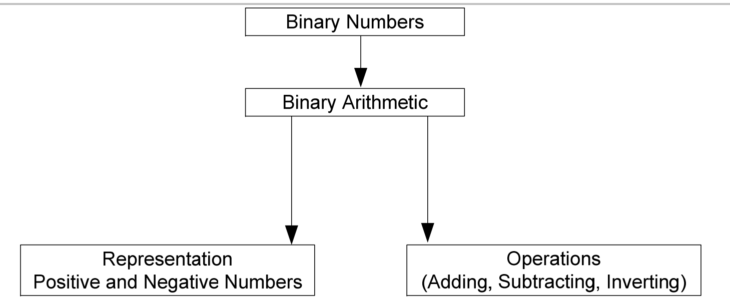
Digital Arithmetic

Digital Arithmetic: Operations and Circuits

Dr. Farahmand

Binary Arithmetic

- Digital circuits are frequently used for arithmetic operations
- Fundamental arithmetic operations on binary numbers and digital circuits which perform arithmetic operations will be examined.
- Binary numbers are added like decimal numbers.
- In decimal, when numbers sum more than 9 a carry results.
- In binary when numbers sum more than 1 a carry takes place.
- Addition is the basic arithmetic operation used by digital devices to perform subtraction, multiplication, and division.



If the numbers are unsigned and positive add them as follow:

1 + 0 = 1 with carry of 0
1 + 1 = 0 with carry of 1
1 + 1 + 1 = 1 with carry of 1

For example:
011.011 +
010.110 =
110.001 = 6.125 base 10

Adding Unsigned Numbers

- Examples:

X	190	10111110
Y	141	10001101

X	127	1111111
Y	63	111111

X	170	10101010
Y	85	1010101

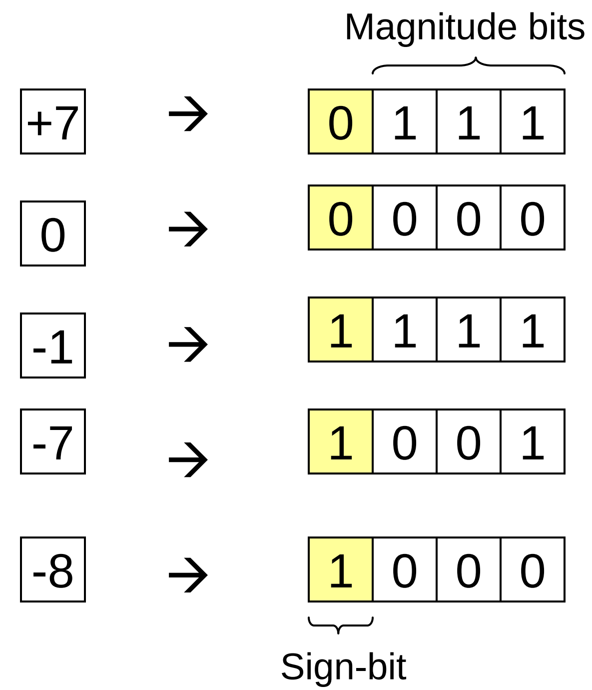
Adding Unsigned Numbers

- Examples:

X	190	10111110
Y	141	10001101
X+Y	331	101001011
X	127	1111111
Y	63	111111
X+Y	190	10111110
X	170	10101010
Y	85	1010101
X+Y	255	11111111

Representing Signed Numbers

- Since it is only possible to show magnitude with a binary number, the sign (+ or -) is shown by adding an extra “sign” bit.
- A sign bit of 0 indicates a positive number.
- A sign bit of 1 indicates a negative number.
- The 2’s complement system is the most commonly used way to represent signed numbers.



Examples of 2’s complement representation

-8	+4	+2	+1
-2^3	+2^2	+2^1	+2^0

Representing Signed Numbers

Converting to 2’s Complement

- In order to change a binary number to 2’s complement it must first be changed to 1’s complement.
  - To convert to 1’s complement, simply change each bit to its complement (opposite).
  - To convert 1’s complement to 2’s complement add 1 to the 1’s complement.
- A positive number is true binary with 0 in the sign bit.
- A negative number is in 2’s complement form with 1 in the sign bit.
- A binary number can be negated by taking the 2’s complement of it.

For example:

+9 → **0**1001 (sign bit = 0, indicating +)

2’s complement of 9 →

01001 → 10110

10110 + 1 → **1**0111 = -9

This is
The sign
BIT

When the sign-bit is zero → Positive number

If the sign-bit is set → Negative number

Remember: 2’s complement is just a conventional way of representing signed numbers in digital arithmetic – Don’t ask why!

2's Complement Representation

- Assuming **N+1** bits representing a 2's Complement (that is representing the number with N bits and one bit is dedicated to indicate the sign):
  - Largest positive number will be **2^N-1**
  - Smallest signed number (largest negative number) will be **-2^N**
  - Total numbers (including zero) that can be represented will be **2^{N+1}**

For example:
Assume 3+1 bits
Largest pos. number will be **0111= +7**
Smallest number will be **1000 = -8**
Remember: **1111= -1**
Zero is represented by **0000 = Zero**

For example:
Assume 6+1 bits
Largest pos. number will be ??
Smallest signed number will be ??
Zero is represented by ??

0 111 111 = 63
1 000 000 = -64
0 000 000 = 0

More Examples

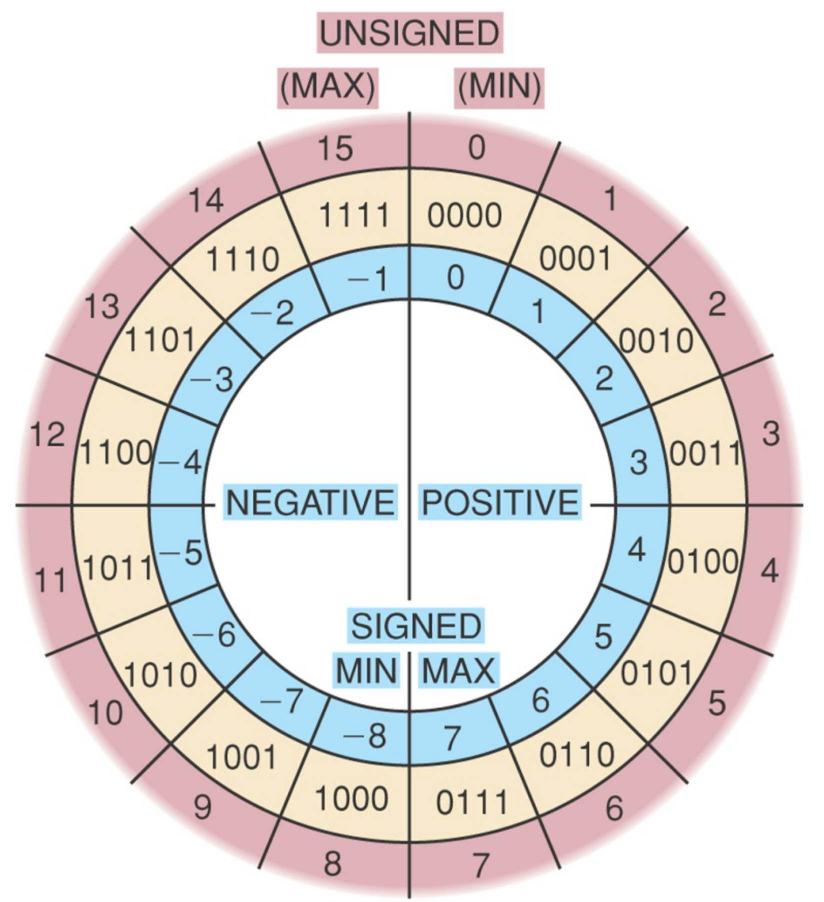
Integer Signed	2's Complement
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Integer		2's Complement
Signed	Unsigned	
5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

Remember: Always know how many bits are provided!

2's Complement Representation

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



Arithmetic Operations using 2's Complement

- Inverting
  - A positive number to a negative number
  - A negative number to a positive number
    - **Either case just take the 2's complement**
- Adding A and B→ A+B (assuming N bits represent the magnitude and one bit is dedicated as the sign-bit)
- Subtracting B from A → A-B
  - Just take the 2's Complement of B
  - Add A and B (A+B)

Magnitude Overflow: **(max unsigned number that can be represented using 8 bits is 255)**	X	10110100
	Y	01010101
	X+Y	100001001
The carry will be ignored	X	0110
	Y	1101
	X-Y	1 0011
Note the value is negative	X	0100
	Y	1001
	X-Y	1101
Overflow: **(max signed number that can be represented using 4 bits is -8)**	X	1101
	Y	1010
	X+Y	1 0111
Overflow: **(max signed number that can be represented using 4 bits is 7)**	X	0 111
	Y	0 111
	X+Y	1 110

Arithmetic Operations using 2's Complement

- Inverting
  - A positive number to a negative number
  - A negative number to a positive number
    - **Either case just take the 2's complement**
- Adding A and B→ A+B (assuming N bits represent the magnitude and one bit is dedicated as the sign-bit)
- Subtracting B from A → A-B
  - Just take the 2's Complement of B
  - Add A and B (A+B)
- NOTE: When a Positive and a Negative number are added together, the overflow will be a signed overflow and it is ok!

The overflow can be discarded when it is sign overflow (max = -16):

X	-4		11100
Y	-9		10111
X+Y	-13		1 10011

Magnitude Overflow:
(max unsigned number that can be represented using 8 bits is 255)

X	180	10110100
Y	85	01010101
X+Y	265	100001001

The carry will be ignored

X	6	0110
Y	-3	1101
X-Y	3	1 0011

Note the value is negative

X	4	0100
Y	-7	1001
X-Y	-3	1101

Overflow:
(max signed number that can be represented using 4 bits is -8)

X	-3	1101
Y	-6	1010
X+Y	-9	1 0111

Overflow:
(max signed number that can be represented using 4 bits is 7)

X	7	0 111
Y	7	0 111
X+Y	14	1 110

BCD Addition

- BDC numbers
  - They are between 0 and 9
  - Hence each decimal number is represented by 4 bits
- Add each number between 0-9 individually

- 0 1 1 0 ← BCD for 6

  0 1 1 1 ← BCD for 7

  ---

  1 1 0 1 ← 13 but invalid!

  0 1 1 0 ← Add 6 to correct

  ---

  0 0 0 1 0 0 1 1 ← BDC for 13!!

BCD Addition – Another Example

- BDC numbers
  - They are between 0 and 9
  - Hence each decimal number is represented by 4 bits
- Add each number between 0-9 individually

```
          1
0 1 0 1   1 0 0 1 ← BCD for 59
0 0 1 1   1 0 0 0 ← BCD for 38
_____________________
1 0 0 1   0 0 0 1 ← 91 but invalid!
          0 1 1 0 ← Add 6
__________________________
1 0 0 1   0 1 1 1 ← BDC for 97!!
```

Hex Arithmetic

- Addition of Hex numbers is similar to decimal addition
  - Remember the largest value is 16 (F) and NOT 9!
  - Carry a 1 if the number is larger than F
  - Use the following steps
    - Add the hex digits in decimal.
    - If the sum is 15 or less express it directly in hex digits.
    - If the sum is greater than 15, subtract 16 and carry 1 to the next position.

9
+ 8
In Hex=11

9
+ 8
In Decimal =17
17-16 =1
In Hex = 11_{16}

3AF
+ 23C
= 5EB

8+B=8+(11)= 19→19-16=3, Carry 1

58
+ 4B
= A3

Hex Arithmetic

- Subtraction of hex numbers follows a similar method as binary numbers
  - Make sure you take the 2's complement of the negative hex number
    - Method 1:
      - Subtract the number from FFFF…
      - Write the results in hex
      - Add one to the final answer
      - For example: 2's complement of 3A5 is:
        - **FFF-3A5=C5A → C5A+1 = C5B**
    - Method 2:
      - Convert the number to Binary
      - Take the 2's complement
      - Convert back the results into hex value
  - Simply add the values together