Jiangxi University of Science and Technology

DIGITAL DESIGN

Lecture 2
Signed Number

Binary Addition

- Single Bit Addition Table

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$ Note “carry”

Example:
Add the following binary numbers

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +0\ 1\ 1\ 1 \\ \hline \end{array}$$

- Solution: Add the following binary numbers

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \quad \text{Note “carry”} \end{aligned}$$

$$\begin{array}{r} {\scriptstyle 1\ \ 1} \\ 1\ 1\ 0\ 1 \\ +0\ 1\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 0 \end{array}$$

Binary Subtraction

- Single Bit Subtraction Table

$$0 - 0 = 0$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$
$$0 - 1 = 1 \text{ with a “borrow”}$$

Example
Subtract the following binary numbers

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1 with a “borrow”

```
  1 1 0 1
- 0 1 1 1
---------
```

```
    1   1
  0 0 0 1
- 0 1 1 1
---------
  0 1 1 0
```

Arithmetic

- Decimal

```
  11
  1234          5274
+ 5678        - 1638
------        ------
  6912          3636
```

- Binary

```
 1 1
  1011          1011
+ 1010        - 0110
------        ------
 10101          0101
```

Negative Binary Numbers

- **In decimal we are quite familiar with placing a "-" sign in front of a number to denote that it is negative**
- **The same is true for binary numbers a computer won't understand that**
- **What happens in memory then?**

2.4 Signed Integer Representation

- Example:
  - Using signed magnitude binary arithmetic, find the sum of 75 and 46.
  - First, convert 75 and 46 to binary, and arrange as a sum, but separate the (positive) sign bits from the magnitude bits.

```
0   1001011
0 + 0101110
```

- Example:
  - Using signed magnitude binary arithmetic, find the sum of 75 and 46.
  - Just as in decimal arithmetic, we find the sum starting with the rightmost bit and work left.

```
0   1001011
0 + 0101110
          1
```

2.4 Signed Integer Representation

- Example:
  - Using signed magnitude binary arithmetic, find the sum of 75 and 46.
  - In the second bit, we have a carry, so we note it above the third bit.

```
      1
  0 1001011
+ 0 0101110
-----------
    01
```

- **Example**:
  - Using signed magnitude binary arithmetic, find the sum of 75 and 46.
  - The third and fourth bits also give us carries.

```
    1 1 1
  0 1001011
+ 0 0101110
-----------
    1001
```

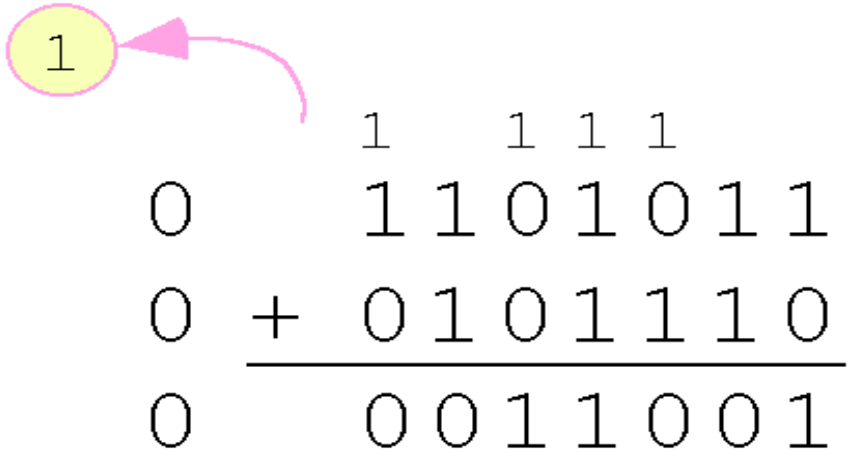
- **Example:**
  - Using signed magnitude binary arithmetic, find the sum of 75 and 46.
  - Once we have worked our way through all eight bits, we are done.

```
    1 1 1
  0 1001011
+ 0 0101110
-----------
  0 1111001
```

In this example, we were careful careful to pick two values whose sum would fit into seven bits. If that is not the case, we have a problem.

2.4 Signed Integer Representation

- Example:
  - Using signed magnitude binary arithmetic, find the sum of 107 and 46.
- We see that the carry from the seventh bit *overflows* and is discarded, giving us the erroneous result: 107 + 46 = 25.



Negative numbers

- How do we write negative binary numbers?
  - Prefix numbers with minus symbol?

- 3 approaches:
  - Sign and magnitude
  - Ones-complement
  - Twos-complement

- **Two's complement is the system used in microprocessors**
- **Most significant bit becomes important**

- All 3 approaches represent positive numbers in the same way

Signed Magnitude

Represent the decimal number as binary

Left bit (MSB) used as the sign bit

Only have 7 bits to express the number

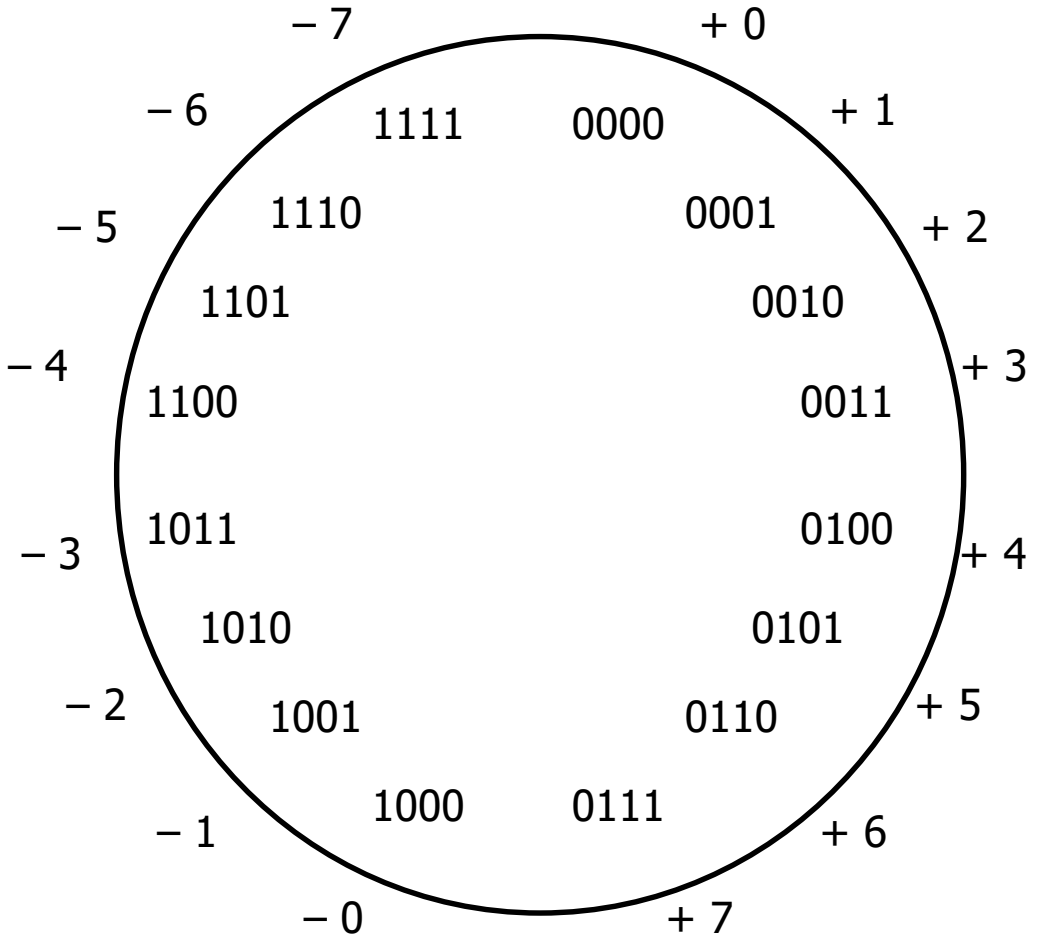
12_{10} = 00001100

-12_{10} = 10001100

How many representations are there for zero?

Sign and magnitude

- Most significant bit (MSB) is the sign bit
  - 0 ≡ positive
  - 1 ≡ negative
- Remaining bits are the number's magnitude



Sign and magnitude

- Problem 1: Two representations of for zero
  - +0 = 0000 and also –0 = 1000
- Problem 2: Arithmetic is cumbersome
  - 4 – 3 != 4 + (-3)

Add		Subtract			Compare and subtract		
4	0100	4	0100	0100	– 4	1100	1100
+ 3	+ 0011	– 3	+ 1011	– 0011	+ 3	+ 0011	– 0011
= 7	= 0111	= 1	≠ 1111	= 0001	– 1	≠ 1111	= 1001

Ones-complement

Method: Invert the ones and zeros

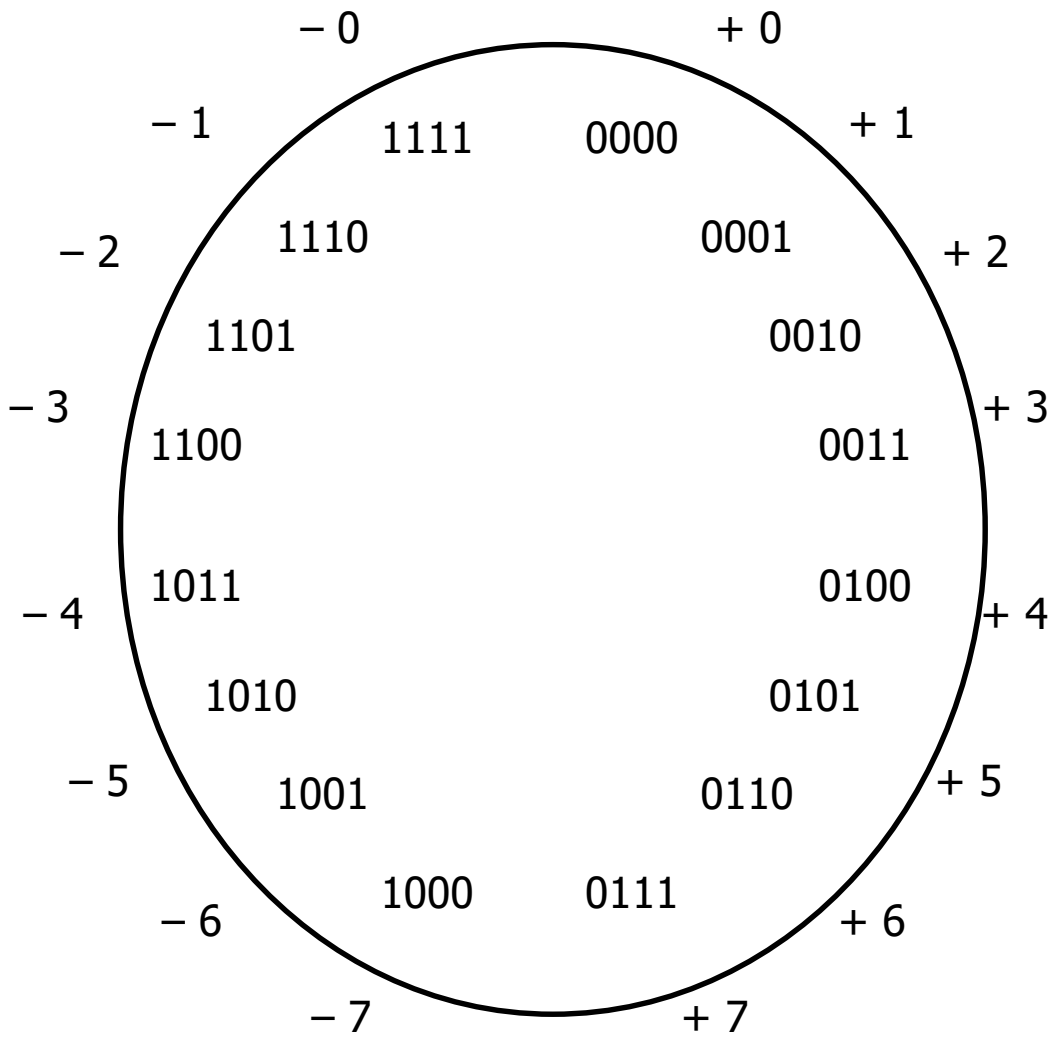
11_{10} = 00001011

-1110 = 11110100

0 in MSB implies positive

1 in MSB implies negative

- Negative number: Bitwise complement of positive number
  - $0111 \equiv 7_{10}$
  - $1000 \equiv -7_{10}$



Ones-complement

- Solves the arithmetic problem

Add		Invert, add, add carry		Invert and add	
4	0100	4	0100	– 4	1011
+ 3	+ 0011	– 3	+ 1100	+ 3	+ 0011
= 7	= 0111	= 1	1 0000	– 1	1110
		add carry:	+1		
			= 0001		

end-around carry

Why ones-complement works

- The ones-complement of an 4-bit positive number y is $1111_2 - y$
  - $0111 \equiv 7_{10}$
  - $1111_2 - 0111_2 = 1000_2 \equiv -7_{10}$

- What is 1111_2?
  - 1 less than $10000_2 = 2^4 - 1$
  - $-y$ is represented by $(2^4 - 1) - y$

Why ones-complement works

- Adding representations of x and $-y$ where x, y are positive numbers, we get $x + ((2^4 - 1) - y) = (2^4 - 1) + (x - y)$
  - If $x < y$, then $x - y < 0$. There will be no carry from $(2^4 - 1) + (x - y)$. Just add representations to get correct negative number.
  - If $x > y$, then $x - y > 0$. There will be a carry. Performing end-around carry subtracts 2^4 and adds 1, subtracting $(2^4 - 1)$ from $(2^4 - 1) + (x - y)$
  - If $x = y$, then answer should be 0, get $(2^4 - 1) = 1111_2$

So what's wrong?

- Still have two representations for zero!
  - +0 = 0000 and also –0 = 1111

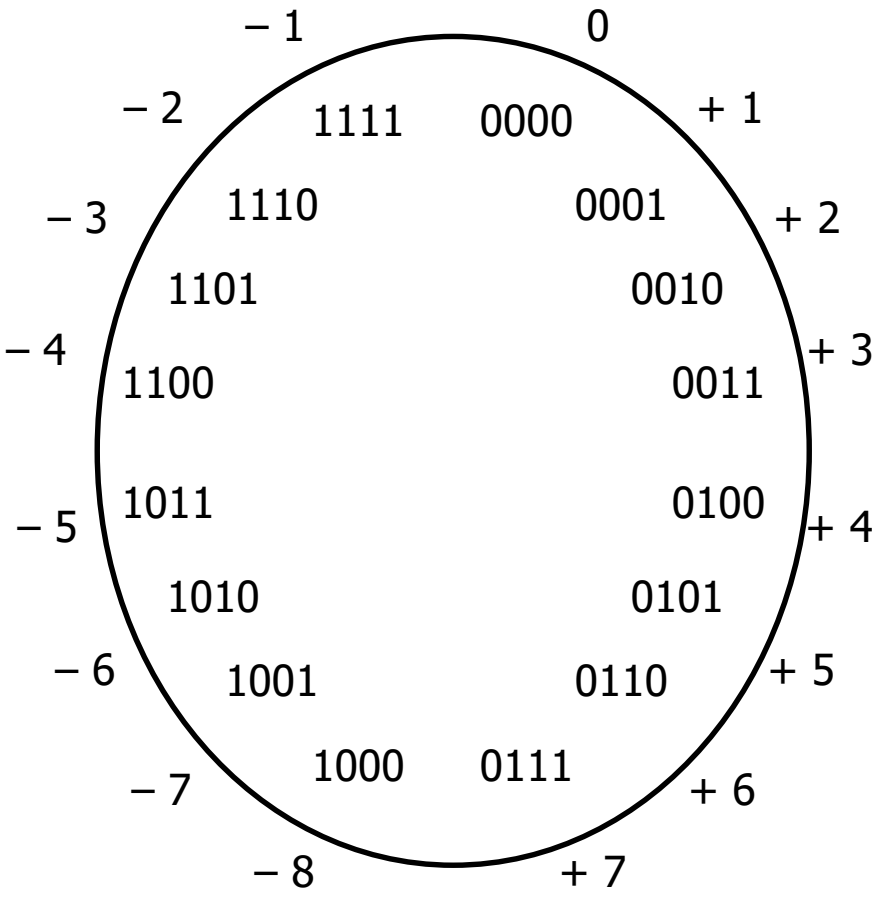
Twos-complement

- Method: Take the one's complement and add 1

  1110 = 00001011

  -1110 = 11110100 one's comp

  -1110 = 11110101 two's comp



- Negative number: Bitwise complement **plus one**
  - $0111 \equiv 7_{10}$
  - $1001 \equiv -7_{10}$
- Benefits:
  - Simplifies arithmetic
  - Only one zero!

Twos-complement

Add		Invert and add		Invert and add	
4	0100	4	0100	– 4	1100
+ 3	+ 0011	– 3	+ 1101	+ 3	+ 0011
= 7	= 0111	= 1	1 0001	– 1	1111
		drop carry	= 0001		

Why twos-complement works

- Recall: The ones-complement of a b-bit positive number y is $(2^b - 1) - y$
- Twos-complement adds one to the bitwise complement, thus, $-y$ is $2^b - y$
  - $-y$ and $2^b - y$ are equal mod 2^b (have the same remainder when divided by 2^b)
  - Ignoring carries is equivalent to doing arithmetic mod 2^b

- One representation of zero (Check)
- Enables subtraction operation by considering the addition of a positive number with a two's complement number
- Only need addition
- MSB indicates the sign of the number

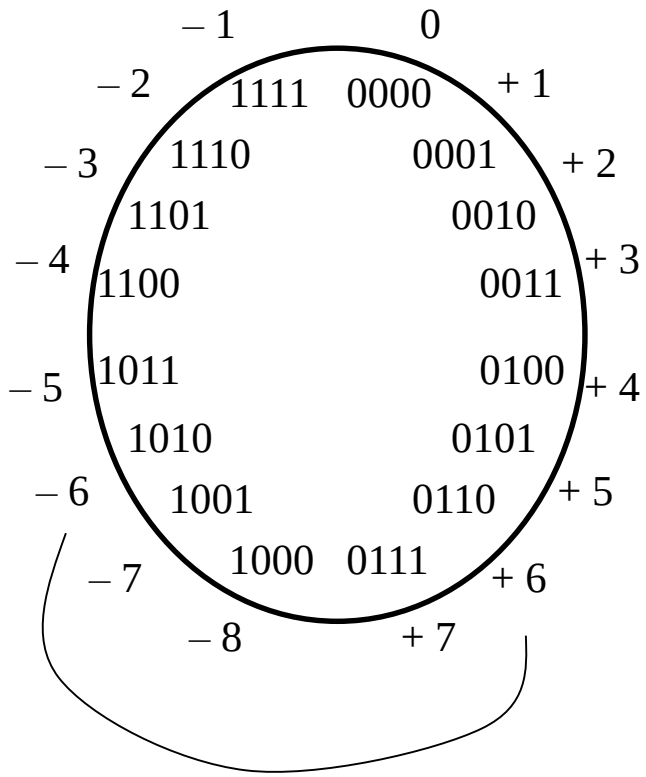
Why twos-complement works

- Adding representations of x and $-y$ where x, y are positive numbers, we get $x + (2^b - y) = 2^b + (x - y)$
  - If there is a carry, that means that $x \geq y$ and dropping the carry yields $x - y$
  - If there is no carry, then $x < y$, then we can think of it as $2^b - (y - x)$, which is the twos-complement representation of the negative number resulting from x – y.

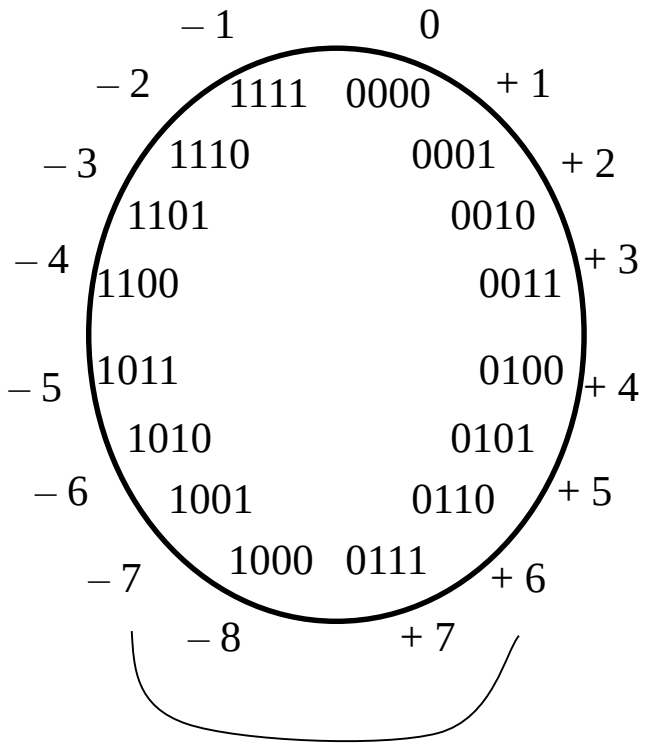
Only need one type of hardware/process to add both signed and unsigned numbers.

Overflow

- Answers only correct mod 2^b
  - Summing two positive numbers can give a negative result
  - Summing two negative numbers can give a positive result



6 + 4 ⇒ –6



–7 – 3 ⇒ +6

Miscellaneous

- Sign-extension
  - Write +6 and –6 as twos-complement
    - 0110 and 1010
  - Sign-extend to 8-bit bytes
    - **0000**0110 and **1111**1010
- Can't infer a representation from a number
  - 11001 is 25 (unsigned)
  - 11001 is -9 (sign and magnitude)
  - 11001 is -6 (ones complement)
  - 11001 is -7 (twos complement)

Compression between Complements / Sign-and-Magnitude

4 bit(signed number) Negative

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

4 bit(signed number) (positive)

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000

BCD (Binary-Coded Decimal)

Decimal Symbols	BCD Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Positional Notation

A positive number N can be written as:

$$N = (a_{n-1}a_{n-2}.....a_1a_0.a_{-1}a_{-2}.....a_{-m})_r$$

Where:

. = radix point

r = radix or base of number system

n = number of integer digits to the left of radix point

m = number of fractional digits to the right of radix point

a_i = integer digit i

a_j = fractional digit j

a_{n-1} = most significant digit

a_{-m} = least significant digit

Radix Complements Definition

Radix Complements

$$[N]_r = r^n - (N)_r$$

Where

$[N]_r$ = Radix Complement of N

$(N)_r$ = Original Number, N

r = radix or base

n = number of digits

Radix Complements Definition

Note:

$$[N]_r + (N)_r = r^n - (N)_r + (N)_r = r^n$$

Or,

N added to its complement is equal to r^n

But this is just 0, if we ignore the Most Significant Digit (MSD)

Complements(r- ۱)
Diminished-Radix

E.g.: (*r*-1)'s complement, or 9s complement, of $(22)_{10}$ is:

$(10^2 - 1) - 22 = (77)_{9s}$ [This means $-(22)_{10}$ is $(77)_{9s}$]

(*r*-1)'s complement, or 1s complement, of $(0101)_2$ is:

$(2^4 - 1) - 0101 = (1010)_{1s}$ [This means $-(0101)_2$ is $(1010)_{1s}$]

مکمل (r-1) عدد x با n رقم صحیح و m رقم اعشار برابراست با :

$(r^n - r^{-m}) - x$

$(10^2 - 1) - 22 = 99 - 22 = (77)_{9s}$

$(2^4 - 1) - 0101 = 1111 - 0101 = (1010)_{1s}$

روش نظری : همه رقم ها را از (r- ۱) کم می کنیم:

Radix Complements

Two's Complement

$$[N]_2 = 2^n - (N)_2$$

Ten's Complement

$$[N]_{10} = 10^n - (N)_{10}$$

16's Complement

$$[N]_{16} = 16^n - (N)_{16}$$

r = radix or base

n = number of digits

Summary

- Two's complement calculations
- Method 1

$$[N]_2 = 2^n - (N)_2$$

- Method 2

$$[N]_2 = [N]_1 + 1$$

- Method 3

Leave all of the least significant 0's and first 1 unchanged and then "flip" or complement the bits for all other digits

Complement

- Complement 10 of 546700

$$10^6 - 546700 = 1000000 - 546700 = 453300$$

- Complement 9 of 546700
  - $(10^6 - 1) - 546700 = 999999 - 546700 = 453299$

- Other way is Complement 9+1
  - $453299 + 1 = 453300$

Complement

- Complement 2 of 1101100

$$\mathbf{2^7 - 1101100 = (10000000 - 1101100)_2 = 0010100}$$

- Complement 1 of 1101100

$$(2^7 - 1) - 1101100 = (1111111 - 1101100)_2 = 0010011$$ •

- **Other way is complement 1+1**

$$0010011 + 1 = 0010100$$

1s Complement Subtraction/add

```
  +3      0011
+ +4    + 0100
----    -------
  +7      0111
----    -------
```

```
  +5      0101
+ -5    + 1010
----    -------
  -0      1111
----    -------
```

```
  -2      1101
+ -5    +  1010
----      ------
  -7      10111
----    +     1
            ------
            1000
```

```
  -3      1100
+ -7    + 1000
----    -------
 -10     10100
----    +     1
            -------
            0101
```

2s Complement Subtraction/add

```
  -2      1110
+ -6    + 1010
----    -------
  -8     11000
----    -------
```

```
  +3      0011
+ +4    + 0100
----    -------
  +7      0111
----    -------
```

```
  +4      0100
+ -7    + 1001
----    -------
  -3      1101
----    -------
```

```
  +6      0110
+ -3    + 1101
----    -------
  +3     10011
----    -------
```

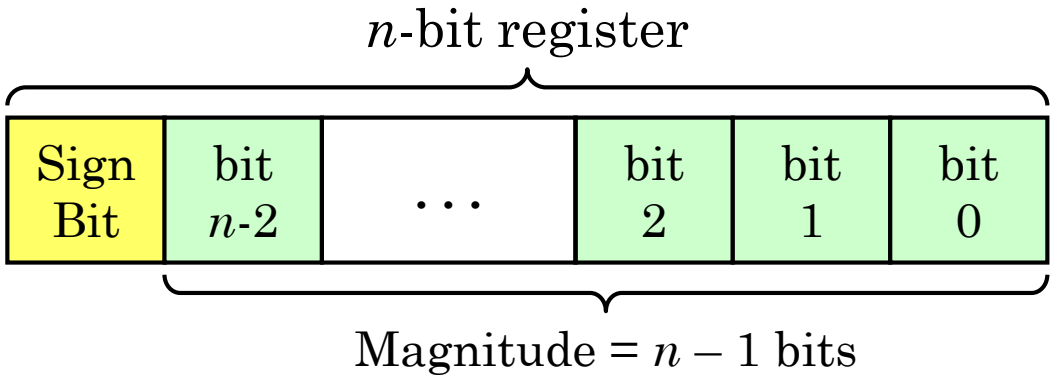
```
  +5      0101
+ +6    + 0110
----    -------
 +11      1011
----    -------
```

```
  -3      1101
+ -6    + 1010
----    -------
  -9     10111
----    -------
```

Signed Numbers

- Several ways to represent a signed number
  - Sign-Magnitude
  - 1's complement
  - 2's complement
- Divide the range of values into 2 equal parts
  - First part corresponds to the positive numbers (≥ 0)
  - Second part correspond to the negative numbers (< 0)
- The 2's complement representation is widely used
  - Has many advantages over other representations

Sign-Magnitude Representation



- Independent representation of the sign and magnitude
- Leftmost bit is the sign bit: 0 is positive and 1 is negative
- Using n bits, largest represented magnitude = $2^{n-1} - 1$

Sign-magnitude representation of +45 using 8-bit register

0	0	1	0	1	1	0	1

Sign-magnitude representation of -45 using 8-bit register

1	0	1	0	1	1	0	1

Properties of Sign-Magnitude

- Two representations for zero: +0 and -0
- Symmetric range of represented values:

  For n-bit register, range is from $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

  For example using 8-bit register, range is -127 to +127
- Hard to implement addition and subtraction
  - Sign and magnitude parts have to processed independently
  - Sign bit should be examined to determine addition or subtraction

    Addition is converted into subtraction when adding numbers of different signs
  - Need a different circuit to perform addition and subtraction

    Increases the cost of the logic circuit

Computing the 2's Complement

starting value	$00100100_2 = +36$
step1: reverse the bits (1's complement)	11011011_2
step 2: add 1 to the value from step 1	$+ \quad 1_2$
sum = 2's complement representation	$11011100_2 = -36$

2's complement of 11011100_2 (-36) = $00100011_2 + 1 = 00100100_2 = +36$

~~The 2's complement of the 2's complement of~~ N is equal to N

Another way to obtain the 2's complement:

Start at the least significant 1

Leave all the 0s to its right unchanged

Complement all the bits to its left

Binary Value

= 00100**1**00 (least significant 1)

2's Complement

= 11011**1**00

Properties of the 2's Complement

- The 2's complement of N is the negative of N
- The sum of N and 2's complement of N must be zero
  The final carry is ignored
- Consider the 8-bit number $N = 00101100_2 = +44$
  -44 = 2's complement of $N = 11010100_2$
  $00101100_2 + 11010100_2 = \mathbf{1}\ 00000000_2$ (8-bit sum is 0)

Ignore final carry

- In general: Sum of N + 2's complement of $N = 2^n$
  where 2^n is the final carry (1 followed by n 0's)
- There is only one zero: 2's complement of 0 = 0

Sign Extension

Step 1: Move the number into the lower-significant bits

Step 2: Fill all the remaining higher bits with the sign bit

- This will ensure the correctness of the signed value
- Examples
  - Sign-Extend 10110011_2 to 16 bits

$$10110011_2 = -77 \Rightarrow 11111111\,10110011 = -77$$

  - Sign-Extend 01100010_2 to 16 bits

$$01100010_2 = +98 \Rightarrow 00000000\,01100010 = +98$$

- Infinite 0's can be added to the left of a positive number
- Infinite 1's can be added to the left of a negative number

Subtraction with 2's Complement

- When subtracting A – B, convert B to its 2's complement
- Add A to (2's complement of B)

borrow -1 -1 -1 carry: 1 1 1 1

:

$$
\begin{array}{rr}
 & 0\,1\,0\,0\,1\,1\,0\,1 \\
- & 0\,0\,1\,1\,1\,0\,1\,0 \\
\hline
 & 0\,0\,0\,1\,0\,0\,1\,1
\end{array}
\quad\Rightarrow\quad
\begin{array}{rr}
 & 0\,1\,0\,0\,1\,1\,0\,1 \\
+ & 1\,1\,0\,0\,0\,1\,1\,0 \quad \text{(2's complement)} \\
\hline
 & 0\,0\,0\,1\,0\,0\,1\,1 \quad \text{(same result)}
\end{array}
$$

- Final carry is ignored, because
  - Negative number is sign-extended with 1's
  - You can imagine infinite 1's to the left of a negative number
  - Adding the carry to the extended 1's produces extended zeros

Carry and Overflow

- Carry is important when …
  - Adding or subtracting unsigned integers
  - Indicates that the unsigned sum is out of range
  - Either < 0 or >maximum unsigned *n*-bit value
- Overflow is important when …
  - Adding or subtracting signed integers
  - Indicates that the signed sum is out of range
- Overflow occurs when
  - Adding two positive numbers and the sum is negative
  - Adding two negative numbers and the sum is positive
  - Can happen because of the fixed number of sum bits

Carry and Overflow Examples

- We can have carry without overflow and vice-versa
- Four cases are possible (Examples are 8-bit numbers)

			1						
+	0	0	0	0	1	1	1	1	15
	0	0	0	0	1	0	0	0	8
	0	0	0	1	0	1	1	1	23

Carry = 0 Overflow = 0

	1	1	1	1	1				
+	0	0	0	0	1	1	1	1	15
	1	1	1	1	1	0	0	0	248 (-8)
	0	0	0	0	0	1	1	1	7

Carry = 1 Overflow = 0

		1							
+	0	1	0	0	1	1	1	1	79
	0	1	0	0	0	0	0	0	64
	1	0	0	0	1	1	1	1	143 (-113)

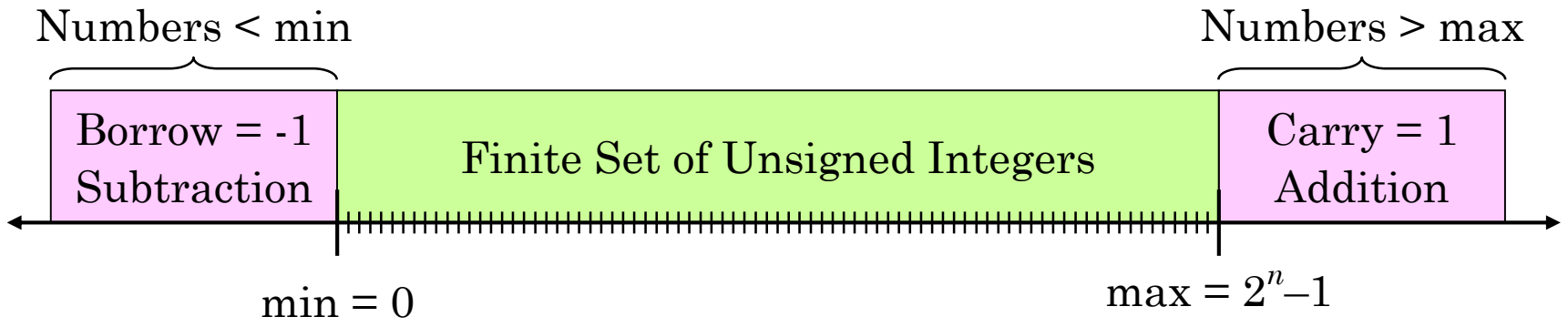
Carry = 0 Overflow = 1

	1				1	1			
+	1	1	0	1	1	0	1	0	218 (-38)
	1	0	0	1	1	1	0	1	157 (-99)
	0	1	1	1	0	1	1	1	119

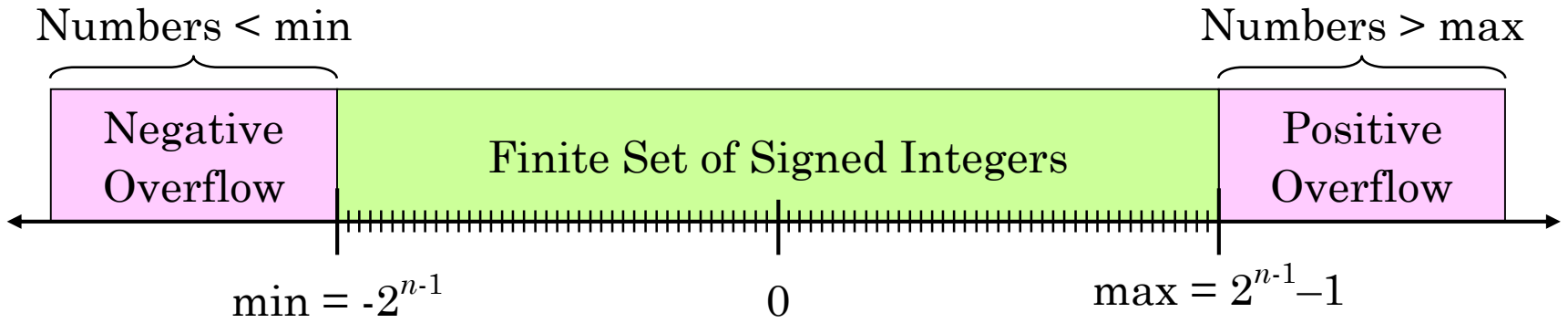
Carry = 1 Overflow = 1

Range, Carry, Borrow, and Overflow

- Unsigned Integers: *n*-bit representation



- Signed Integers: *n*-bit 2's complement representation



Signed number ADD

6 + 9

```
  6   0110
+) 9   1001
-------------
 15   1111 -> 01111
```

6 + (- 9)

```
  9   1001
-) 6   0110
-------------
 - 3   0011 -> 10011
```

-6 + 9

```
  9   1001
-) 6   0110
-------------
  3   0011 -> 00011
```

-6 + (-9)

```
  6   0110
+) 9   1001
-------------
-15   1111 -> 11111
```

Overflow 9 + 9 or (-9) + (-9)

Summation of complement number


```
+)   6   0   0110
     9   0   1001
-------------------
    15   0   1111
```

```
+)  -6   1   1010
     9   0   1001
-------------------
     3   0   0011
```

```
+)   6   0   0110
    -9   1   0111
-------------------
    -3   1   1101
```

```
+)   9   0   1001
     9   0   1001
-------------------
    18   1   0010
```

overflow

$x'_{n-1} y'_{n-1} s_{n-1}$

```
+)  -9   1   0111
    -9   1   0111
-------------------
-18 (1)0   1110
```

$x_{n-1} y_{n-1} s'_{n-1}$

Signed number ADD with complement-

- end-around carry
  - دو عدد را همراه با بيت علامتشان جمع مي كنيم. اگر يك رقم نقلي از پر ارزش ترين بيت خارج شد حاصل جمع را با 1 جمع مي‌كنيم و از رقم نقلي صرفنظر مي كنيم.
  - سرريز: $c_{n-1} \oplus c_n$

+)	6	0	0110
	-9	1	0110
	-3	1	1100

+)	11	0	1011
	-9	1	0110
	(1)	0	0001
+)			1
	2	0	0010

+)	9	0	1001
	9	0	1001
	-13	1	0010

+)	-9	1	0110
	-9	1	0110
	(1)	0	1100
+)			1
	13	0	1101

overflow

Subtraction

- تفریق در سیستم مکمل دو
  - مکمل دوی تفریق‌شونده (همراه با بیت علامت) را گرفته و حاصل را با عملگر دیگر جمع می کنیم.

$$(\pm A) - B = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + B$$

Numbers with the float point

در این نمایش مکان ممیز قسمت کسری ثابت نیست.

$F = E\ M$ طول عدد قابل نَمایش در این روش بسیار گسترده است

m_n	$e_k e_{k-1} \dots e_0$	$m_{n-1}\ m_{n-2}\ \dots\ m_0 . m_{-1}\ \dots\ m_{-m}$
علامت	نما	مانتیس

مانتیس: عدد علامت‌دار با ممیز ثابت (خواه عدد صحیح و یا عدد کسری)

نما: مکان ممیز را مشخص می کند

مقدار معادل ده‌دهی

$V(F) = V(M) * RV(E)$

M: Mantissa, E: Exponent, R: Radix

Numbers with the float point

مثال:

sign		sign	
0	.1234567	0	04
mantissa		exponent	

$$==> \ +.1234567 \times 10^{+04}$$

نکته: در نمایش اعداد به صورت شناور تنها مانتیس(M)و نما(E)صریحاًنمایش داده می شوند. مبنا و مکان ممیز به صورت ضمنی استنباط می شوند.

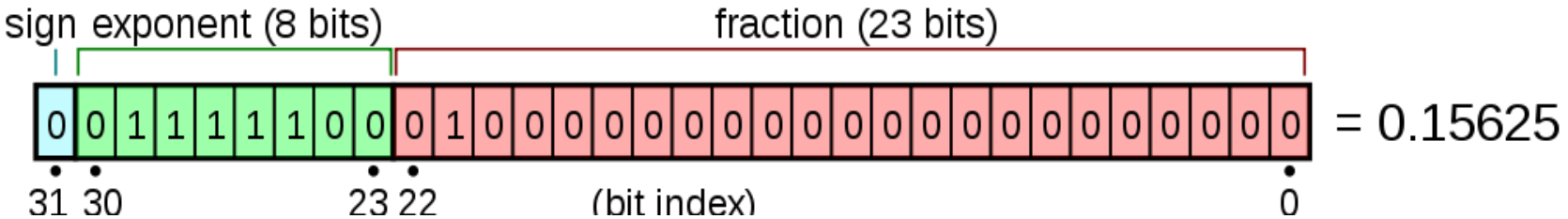
Numbers with the float point

مثال:عدد 1001.11+را به صورت يك عدد با ممیز شناور 16بيتي (6بيت نما و 10 بيت مانتيس)نمايش دهيد.

علامت	نما	مانتيس
0	0 00100	100111000
0	0 00101	010011100

يا

نمایش استانداردIEEE 754



- The real value assumed by a given 32 bit data with a given biased exponent e and a 23 bit fraction is

$$= (-1)^{sign}(1.b_{-1}b_{-2}...b_{-23})_2 \times 2^{e-127}$$

$$value = (-1)^{sign}(1 + \sum_{i=1}^{23} b_{-i}2^{-i}) \times 2^{(e-127)}$$

Reference