ECE 0142 Computer Organization

Lecture 2 Number Representation

Slides adapted from
Ron Hoelzeman

Data Representation

- **Computers manipulate discrete digital quantities - that is 1's or 0's**
- **Example representations of data**

Numeric	Non Numeric
Binary	**ASCII**
Decimal	**EBCDIC** (Extended Binary Coded Decimal Interchange Code)
BCD (binary-coded decimal)	

Examples of Data

Binary - 000001010011100

668

BCD - 0000 0010 0011 0100 1100 …

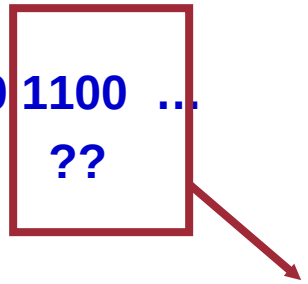
0 2 3 4 ??

ASCII - 0000000 NUL

0110000 0

0110001 1

0110010 2



Does not exist because
12 is not a decimal digit

Powers of 2

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

ASCII Standard Code

$B_4B_3B_2B_1$ \ $B_7B_6B_5$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

start of heading

1100100 - d

Data Representation Examples

	16 bit word	
	Byte 1	Byte 2
Binary	0000 0000	0001 0101
	21	
BCD	0011 0001	0010 0001
	3 1	2 1
ASCII	0011 0010	0011 0001
	2	1

Pure Binary vs. BCD

- **Advantages**
  - Scaling by a factor of 10 (or a power of 10) is simple, especially helpful for non-integer quantities (e.g., in financial calculations).
  - Rounding at a decimal digit boundary is easier
  - Conversion to a character form or for display is a simple per-digit mapping.
- **Disadvantages**
  - Some arithmetic operations are more complex to implement.
  - BCD does not make optimal use of storage

Number Representation

- Integers
- Fractions
- Negative numbers
- Floating point

Number Representation

Base 10

$$1109_{10} = 1*10^3 + 1*10^2 + 0*10^1 + 9*10^0$$

General Form

$$a_{n-1}a_{n-2} \ldots a_2a_1a_0.a_{-1}a_{-2}a_{-3}$$

Integer

"Decimal" point

$$a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \ldots + a_2b^2 + a_1b^1 + a_0b^0$$
$$+ a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3}$$

Fraction

Number Representations

Base	Data Range for single digit
Binary - base 2	0-1
Decimal - base 10	0-9
Octal - base 8	0-7
Hexadecimal - base 16	0-9,A-F

Number Representations

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Binary Integer Numbers

10110_2 =

$$1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$

$$= 16 + 0 + 4 + 2 + 0 = 22_{10}$$

10101_2 =

$$0*2^5+1*2^4+0*2^3+1*2^2+0*2^1+1*2^0$$

$$= 0 +16 + 0 + 4 + 0 + 1 = 21_{10}$$

Fractional Numbers

- Fixed Point
- Floating Point
- Single Precision
- Double Precision

Binary Fraction Examples

011.11_2

$= 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2}$

$= 0 + 2 + 1 + ½ + ¼ = 3.75_{10}$

110.101_2

$= 1*2^2+1*2^1+0*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}$

$= 4 +2 + 0 + ½ + 0 + ⅛ = 6.625_{10}$

Other Number Systems: Octal

654_8

$= 6*8^2 + 5*8^1 + 4*8^0$

$= 384 + 40 + 4 = 428_{10}$

255_8

$= 2*8^2 + 5*8^1 + 5*8^0$

$= 128 + 40 + 5 = 173_{10}$

Octal Fractions

177.4_8

$= 1*8^2 + 7*8^1 + 7*8^0 + 4*8^{-1}$

$= 64 + 56 + 7 + ½ = 127.5_{10}$

167.24_8

$= 1*8^2 + 6*8^1 + 7*8^0 + 2*8^{-1} + 4*8^{-2}$

$= 64 + 48 + 7 + 2/8 + 4/64 = 119.3125_{10}$

Hexadecimal

$0x710_{16}$

$= 7*16^2 + 1*16^1 + 0*16^0$

$= 1792 + 16 + 0 = 1808_{10}$

$0xA21_{16}$

$= 10*16^2 + 2*16^1 + 1*16^0$

$= 2560 + 32 + 1 = 2593_{10}$

Hexadecimal Fractions

$0x1F.F_{16}$

$= 1*16^{1} + 15*16^{0} + 15*16^{-1}$

$= 16 + 15 + 15/16 = 31.9375_{10}$

$0x3F.4_{16}$

$= 3*16^{1} + 15*16^{0} + 4*16^{-1}$

$= 48 + 15 + 4/16 = 63.25_{10}$

Conversions

- To decimal - already done
- Binary to octal
- Binary to hexadecimal
- Decimal to binary
- Decimal to octal
- Decimal to hexadecimal

Binary to Octal

Just take 3 bits at a time from bottom

010 110 100 111

2 6 4 7

= 2647_8

001 011 111 101 110

1 3 7 5 6

= 13756_8

Binary to Hexadecimal

Take 4 bits at a time from bottom

0101 1010 0111

5 A 7

= $5A7_{16}$

Note
A =10

001 0111 1110 1110

1 7 E E

= $17EE_{16}$

Note
E =14

Octal and Hexadecimal

Note:

The primary purpose of octal and hexadecimal is a convenient way to represent binary numbers

Integer Decimal to Binary

Recall Base 2

$11010_2 =$

$$1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$$

$$= 16 + 8 + 0 + 2 + 0 = 26_{10}$$

or

$26_{10} =$

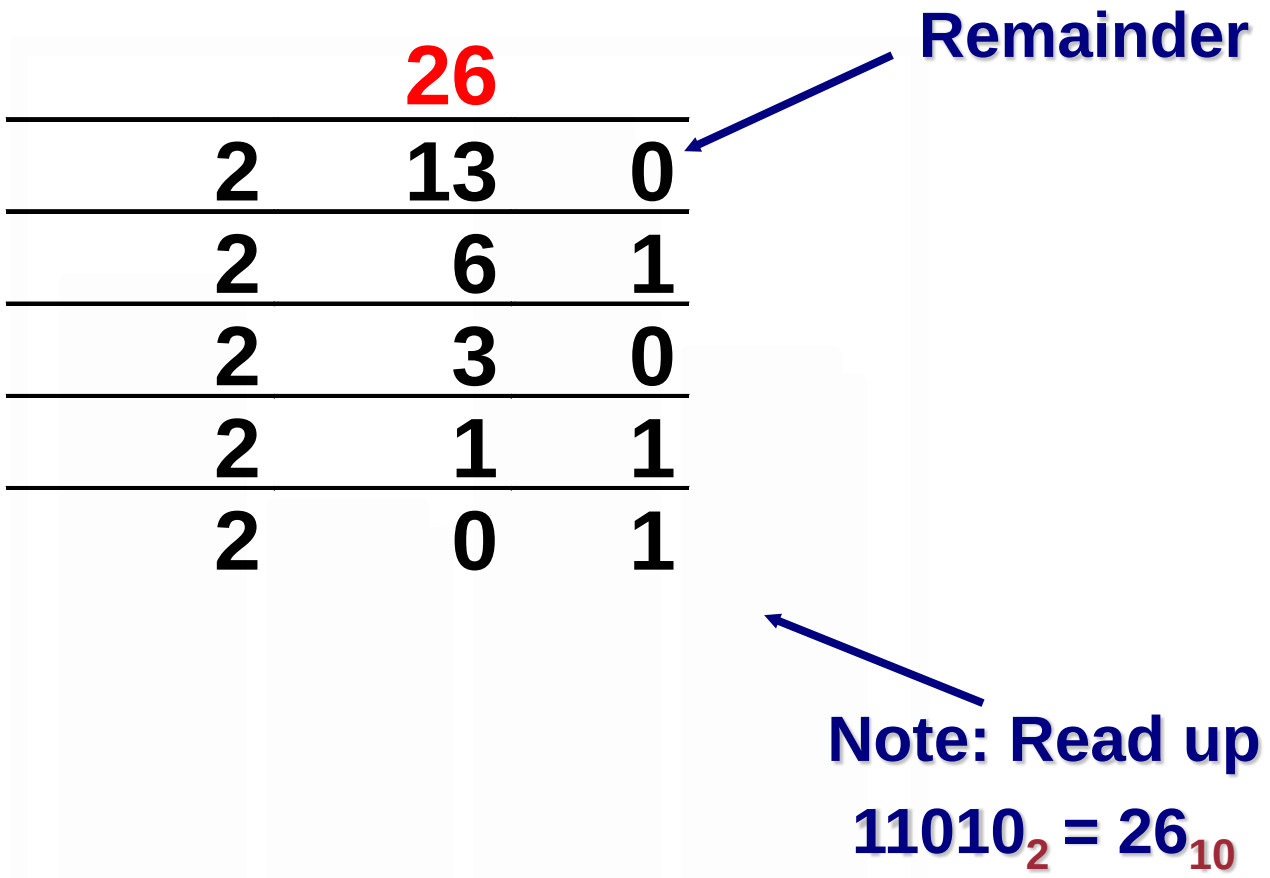
$$16 + 8 + 0 + 2 + 0$$

$$= x*2^4 + y*2^3 + z*2^2 + u*2^1 + v*2^0$$

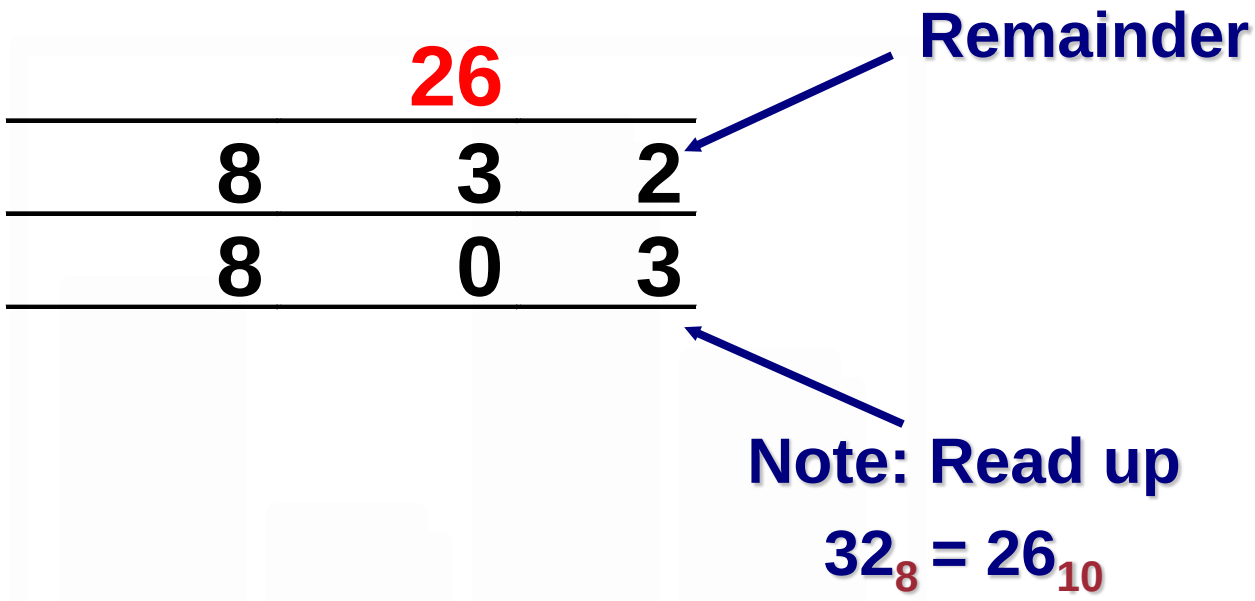
$$= xyzuv_2$$

$$= (((x*2 + y)*2 + z)*2 + u)*2 + v$$

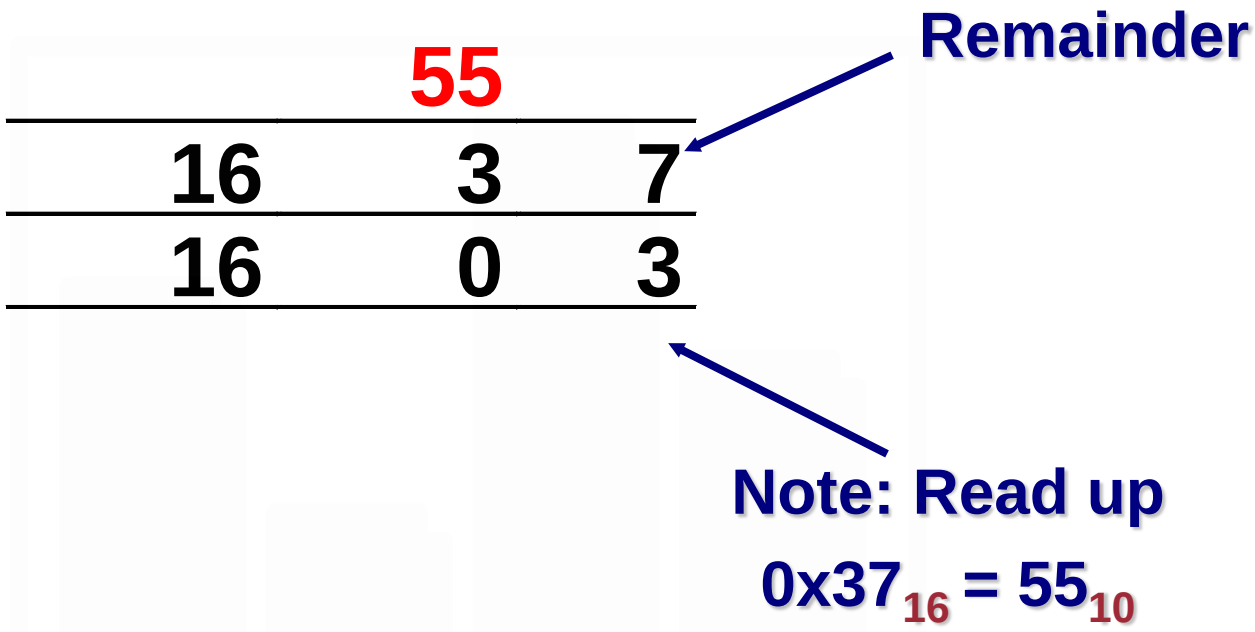
Integer Decimal to Binary



Integer Decimal to Octal



Integer Decimal to Hexadecimal



Exercise

- How to convert fractional numbers into binaries ?

Negative Numbers

- **Use MSB - most significant bit**
  - **“0” is plus +**
  - **“1” is minus -**
- **Three primary conventions**
  - **Sign magnitude**
  - **One’s complement— covered in recitation**
  - **Two’s complement**

Negative Numbers

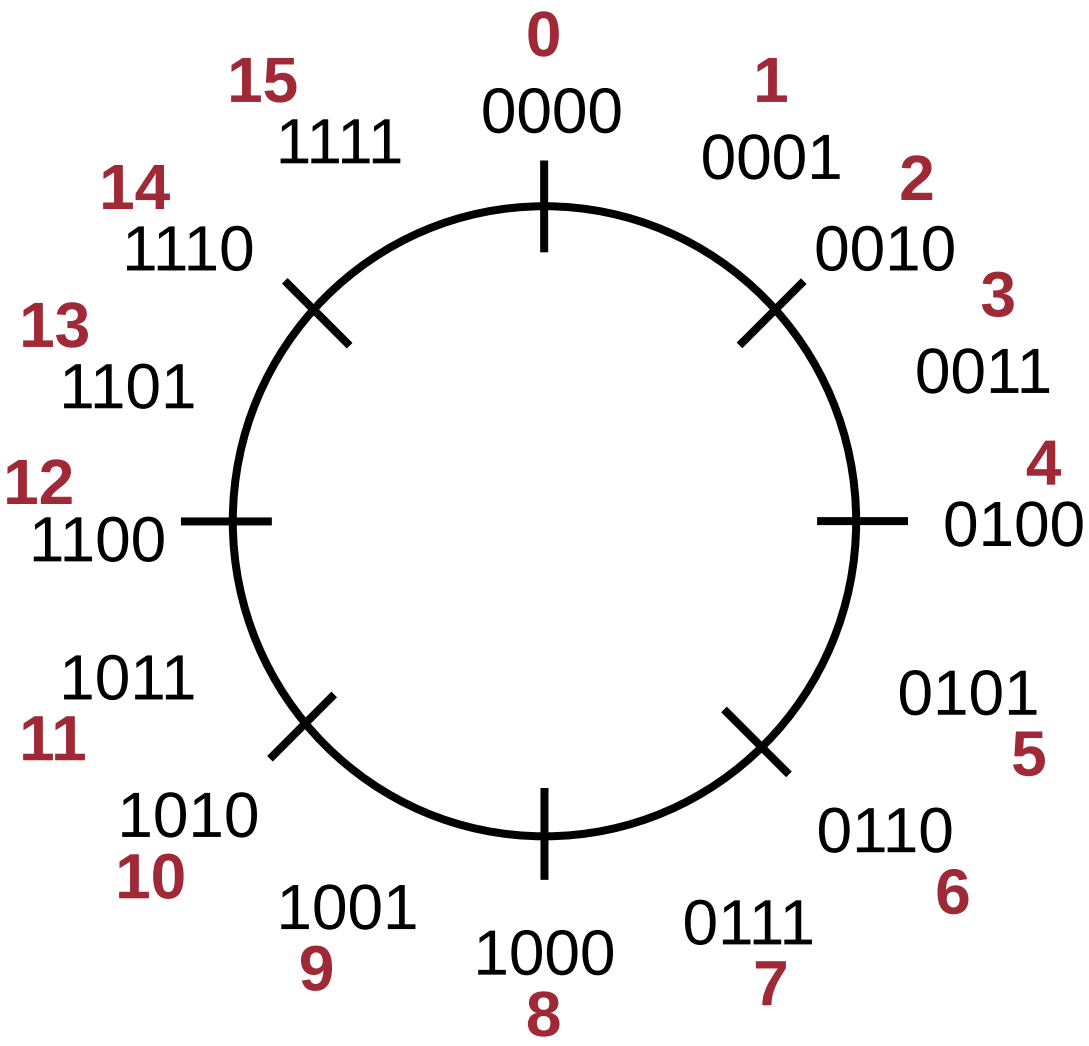
- **If negative numbers are included, range of magnitude is cut in half**

- **Example – for 9-bit binary numbers**

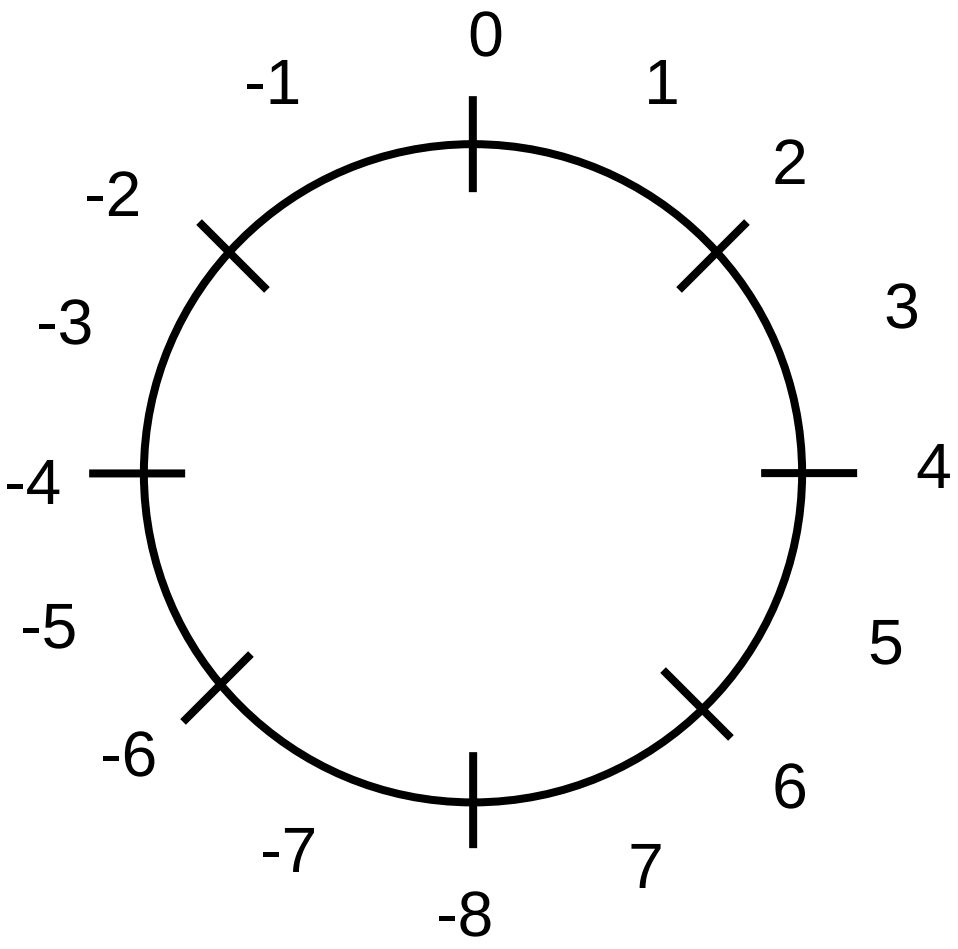
  All positive (0 to 511)

  When include Negative (+255 to 0 to -256)

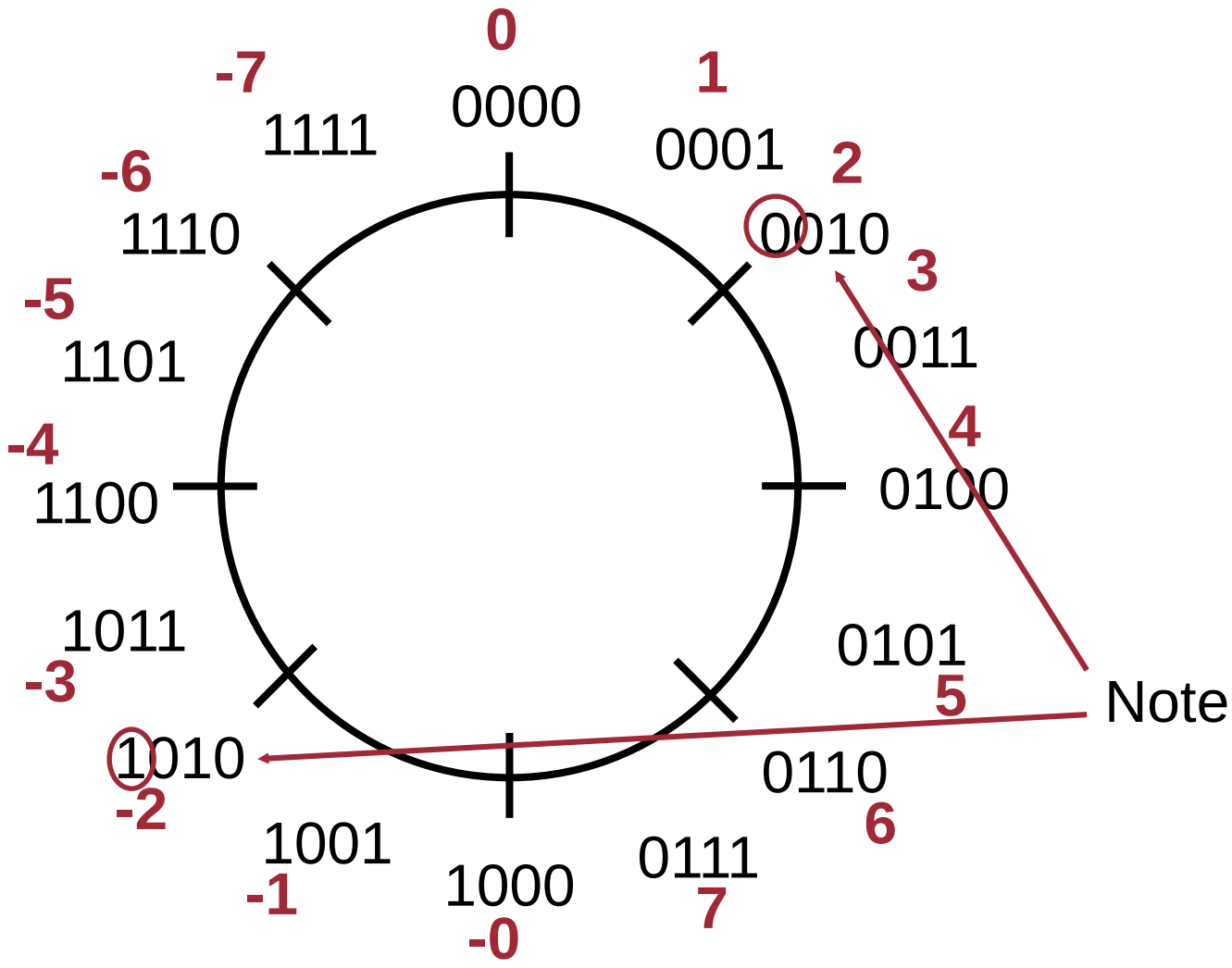
Positive Binary Numbers



Negative Numbers



Sign-Magnitude Numbers



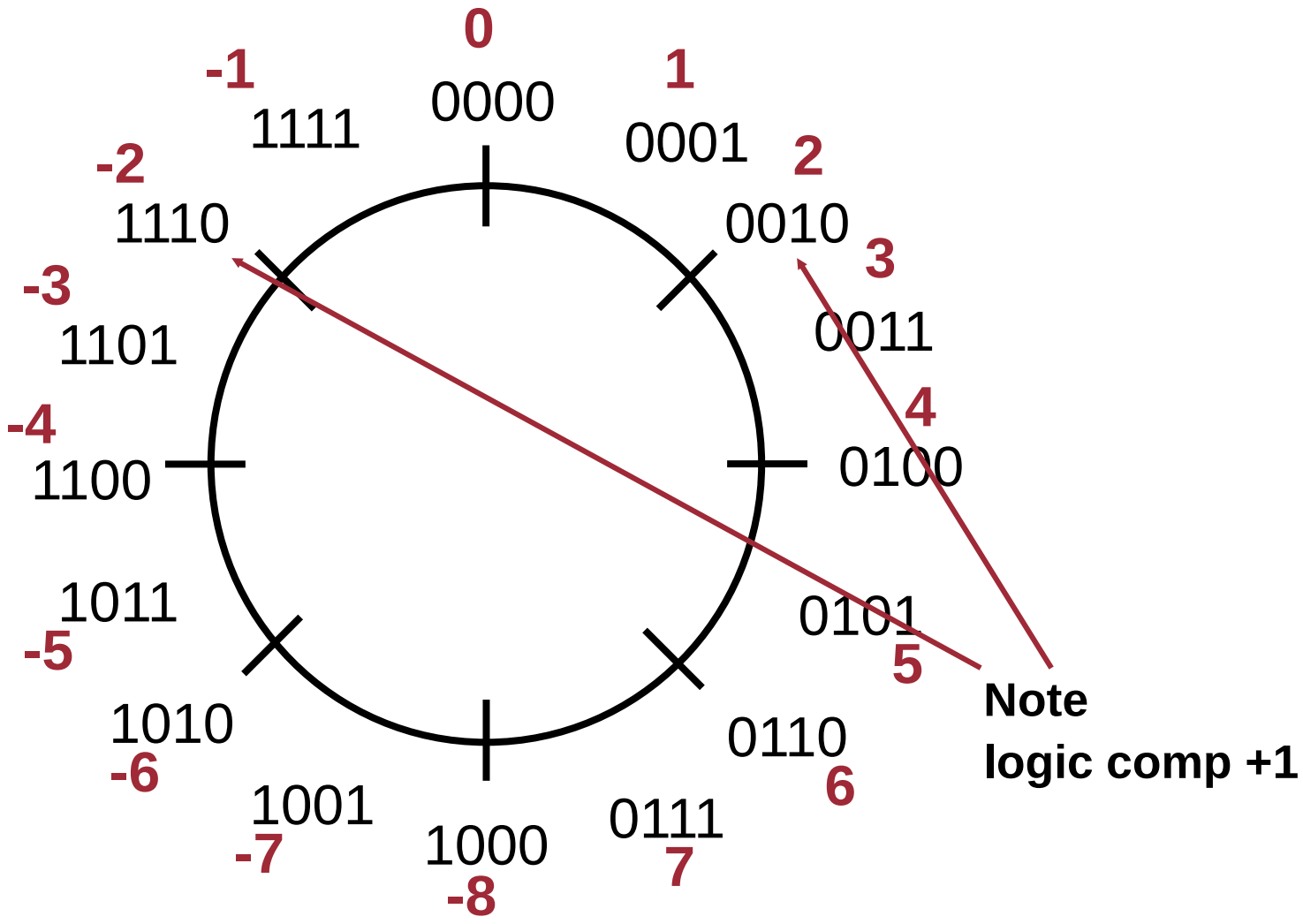
Sign Magnitude

- **Pro - easy to understand**

  easy to get negative

  example +4 (0100) » -4 (1100)
- **Con – 2 zero's**
- **Seldom directly used**

Two's Complement Numbers



Two's Complement Numbers

- **Pro - To get negative**
  - **complement all bits & add 1**
  - **example +4 (0100) » -4 (1100)**
  - **one zero**
- **Fastest for logic implementation**
  - **no end around carry – will be more clear in ALU implementation**
- **General form is different**

$$-a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \ldots + a_2b^2 + a_1b^1 + a_0b^0 \quad \text{integer}$$

$$+ a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3} \quad \text{fraction}$$

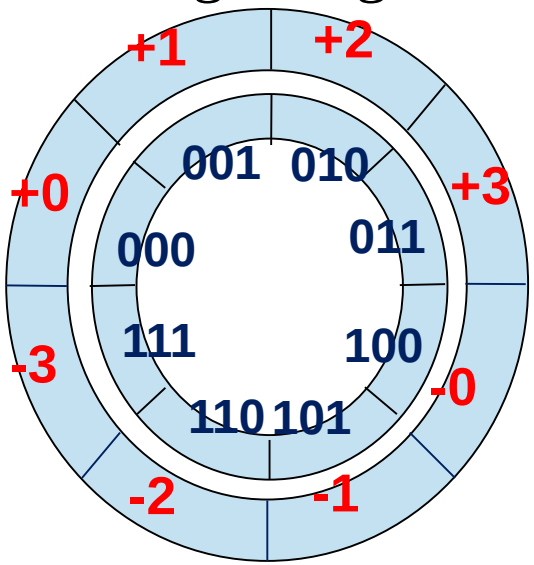
Notice "-" sign for MSB

- **Sign extension**

	4 bit	5 bit	6 bit
-7	1001	11001	111001

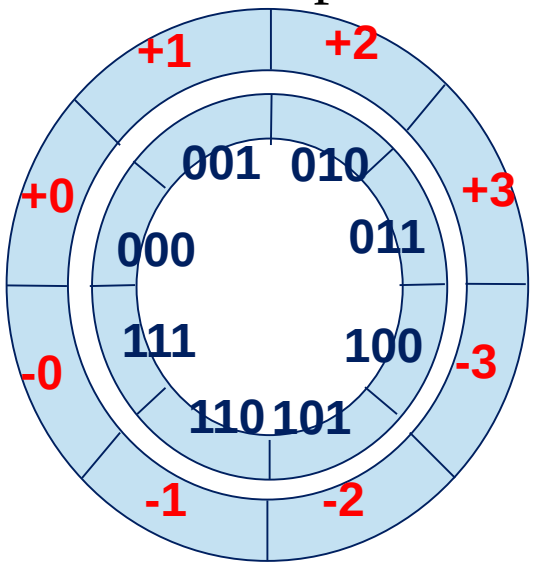
Signed Number Representation

sign-magnitude



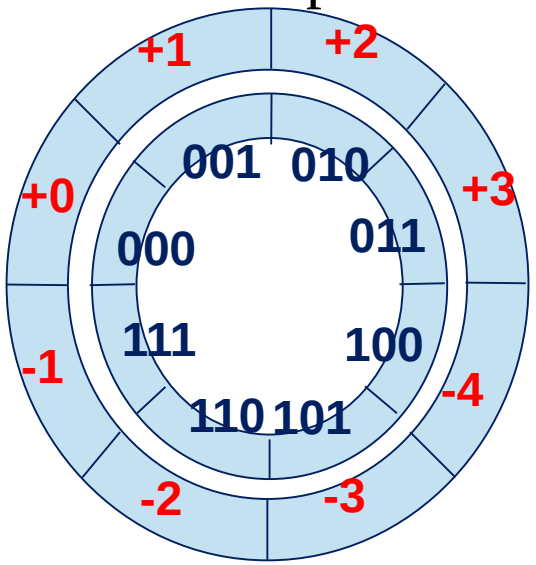
sign	magnitude

1’s complement



0	magnitude
1	Inv(magnitude)

2’s complement



0	magnitude
1	Inv(magnitude) +1

What About Fractions in 2's Complement? Examples

- What's 2's complement representation of -15.9_{10} ? Assume you have 16 bits in total, and 6 bits are used for fraction.

15_{10} = 00 0000 1111 (10 digits)

$.9_{10}$ = .1110011001100…

$.9_{10} \times 2 = \underline{1}.8$ $.8_{10} \times 2 = \underline{1}.6$ $.6_{10} \times 2 = \underline{1}.2$

$.2_{10} \times 2 = \underline{0}.4$ $.4_{10} \times 2 = \underline{0}.8$

15.9_{10} = 00 0000 1111.1110 01

2's complement of -15.9_{10} is:

11 1111 0000.0001 11

Binary Addition

Carry	Binary						Decimal		
	1	1	1	1	0		0	0	
		1	1	0	1	1		2	7
		1	0	1	1	0		2	2
	1	1	0	0	0	1		4	9

Return later with signs