Lecture 2

Number Systems and Codes
(Cont.)

Complementary Number System

- Radix Complement Arithmetic
  - CASE 1 : Compute A = B + C

$$(A)_2 = (B)_2 + (C)_2$$

Compute $(9)_{10} + (5)_{10}$ using 5-bit two's complement arithmetic.

$$+(9)_{10} = +(1001)_2 = (0.1001)_{2cmp}$$
$$+(5)_{10} = +(0101)_2 = (0.0101)_{2cmp}$$

Adding these two 5-bit codes gives

$$\begin{array}{rccccc} & 0 & 1 & 0 & 0 & 1 \\ + & 0 & 0 & 1 & 0 & 1 \\ \hline & 0 & 1 & 1 & 1 & 0 \end{array} = +(1110)_2 = +(14)_{10}$$

Complementary Number System

Compute $(12)_{10} + (7)_{10}$.

From Table 1.6,

$$(12)_{10} = +(1100)_2 = (0, 1100)_{2\text{'s}}$$
$$(7)_{10} = +(0111)_2 = (0, 0111)_{2\text{'s}}$$

Adding the two 5-bit codes gives

$$\begin{array}{rccccc} & 0 & 1 & 1 & 0 & 0 \\ + & 0 & 0 & 1 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 & 1 \end{array}$$

The result is $(1, 0011)_{2\text{'s}}$, which from Table 1.6 is interpreted as

$$(1, 0011)_{2\text{'s}} = -(1101)_2 = -(13)_{10}$$

Hence an overflow condition has occurred.

Complementary Number System

- CASE 2: Compute A = B – C → A = B + (-C)

$$(A)_2 = (B)_2 + (-C)_2$$

$$\begin{aligned} A &= (B)_2 + [C]_2 \\ &= (B)_2 + 2^n - (C)_2 \\ &= 2^n + (B - C)_2 \end{aligned}$$

Compute $(12)_{10} - (5)_{10}$.

We perform this computation as $(12)_{10} + (-(5)_{10})$

$$(12)_{10} = +(1100)_2 = (0, 1100)_{2\text{'s}}$$
$$-(5)_{10} = -(0101)_2 = (1, 1011)_{2\text{'s}}$$

Adding the two 5-bit codes gives

$$\begin{array}{rccccc} & 0 & 1 & 1 & 0 & 0 \\ + & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 \end{array}$$

↑
Carry

Discarding the carry, the sign bit is seen to be 0, and therefore interpreted as

$$(0, 0111)_{2\text{'s}} = +(0111)_2 = +(7)_{10}$$

Complementary Number System

Reversing the order of the operands from the previous example, compute $(5)_{10} - (12)_{10}$.

We perform the computation as $(5)_{10} + (-(12)_{10})$.

$$(5)_{10} = +(0101)_2 = (0, 0101)_{2cns}$$
$$-(12)_{10} = -(1100)_2 = (1, 0100)_{2cns}$$

Adding the two 5-bit codes gives

$$\begin{array}{rrrrrr} & 0 & 0 & 1 & 0 & 1 \\ + & 1 & 0 & 1 & 0 & 0 \\ \hline & 1 & 1 & 0 & 0 & 1 \end{array}$$

In this case there is no carry, and the sign bit is 1,

$$(1, 1001)_{2cns} = -(0111)_2 = -(7)_{10}$$

Complementary Number System

Compute $(0, 0111)_{2cns} - (1, 1010)_{2cns}$.

Therefore,

$$-(1, 1010)_{2cns} = (0, 0110)_{2cns}$$

Adding the two 5-bit codes gives

$$\begin{array}{rrrrrr} & 0 & 0 & 1 & 1 & 1 \\ + & 0 & 0 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & 0 & 1 \end{array}$$

The result is positive, as indicated by the 0 sign bit, and is interpreted as

$$(0, 1101)_{2cns} = +(1101)_2 = +(13)_{10}$$

The reader should verify that this computation is equivalent to computing $(7)_{10} - (-(6)_{10}) = (13)_{10}$.

Complementary Number System

- CASE 3: Compute A = - B – C → A = - (B + C)
  → A = (- B) + (- C)

Compute $-(9)_{10} - (5)_{10}$.

We perform the computation as $(-(9)_{10}) + (-(5)_{10})$.

$$-(9)_{10} = -(1001)_2 = (1, 0111)_{2cns}$$
$$-(5)_{10} = -(0101)_2 = (1, 1011)_{2cns}$$

Adding the two 5-bit codes gives

$$\begin{array}{rccccc} & 1 & 0 & 1 & 1 & 1 \\ + & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

↑ Carry

$$A = [B]_2 + [C]_2$$
$$= 2^n - (B)_2 + 2^n - (C)_2$$
$$= 2^n + 2^n - (B + C)_2$$
$$= 2^n + [B + C]_2$$

Discarding the carry leaves a sign bit of 1. Therefore, the result is correct and is interpreted as

$$(1, 0010)_{2cns} = -(1110)_2 = -(14)_{10}$$

Complementary Number System

Compute $-(12)_{10} - (5)_{10}$.

We perform the computation as $(-(12)_{10}) + (-(5)_{10})$

$$-(12)_{10} = -(1100)_2 = (1, 0100)_{2cns}$$
$$-(5)_{10} = -(0101)_2 = (1, 1011)_{2cns}$$

Adding the two 5-bit codes gives

$$\begin{array}{rccccc} & 1 & 0 & 1 & 0 & 0 \\ + & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

↑ Carry

Discarding the carry, the result is interpreted as

$$(0, 1111)_{2cns} = +(1111)_2 = +(15)_{10}$$

Hence an overflow condition has occurred.

Complementary Number System

TABLE SUMMARY OF TWO'S COMPLEMENT ADDITION AND SUBTRACTION

Case*	Carry	Sign Bit	Condition	Overflow?
$B + C$	0	0	$B + C \le 2^{n-1} - 1$	No
	0	1	$B + C > 2^{n-1} - 1$	Yes
$B - C$	1	0	$B \le C$	No
	0	1	$B > C$	No
$-B - C$	1	1	$-(B + C) \ge -2^{n-1}$	No
	1	0	$-(B + C) < -2^{n-1}$	Yes

* B and C are positive numbers.

Complementary Number System

Add $+(75)_{10}$ and $-(21)_{10}$ using 3-digit ten's complement arithmetic.

$(75)_{10} = (0, 75)_{10cms}$

$-(21)_{10} = (9, 79)_{10cms}$

Then we perform the computation as $(75)_{10} + (-(21)_{10})$.

$$\begin{array}{rrrr} & 0 & 7 & 5 \\ + & 9 & 7 & 9 \\ \hline 1 & 0 & 5 & 4 \end{array}$$

↑ Carry

Discarding the carry digit, the result is $(0, 54)_{10cms} = (54)_{10}$.

Complementary Number System

Add $+(21)_{10}$ and $-(75)_{10}$.

Again, we begin by determining the ten's complement codes

$$(21)_{10} = (0.21)_{10\text{cns}}$$
$$-(75)_{10} = (9.25)_{10\text{cns}}$$

Adding the two 3-digit codes gives

$$\begin{array}{rrrr} & 0 & 2 & 1 \\ + & 9 & 2 & 5 \\ \hline & 9 & 4 & 6 \end{array}$$

$-(54)_{10}$.

Complementary Number System

- Diminished Radix Complement Number System : $[N]_{r-1}$

$$[N]_{r-1} = r^n - (N)_r - 1$$

  - The one's complement =

$$[N]_{2-1} = 2^n - (N)_2 - 1$$

where n is the number of bits in $(N)_2$.

Complementary Number System

Determine the one's complement of $(01100101)_2$.

$$\begin{aligned}[N]_{2-1} &= 2^8 - (01100101)_2 - 1 \\ &= (100000000)_2 - (01100101)_2 - (00000001)_2 \\ &= (10011011)_2 - (00000001)_2 \\ &= (10011010)_2.\end{aligned}$$

Determine the one's complement of $(11010100)_2$.

$$\begin{aligned}[N]_{2-1} &= 2^8 - (11010100)_2 - (00000001)_2 \\ &= (100000000)_2 - (11010100)_2 - (00000001)_2 \\ &= (00101100)_2 - (00000001)_2 \\ &= (00101011)_2.\end{aligned}$$

Complementary Number System

Find the nine's complement of $(40960)_{10}$.

$$\begin{aligned}[N]_{10-1} &= 10^5 - (40960)_{10} - (00001)_{10} \\ &= (100000)_{10} - (40960)_{10} - (00001)_{10} \\ &= (59040)_{10} - (00001)_{10} \\ &= (59039)_{10}.\end{aligned}$$

Complementary Number System

TABLE 1.6 SIGNED NUMBER REPRESENTATION EXAMPLES*

Signed Decimal	Sign Magnitude Binary	Two's Complement System	One's Complement System
+15	0,1111	0,1111	0,1111
+14	0,1110	0,1110	0,1110
+13	0,1101	0,1101	0,1101
+12	0,1100	0,1100	0,1100
+11	0,1011	0,1011	0,1011
+10	0,1010	0,1010	0,1010
+9	0,1001	0,1001	0,1001
+8	0,1000	0,1000	0,1000
+7	0,0111	0,0111	0,0111
+6	0,0110	0,0110	0,0110
+5	0,0101	0,0101	0,0101
+4	0,0100	0,0100	0,0100
+3	0,0011	0,0011	0,0011
+2	0,0010	0,0010	0,0010
+1	0,0001	0,0001	0,0001
0	0,0000 (1,0000)	0,0000	0,0000 (1,1111)
-1	1,0001	1,1111	1,1110
-2	1,0010	1,1110	1,1101
-3	1,0011	1,1101	1,1100
-4	1,0100	1,1100	1,1011
-5	1,0101	1,1011	1,1010
-6	1,0110	1,1010	1,1001
-7	1,0111	1,1001	1,1000
-8	1,1000	1,1000	1,0111
-9	1,1001	1,0111	1,0110
-10	1,1010	1,0110	1,0101
-11	1,1011	1,0101	1,0100
-12	1,1100	1,0100	1,0011
-13	1,1101	1,0011	1,0010
-14	1,1110	1,0010	1,0001
-15	1,1111	1,0001	1,0000
-16	—	1,0000	—

Computer Code

- Numeric Code

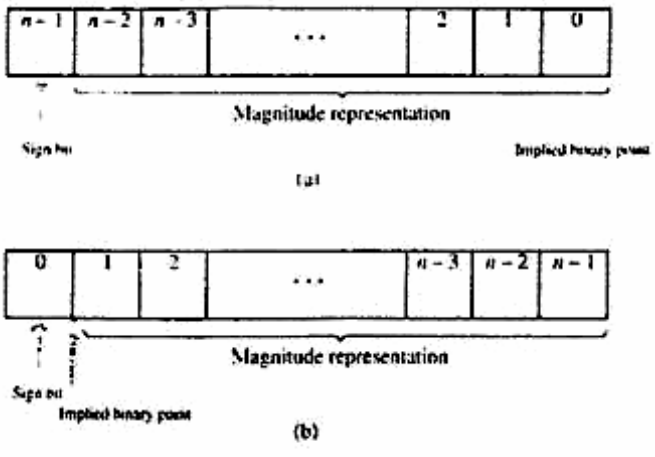

Figure 1.3 Fixed-point number representations. (a) Fixed-point integer. (b) Fixed-point fraction.

Give two possible interpretations of the 8-bit fixed-point number 01101010, using the two's complement system.

= +1101010 (or)
= +0.1101010

Give two possible interpretations of the 8-bit fixed-point number 11101010, using the two's complement system.

= - 0010110 (or)
= - 0.0010110

Computer Code

- Excess or Biased Representation
  - Frequently used in the representation of the exponents of floating-point numbers.
  - Excess-2^n number are just the two's complement number with sign bit reversed.
  - The Excess-8 produced by adding $(1000)_2$ to the two's complement code.

Computer Code

TABLE 1.8 EXCESS-8 CODE

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

Computer Code

- Floating-point Number
  - Floating-point of number N is written as :

$$N = M \times r^E$$

when : M is mantissa or significant

E is Exponent

$$N = \pm(a_{n-1} \ldots a_0 . a_{-1} \ldots a_{-m})_r$$

then in floating-point form

$$N = \pm(.a_{n-1} \ldots a_{-m})_r \times r^n$$

Computer Code

$$M = (S_M . a_{n-1} \ldots a_{-m})_{r\text{sm}}$$

$$M = (-1)^{S_M} \times (.a_{n-1} \ldots a_{-m})_r$$

S_M = 0 indicate a positive number.

S_M = 1 indicate a negative number.

$$\begin{aligned} N &= M \times r^E \\ &= (M \div r) \times r^{E+1} \\ &= (M \times r) \times r^{E-1} \end{aligned}$$

$$\begin{aligned} M &= +(1101.0101)_2 \\ &= (0.11010101)_2 \times 2^4 \\ &= (0.011010101)_2 \times 2^5 \\ &= (0.0011010101)_2 \times 2^6 \\ &\vdots \end{aligned}$$

Computer Code

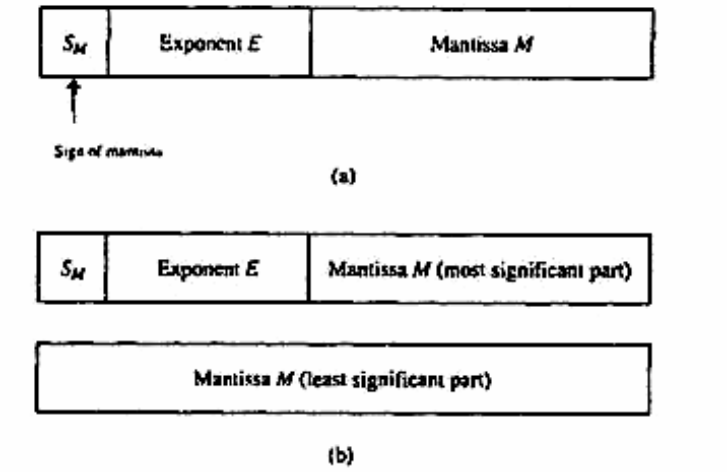

Figure 1.4 Floating-point number formats. (a) Typical single-precision format. (b) Typical extended-precision format.

Computer Code

TABLE 1.9 SOME COMMON FLOATING POINT NUMBER FORMATS*

System/ Format	Total bits	Significand bits	Exponent bits	Exponent bias
IEEE Std. 754-1985:				
Single Precision	32	23 (+1)	8	127
Double Precision	64	52 (+1)	11	1023
IBM System/360:				
Single Precision	32	24	7	64
Double Precision	64	56	7	64
DEC VAX 11/780:				
F Format	32	23 (+1)	8	128
D Format	64	55 (+1)	8	128
G Format	64	52 (+1)	11	1024
CDC Cyber 70:	60	48	11	1024

Computer Code

Write the binary number $N = (101101.101)_2$ in the floating-point format of Eq. 1.17, where $n + m = 10$ and $e = 5$. Assume that a normalized sign magnitude fraction is used to represent M and that Excess-16 two's complement is used for E.

$$N = (101101.101)_2 = (0.101101101)_2 \times 2^6$$

Writing the mantissa in the format of Eq. 1.14:

$$M = +(0.1011011010)_2$$
$$= (0.1011011010)_{2sm}$$

The exponent is coded by determining its two's complement form adding a bias of 16. (Note that the number of exponent bits $e = 5$ a bias value is $2^{e-1} = 2^4 = 16$). Therefore,

$$E = +(6)_{10}$$
$$= +(0110)_2$$
$$= (00110)_{2cs}$$

Adding the bias value of $16 = (10000)_2$ to the two's complement

$$\begin{array}{rr} & 00110 \\ + & 10000 \\ \hline & 10110 \end{array}$$

So,

$$E = (1.0110)_{\text{excess-16}}$$

Note that the sign of the exponent, b_{e-1}, is 1, indicating a positive exponent value.

Combining M and E gives

$$N = (0.1.0110.1011011010)_{fp}$$

Computer Code

- Character and Other Codes
  - Binary Coded Decimal (BCD)

TABLE 1.10 BINARY CODED DECIMAL (BCD) CODES

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

Encode the decimal number $N = (9750)_{10}$ in BCD.

First, the individual digits are encoded from Table 1.10.

$$9 \rightarrow 1001,\ 7 \rightarrow 0111,\ 5 \rightarrow 0101,\ \text{and}\ 0 \rightarrow 0000$$

Then the individual codes are concatenated to give

$$N = (1001011101010000)_{\text{BCD}}$$

Computer Code

- ASCII (American Standard Code for Information Interchange)

TABLE 1.11 ASCII CHARACTER CODE

$c_3c_2c_1c_0$ \ $c_6c_5c_4$	000	001	010	011	100	101	110	111
0000	NUL	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

Computer Code

- Gray Codes : A cyclic code

TABLE 1.12 GRAY CODE FOR DECIMAL NUMBERS 0 THROUGH 15

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000