

Chapter 2

Number Systems, Arithmetic, and Code

Positional number systems

- What is the underlying principle?
- Can you find an example of a number system that is not positional?
- What are the reasons for using different bases?

Notational convention

- 234.16
 $= 2 \times 100 + 3 \times 10 + 4 \times 1 + 1 \times 0.1 + 6 \times 0.01$
 $= 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + 1 \times 10^{-1} + 6 \times 10^{-2}$

Basic arithmetic operations

- The basic operations are addition, subtraction, multiplication, and division.
- They are very similar for positional number systems with different bases.
- Solutions to any computational problems to be solved on a computer must be expressed in terms of these operations.

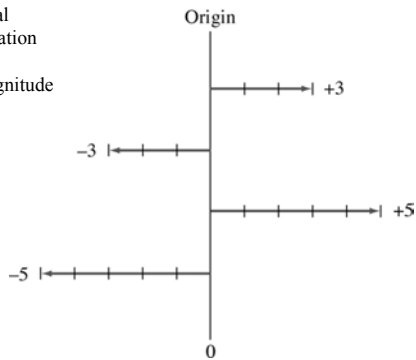
Methods of number conversion

- Polynomial method
- Iterative method
- Special conversion method

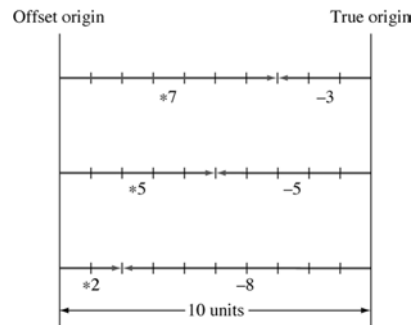
Signed numbers

- sign and magnitude
- r 's-complement
- $(r-1)$'s complement

Graphical interpretation of sign-magnitude numbers



Graphical interpretation of complements



Subtraction

- Shown below are the steps involved in performing $M-N$ through complementation and addition.
- Here M and N are unsigned numbers.

$(r-1)$'s vs. r 's complement

	$(r-1)$'s complement	r 's complement
Definition: given N with n digits	$(r^n - 1) - N$	$r^n - N$

$(r-1)$'s vs. r 's complement

	$(r-1)$'s complement	r 's complement
Computation involved in obtaining the complement	digitwise (bitwise) complementation	digitwise (bitwise) complementation + 1

$(r-1)$'s vs. r 's complement

	$(r-1)$'s complement	r 's complement
Number of zero	two (+0 and -0)	one

(r-1)'s vs. r's complement

	(r-1)'s complement	r's complement
Subtraction operation	$M - N \rightarrow M + (r^n - 1) - N$ $= r^n + M - N - 1$ $= (r^n - 1) - (N - M)$	$M - N \rightarrow M + r^n - N$ $= r^n + M - N$ $= r^n - (N - M)$

(r-1)'s vs. r's complement

	(r-1)'s complement	r's complement
If $M > N$	$M - N \rightarrow r^n + M - N - 1$ <i>What will happen?</i> There is a carry. <i>How to produce M-N?</i> (1) Subtract r^n by discarding the carry. (2) Add 1 to it.	$M - N \rightarrow M + r^n - N$ <i>What will happen?</i> There is a carry. <i>How to produce M-N?</i> Subtract r^n by discarding the carry.

(r-1)'s vs. r's complement

	(r-1)'s complement	r's complement
if $M < N$	$M - N \rightarrow (r^n - 1) - (N - M)$ <i>What will happen?</i> There is no carry. <i>How to produce M-N (in sign-and-magnitude)?</i> The result is negative and in (r-1)'s complement. (1) Perform (r-1)'s complement to obtain (N-M). (2) Prefix it with a minus sign to indicate that it is negative.	$M - N \rightarrow r^n - (N - M)$ <i>What will happen?</i> There is no carry. <i>How to produce M-N (in sign-and-magnitude)?</i> The result is negative and in r's complement. (1) Perform r's complement (i.e., (r-1)'s complement plus 1) to obtain (N-M). (2) Prefix it with a minus sign to indicate that it is negative.

(r-1)'s vs. r's complement

	(r-1)'s complement	r's complement
if $M = N$	$M - N \rightarrow r^n + M - N - 1$ <i>What will happen?</i> There is no carry. <i>How to produce M-N?</i> It is treated as if $M < N$, producing a "-0" as the result.	$M - N \rightarrow M + r^n - N$ <i>What will happen?</i> There is a carry. <i>How to produce M-N?</i> It is treated as if $M > N$, producing a "0" as the result.

Interpretation of four-bit signed binary integers

$b_3b_2b_1b_0$	sign and mag.	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

The use of 2's complement

- In practice, signed numbers are always represented by 2's complements because then there is only one zero.
- Existence of more than one zero leads to complication in programming.

$$\begin{array}{r} (+5) \quad 0101 \\ + (-2) \quad +0010 \\ \hline (+7) \quad 0111 \end{array} \quad \begin{array}{r} (-5) \quad 1010 \\ + (-2) \quad +0010 \\ \hline (-3) \quad 1100 \end{array}$$

$$\begin{array}{r} (+5) \quad 0101 \\ + (-2) \quad +1101 \\ \hline (+3) \quad 10010 \\ \quad \leftarrow 1 \\ \hline \quad 0011 \end{array} \quad \begin{array}{r} (-5) \quad 1010 \\ + (-2) \quad +1101 \\ \hline (-7) \quad 10111 \\ \quad \leftarrow 1 \\ \hline \quad 1000 \end{array}$$

Examples of 1's complement addition

$$\begin{array}{r} (+5) \quad 0101 \\ + (+2) \quad +0010 \\ \hline (+7) \quad 0111 \end{array} \quad \begin{array}{r} (-5) \quad 1011 \\ + (+2) \quad +0010 \\ \hline (-3) \quad 1101 \end{array}$$

$$\begin{array}{r} (+5) \quad 0101 \\ + (-2) \quad +1110 \\ \hline (+3) \quad 10011 \\ \quad \uparrow \\ \text{ignore} \end{array} \quad \begin{array}{r} (-5) \quad 1011 \\ + (-2) \quad +1110 \\ \hline (-7) \quad 11001 \\ \quad \uparrow \\ \text{ignore} \end{array}$$

Examples of 2's complement addition

$$\begin{array}{r} (+5) \quad 0101 \\ - (-2) \quad -0010 \\ \hline (+3) \quad 0101 \\ \quad \uparrow \\ \text{ignore} \end{array} \Rightarrow \begin{array}{r} 0101 \\ +1110 \\ \hline 10011 \\ \quad \uparrow \\ \text{ignore} \end{array}$$

$$\begin{array}{r} (-5) \quad 1011 \\ - (-2) \quad -0010 \\ \hline (-7) \quad 1011 \\ \quad \uparrow \\ \text{ignore} \end{array} \Rightarrow \begin{array}{r} 1011 \\ +1110 \\ \hline 11001 \\ \quad \uparrow \\ \text{ignore} \end{array}$$

$$\begin{array}{r} (+5) \quad 0101 \\ - (-2) \quad -1110 \\ \hline (+7) \quad 0101 \\ \quad \uparrow \\ \text{ignore} \end{array} \Rightarrow \begin{array}{r} 0101 \\ +0010 \\ \hline 0111 \end{array}$$

$$\begin{array}{r} (-5) \quad 1011 \\ - (-2) \quad -1110 \\ \hline (-3) \quad 1011 \\ \quad \uparrow \\ \text{ignore} \end{array} \Rightarrow \begin{array}{r} 1011 \\ +0010 \\ \hline 1101 \end{array}$$

Examples of 2's complement subtraction

Overflow

- In adding two binary numbers, an *overflow* condition is said to occur if the resulting sum requires more bits than are available.
- Let x be the carry into the sign-bit position and y be the carry from the sign-bit position, then there is an overflow if and only if $x \oplus y = 1$.

$$\begin{array}{r} (+7) \quad 0111 \\ + (+2) \quad +0010 \\ \hline (+9) \quad 1001 \\ c_4 = 0 \\ c_3 = 1 \end{array} \quad \begin{array}{r} (-7) \quad 1001 \\ + (+2) \quad +0010 \\ \hline (-5) \quad 1011 \\ c_4 = 0 \\ c_3 = 0 \end{array}$$

$$\begin{array}{r} (+7) \quad 0111 \\ + (-2) \quad +1110 \\ \hline (+5) \quad 10101 \\ c_4 = 1 \\ c_3 = 1 \end{array} \quad \begin{array}{r} (-7) \quad 1001 \\ + (-2) \quad +1110 \\ \hline (-9) \quad 10111 \\ c_4 = 1 \\ c_3 = 0 \end{array}$$

There is an overflow if $c_3 \oplus c_4 = 1$

Examples of determination of overflow

Decimal codes

- Weighted decimal codes
- Non-weighted decimal codes
- Bar codes

Weighted decimal codes

decimal digit	8421 code (BCD)	2421 code	5421 code	7536 code	biquinary code 5043210
0	0000	0000	0000	0000	0100001
1	0001	0001	0001	1001	0100010
2	0010	0010	0010	0111	0100100
3	0011	0011	0011	0010	0101000
4	0100	0100	0100	1011	0110000
5	0101	1011	1000	0100	1000001
6	0110	1100	1001	1101	1000010
7	0111	1101	1010	1000	1000100
8	1000	1110	1011	0110	1001000
9	1001	1111	1100	1111	1010000

J. C. Huang, 2003

Digital Logic Design

24

Nonweighted decimal codes

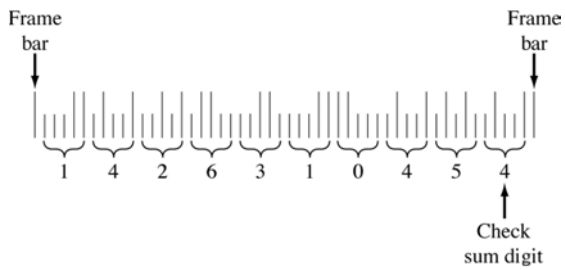
decimal digit	excess-3 code	2-out-of-5 code
0	0011	11000
1	0100	00011
2	0101	00101
3	0110	00110
4	0111	01001
5	1000	01010
6	1001	01100
7	1010	10001
8	1011	10010
9	1100	10100

J. C. Huang, 2003

Digital Logic Design

25

US Postal Service bar code (2 out of 5)



J. C. Huang, 2003

Digital Logic Design

26

Gray code (a unit distance code)

decimal number	Gray code
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

J. C. Huang, 2003

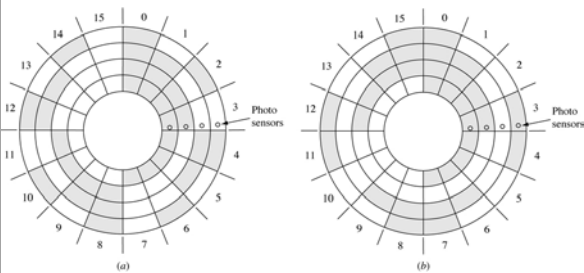
Digital Logic Design

27

Angular position encoders

conventional binary

Gray code

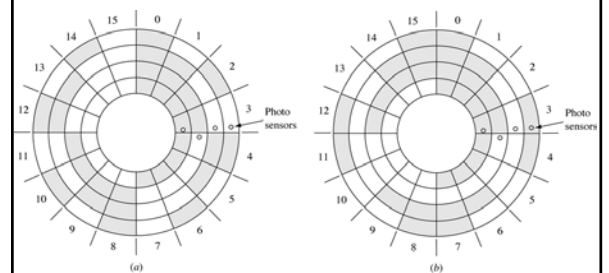


J. C. Huang, 2003

Digital Logic Design

28

The effects of misaligned sensors on the encoders



J. C. Huang, 2003

Digital Logic Design

29

Alphanumeric Codes

- ASCII code
- Unicode Standard

Error detection

- A parity bit can be used to detect single-bit errors.
- Additional parity bits can be used to detect multiple errors.

Examples of ASCII code with parity bit

characters	with even parity	with odd parity
.	.	.
.	.	.
.	.	.
A	10000010	10000011
B	10000100	10000101
C	10000111	10000110
D	10001000	10001001
.	.	.
.	.	.
.	.	.

Error correction

- Hamming code
- Use of check-sum digits

Hamming code

- Hamming code is an error-detection and error-correction binary code.
- A single-bit error can be automatically corrected if we can determine which bit is in error.
- A single-bit error can be detected by using a parity bit.
- Multiple parity bits can be used to pinpoint the bit in error.

Hamming code

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓

Example

For example, to construct the Hamming code of 00101110

Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
			0		0	1	0		1	1	1	0
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓
Bit position	1	2	3	4	5	6	7	8	9	10	11	12

Example (continued)

Choose 0 for P₁ (assuming the use of even parity)

Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
	0		0		0	1	0		1	1	1	0
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓
Bit position	1	2	3	4	5	6	7	8	9	10	11	12

Example (continued)

Choose 1 for P₂

Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
	0	1	0		0	1	0		1	1	1	0
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓
Bit position	1	2	3	4	5	6	7	8	9	10	11	12

Example (continued)

Choose 1 for P₄

Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
	0	1	0	1	0	1	0		1	1	1	0
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓
Bit position	1	2	3	4	5	6	7	8	9	10	11	12

Example (continued)

Choose 1 for P₈

Use	P ₁	P ₂	D ₃	P ₄	D ₅	D ₆	D ₇	P ₈	D ₉	D ₁₀	D ₁₁	D ₁₂
	0	1	0	1	0	1	0	1	1	1	1	0
Scope of P ₁	✓		✓		✓		✓		✓		✓	
Scope of P ₂		✓	✓			✓	✓			✓	✓	
Scope of P ₄				✓	✓	✓	✓					✓
Scope of P ₈								✓	✓	✓	✓	✓
Bit position	1	2	3	4	5	6	7	8	9	10	11	12

General properties of Hamming code

- It can be used for any code words with m information bits. It uses k parity bits such that $m \leq 2^k - k - 1$.
- By adding an additional parity bit to a Hamming code, we will be able to achieve single-error correction and double-error detection.

Check sum digit is inserted to satisfy the relation:
 $\text{ZIP digit sum} + \text{check sum digit} = 0 \text{ modulo } 10$
to make error correction possible. (Error detection is achieved by using the 2-out-of-5 code of individual digit)

