

Switching Circuits & Logic Design

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University



Fall 2012

1

§1 Number Systems and Conversion

1	𐎶	11	𐎶𐎵	21	𐎶𐎵𐎶	31	𐎶𐎵𐎶𐎶	41	𐎶𐎵𐎶𐎶𐎶	51	𐎶𐎵𐎶𐎶𐎶𐎶
2	𐎶𐎶	12	𐎶𐎵𐎶𐎶	22	𐎶𐎵𐎶𐎶𐎶	32	𐎶𐎵𐎶𐎶𐎶𐎶	42	𐎶𐎵𐎶𐎶𐎶𐎶𐎶	52	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶
3	𐎶𐎶𐎶	13	𐎶𐎵𐎶𐎶𐎶	23	𐎶𐎵𐎶𐎶𐎶𐎶	33	𐎶𐎵𐎶𐎶𐎶𐎶𐎶	43	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶	53	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶
4	𐎶𐎶𐎶𐎶	14	𐎶𐎵𐎶𐎶𐎶𐎶	24	𐎶𐎵𐎶𐎶𐎶𐎶𐎶	34	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶	44	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶	54	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
5	𐎶𐎶𐎶𐎶𐎶	15	𐎶𐎵𐎶𐎶𐎶𐎶𐎶	25	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶	35	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶	45	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	55	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
6	𐎶𐎶𐎶𐎶𐎶𐎶	16	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶	26	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶	36	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	46	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	56	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
7	𐎶𐎶𐎶𐎶𐎶𐎶𐎶	17	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶	27	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	37	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	47	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	57	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
8	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	18	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	28	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	38	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	48	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	58	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
9	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	19	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	29	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	39	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	49	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	59	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
10	𐎶	20	𐎶𐎵	30	𐎶𐎵𐎶	40	𐎶𐎵𐎶𐎶	50	𐎶𐎵𐎶𐎶𐎶		

Babylonian number system (3100 B.C.)



Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
 - Negative number representation
 - Signed magnitude
 - 1's complement
 - 2's complement
 - Arithmetic under 1's and 2's complements
- Binary codes

3

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
 - Negative number representation
 - Signed magnitude
 - 1's complement
 - 2's complement
 - Arithmetic under 1's and 2's complements
- Binary codes

4

Digital Systems and Switching Circuits

- Given some inputs, a digital system produces some outputs
 - It may or may not have “memory” (i.e., its outputs may or may not depend on the *history* of the inputs)



x_i and y_j are often binary, i.e., $x_i, y_j \in \{0,1\}$

5

Digital Systems and Switching Circuits

- Given a specification constraining the input and output relation of a digital system, we'll learn how to design a logic circuit that implements the specification

6

Digital Systems and Switching Circuits

- A digital system usually contains **datapath** and **control** parts

- Datapath part

- Performs arithmetic operations, data transfer, etc.
 - Unit 1 introduces digital *data representation* and *arithmetic operation*

- Control part

- Controls when and where data should be ready in the datapath
 - Later we'll study *combinational* and *sequential circuits* for control logic

7

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
 - Negative number representation
 - Signed magnitude
 - 1's complement
 - 2's complement
 - Arithmetic under 1's and 2's complements
- Binary codes

8

Number Systems and Conversion

□ Decimal number (base 10):

$$\begin{aligned} & \blacksquare 953.78_{10} \\ & = 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2} \end{aligned}$$

□ Binary number (base 2):

$$\begin{aligned} & \blacksquare 1011.11_2 \\ & = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ & = 11.75_{10} \end{aligned}$$

□ Octal number (base 8):

$$\begin{aligned} & \blacksquare 147.3_8 \\ & = 1 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} \\ & = 103.375_{10} \end{aligned}$$

□ Hexadecimal number (base 16):

$$\begin{aligned} & \blacksquare A2F_{16} \\ & = 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0 \\ & = 2607_{10} \end{aligned}$$

The decimal, binary, octal, and hexadecimal points separate positive and negative powers of 10, 2, 8, 16, respectively.

9

Number Systems and Conversion

Generalized Radix System

□ Any integer $R > 1$ can be chosen as the **radix** or **base**

■ Base (radix) R : $(0, 1, \dots, R-1)$, where $R > 1$

E.g.,

■ Base 5: $(0, 1, 2, 3, 4)$

■ Base 16: $(0, 1, 2, \dots, 9, A, B, C, D, E, F)$

□ $N_{(10)}$

$$\begin{aligned} & = (a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3})_R \\ & = a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0 + a_{-1} \times R^{-1} + a_{-2} \times R^{-2} \\ & \quad + a_{-3} \times R^{-3} \end{aligned}$$

Note that R is in base 10!

E.g.,

$$\begin{aligned} & \blacksquare A2F_{16} \\ & = 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0 \\ & = 2607_{10} \end{aligned}$$

10

Number Systems and Conversion

Generalized Radix System

Name	Decimal	Binary	Octal	Hexadecimal
Base	10	2	8	16
Digits	0,1,2,3,4,5,6,7,8,9	0,1	0,1,2,3,4,5,6,7	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
Numbers	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	0 1 10 11 100 101 110 111 1000 1001 1010 1011 1100 1101 1110 1111 10000	0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20	0 1 2 3 4 5 6 7 8 9 A B C D E F 10

11

Number Systems and Conversion

Integer Conversion by Division

- $$\square N = (a_n a_{n-1} \dots a_2 a_1 a_0)_R$$

$$= a_n \times R^n + a_{n-1} \times R^{n-1} + \dots + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0$$
- $$\square N/R = a_n \times R^{n-1} + a_{n-1} \times R^{n-2} + \dots + a_2 \times R^1 + a_1 = Q_1$$

remainder a_0
- $$\square Q_1/R = a_n \times R^{n-2} + a_{n-1} \times R^{n-3} + \dots + a_3 \times R^1 + a_2 = Q_2$$

remainder a_1
- $$\square Q_2/R = a_n \times R^{n-3} + a_{n-1} \times R^{n-4} + \dots + a_4 \times R^1 + a_3 = Q_3$$

remainder a_2

12

Number Systems and Conversion

Exercise

□ Convert 53_{10} to binary

$$\begin{array}{rcl} 2 & \overline{) 53} & \\ 2 & \overline{) 26} & \dots \text{remainder } 1 = a_0 \\ 2 & \overline{) 13} & \dots \text{remainder } 0 = a_1 \\ 2 & \overline{) 6} & \dots \text{remainder } 1 = a_2 \\ 2 & \overline{) 3} & \dots \text{remainder } 0 = a_3 \\ 2 & \overline{) 1} & \dots \text{remainder } 1 = a_4 \\ & 0 & \dots \text{remainder } 1 = a_5 \end{array}$$

$$\text{So } 53_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = 110101_2$$

13

Number Systems and Conversion

Fraction Conversion by Multiplication

$$\begin{aligned} \square F &= (.a_{-1}a_{-2}a_{-3} \dots a_{-m})_R \\ &= a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + a_{-3} \times R^{-3} + \dots + a_{-m} \times R^{-m} \end{aligned}$$

$$\begin{aligned} \square F \cdot R &= a_{-1} + a_{-2} \times R^{-1} + a_{-3} \times R^{-2} + \dots + a_{-m} \times R^{-m+1} \\ &= a_{-1} + F_1 \end{aligned}$$

$$\begin{aligned} \square F_1 \cdot R &= a_{-2} + a_{-3} \times R^{-1} + \dots + a_{-m} \times R^{-m+2} \\ &= a_{-2} + F_2 \end{aligned}$$

$$\begin{aligned} \square F_2 \cdot R &= a_{-3} + \dots + a_{-m} \times R^{-m+3} \\ &= a_{-3} + F_3 \end{aligned}$$

14

Number Systems and Conversion

Exercise

□ Convert 0.625_{10} to binary

$$F = .625$$

$$F1 = .250$$

$$F2 = .500$$

$$\begin{array}{r} \times 2 \\ \hline 1.250 \end{array}$$

a_{-1}

$$\begin{array}{r} \times 2 \\ \hline 0.500 \end{array}$$

a_{-2}

$$\begin{array}{r} \times 2 \\ \hline 1.000 \end{array}$$

a_{-3}

$$\text{So } 0.625_{10} = (. a_{-1} a_{-2} a_{-3})_2 = 0.101_2$$

15

Number Systems and Conversion

Exercise

□ Convert 0.7_{10} to binary

$$\begin{array}{r} 0.7 \\ \times 2 \\ \hline \end{array}$$

a_{-1} (1).4

$$\begin{array}{r} \times 2 \\ \hline \end{array}$$

a_{-2} (0).8

$$\begin{array}{r} \times 2 \\ \hline \end{array}$$

a_{-3} (1).6

$$\begin{array}{r} \times 2 \\ \hline \end{array}$$

a_{-4} (1).2

$$\begin{array}{r} \times 2 \\ \hline \end{array}$$

a_{-5} (0).4

$$\begin{array}{r} \times 2 \\ \hline \end{array}$$

a_{-6} (0).8

repeat

$$\text{So } 0.7_{10} = (0.1 \underline{0110} \underline{0110} \underline{0110} \dots)_2$$

Not finite conversion!

16

Number Systems and Conversion

Exercise

□ Convert 231.3_4 to base 7

$$231.3_4 = 2 \times 4^2 + 3 \times 4^1 + 1 \times 4^0 + 3 \times 4^{-1} = 45.75_{10}$$

$$\begin{array}{r} 7 \overline{) 45} \\ 7 \overline{) 6} \quad \dots \text{remainder } 3 = a_0 \\ 0 \quad \dots \text{remainder } 6 = a_1 \end{array}$$

$$\begin{array}{r} 0.75 \\ \times 7 \\ \hline (5).25 \\ \times 7 \\ \hline (1).75 \end{array}$$

← repeat

a_{-1}

a_{-2}

$$\text{So } 231.3_4 = (63.\underline{51} \underline{51} \dots)_7$$

Decimal system is useful as an intermediate representation for such conversion (though not necessary)

17

Number Systems and Conversion

Exercise

□ Conversion from binary to Octal

$$\begin{aligned} &101110.0011_2 \\ &= \underline{101} \underline{110} \underline{001} \underline{100}_2 \\ &= 56.14_8 \end{aligned}$$

□ Conversion from binary to Hexadecimal

$$\begin{aligned} &1001101.010111_2 \\ &= \underline{0100} \underline{1101} \underline{0101} \underline{1100}_2 \\ &= 4D.5C_{16} \end{aligned}$$

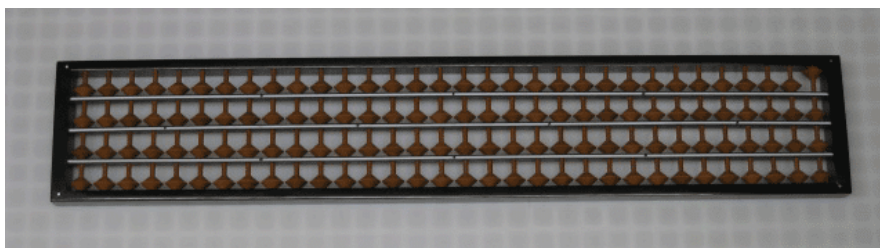
18

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
 - Negative number representation
 - Signed magnitude
 - 1's complement
 - 2's complement
 - Arithmetic under 1's and 2's complements
- Binary codes

19

Binary Arithmetic



extracting the square root of $1/2$ on a binary abacus

Binary Arithmetic

Division

□ Division of 145_{10} by 11_{10} in binary

$$\begin{array}{r} 1101 \text{ quotient} \\ 1011 \overline{) 10010001} \\ \underline{1011} \\ 1110 \\ \underline{1011} \\ 1101 \\ \underline{1011} \\ 1011 \\ \underline{1011} \\ 10 \text{ remainder} \end{array}$$

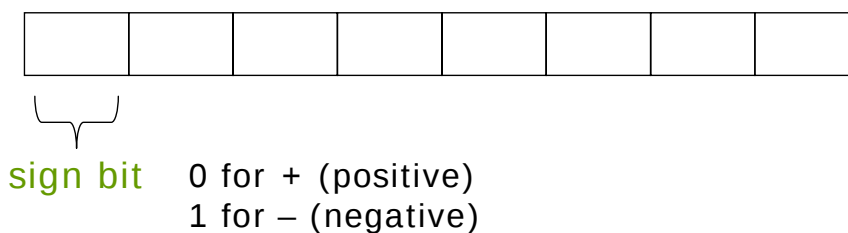
25

Negative Number Representation

□ The 1st bit (most significant bit, MSB) is reserved for sign indication

■ Assume base-2 system

An 8-bit **word** example



(Modern computers are of word size 64 or 32 bits.)

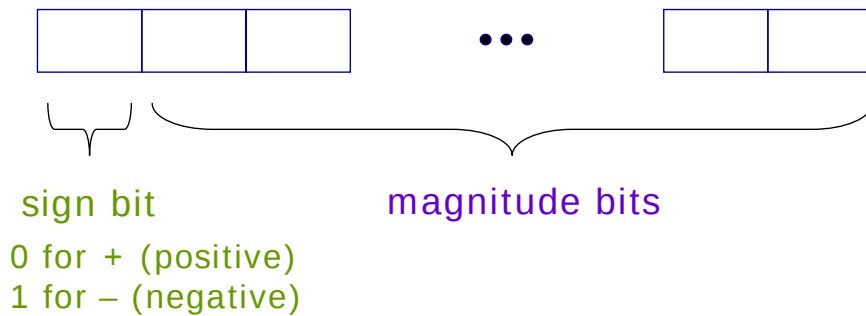
26

Negative Number Representation

Signed Magnitude Numbers

$$\square N^{\text{sm}} = (a_{n-1}a_{n-2} \cdots a_1a_0)_2$$

$$= (-1)^{a_{n-1}} \cdot (a_{n-2} \times 2^{n-2} + \cdots + a_1 \times 2^1 + a_0 \times 2^0)$$



E.g., -12 = 11100 in a 5-bit word = 101100 in a 6-bit word

27

Negative Number Representation

1's Complement

■ For an n-bit word, the negative number $-N$ of N in the 1's complement is

$$N^{(1)} = (2^n - 1) - N$$

- Both N and $-N$ are in the 1's complement (N can be positive/negative)
- $N^{(1)}$ is the bitwise complement of N
- $N = (2^n - 1) - N^{(1)}$

E.g., $n = 6$ and $N = 010101$ (= 21)

$$\begin{array}{rcl}
 (2^n - 1) & = & 111111 \\
 - \quad N & = & 010101 \\
 \hline
 N^{(1)} & = & 101010 \text{ (= -21)}
 \end{array}$$

28

Negative Number Representation 2's Complement

- For an n-bit word, the **negative** number $-N$ of N in the 2's complement is
$$N^{(2)} = 2^n - N = N^{(1)} + 1$$
 - Both N and $-N$ are in the 2's complement (N can be positive/negative)
 - Mind the marginal case when $N = 0$ and $N = -2^{n-1}$
 - $N^{(2)}$ is **the bitwise complement of N plus 1**
 - Effectively, only complementing every bit of N starting from the MSB until the last bit with value 1 (**not applicable when $N = 0$**)

E.g., $n = 6$, $N = 0011\mathbf{10}$ (14)



last bit with value 1

$\Rightarrow N^{(2)} = 110001 + 000001 = 1100\mathbf{10}$ (-14)

29

Negative Number Representation 2's Complement

- The value of $N^{(2)}$ can be obtained from N

E.g., $n = 6$

$N^{(2)} = 101010$ (?) $\Rightarrow N = 010110$ (22) $\Rightarrow N^{(2)}$ represents the negative number **-22**

$N^{(2)} = 100000$ (**-32**) $\Rightarrow$ no corresponding N !

$N = 000000$ (**0**) $\Rightarrow$ no corresponding $N^{(2)}$!

30

Negative Number Representation

- The signed magnitude, 1's complement, and 2's complement systems all have the same representation for any **nonnegative** number
 - For a negative number, we need to specify which system we use (but not for a positive number)

E.g., $n = 6$

$N^{(1)}, N^{(2)}, N^{sm} = 011111$ (31)
 $\Rightarrow N = 011111$ (31)

31

Negative Number Representation

Signed binary integers (word length: $n = 4$)

+N	Positive integers (same for all systems!)	-N	Negative integers		
			Sign and magnitude N^{sm}	2's complement $N^*, N^{(2)}$	1's complement $\bar{N}, N^{(1)}$
+0	0000	-0	1000	-----	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	-----	1000	-----

32

Negative Number Representation Arithmetic with 2's Complement

- ❑ Subtraction is replaced with addition
- ❑ Addition is computed as if its two operands were positive numbers
- Carry from the sign bit is ignored (bit-length preserved)

E.g.,

$$\begin{array}{r} -5 \quad 1011 \\ +6 \quad 0110 \\ \hline +1 \quad (1)0001 \end{array}$$

discard the carry

- Result is correct when no **overflow** occurs (why?)

- ❑ Overflow occurs when

1. adding two positive numbers yields a negative number, or
2. adding two negative numbers yields a positive number.

E.g.,

$$\begin{array}{r} +5 \quad 0101 \\ +6 \quad 0110 \\ \hline 1011 \end{array}$$

wrong! (overflow)

$$\begin{array}{r} -5 \quad 1011 \\ -6 \quad 1010 \\ \hline (1)0101 \end{array}$$

wrong! (overflow)

33

Negative Number Representation Arithmetic with 2's Complement

A B A+B overflow

+	+	+	no
+	+	−	yes
+	−	+ / −	no
−	+	+ / −	no
−	−	+	yes
−	−	−	no

Checking sign-bits for overflow detection

34

Negative Number Representation

Arithmetic with 1's Complement

- Similar to the addition of 2's complement except for having **end-around carry** (instead of discarding the last carry)

- Add the last carry to the sum

E.g.,

$$\begin{array}{r} -5 \quad 1010 \\ +6 \quad 0110 \\ \hline (1) 0000 \\ \quad \rightarrow 1 \quad (\text{end-around carry}) \\ \hline 0001 \end{array}$$

$$\begin{array}{r} +5 \quad 0101 \\ +6 \quad 0110 \\ \hline 1011 \\ \text{wrong! (overflow)} \end{array}$$

$$\begin{array}{r} -5 \quad 1010 \\ -6 \quad 1001 \\ \hline (1) 0011 \\ \quad \rightarrow 1 \\ \hline 0100 \quad \text{wrong! (overflow)} \end{array}$$

35

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
 - Negative number representation
 - Signed magnitude
 - 1's complement
 - 2's complement
 - Arithmetic under 1's and 2's complements
- Binary codes

36

Binary Codes



Photo: datapeak.net

37

Binary Codes

- Use binary codes to represent *symbols*
 - Not associated with arithmetic
 - Different from the *binary number system*

E.g.,

3.1415

└───┴───┴───┴───┴───┘

0011 . 0001 0100 0001 0101

in Binary Coded Decimal (BCD)

Different from converting the number as a whole into binary!

38

Binary Codes

Common binary codes for decimal digits

Decimal digit	8-4-2-1 code (BCD)	6-3-1-1 code	Excess-3 code (BCD+3)	2-out-of-5 code (good for error checking)	Gray code (good for low power and reliability)
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

$a_3a_2a_1a_0$ in $w_3-w_2-w_1-w_0$ (4-bit weighted) code represents $N = w_3a_3 + w_2a_2 + w_1a_1 + w_0a_0$
E.g., 1011 in 6-3-1-1 code represents $N = 6 \cdot 1 + 3 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 8$

39

Binary Codes

□ ASCII code

- 7 bits

- Used to represent computer characters

40

Binary Codes

ASCII printable characters (decimal code, character)					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	delete

41

Other Binary Codes

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	• ■■	U	• • ■■
B	■■■ •	V	• • • ■■
C	■■■ • •	W	• ■■ ■■
D	■■■ • •	X	■■■ • • ■■
E	•	Y	■■■ • ■■ ■■
F	• • ■■ •	Z	■■■ ■■ • •
G	■■■ ■■ •		
H	• • • •		
I	• •		
J	• ■■ ■■ ■■		
K	■■■ • ■■		
L	• ■■ • •		
M	■■■ ■■		
N	■■■ •		
O	■■■ ■■ ■■		
P	• ■■ ■■ •		
Q	■■■ ■■ • ■■		
R	• ■■ •		
S	• • •		
T	■■■		
		1	• ■■ ■■ ■■
		2	• • ■■ ■■ ■■
		3	• • • ■■ ■■
		4	• • • • ■■
		5	• • • • •
		6	■■■ • • • •
		7	■■■ ■■ • • •
		8	■■■ ■■ ■■ • •
		9	■■■ ■■ ■■ ■■ •
		0	■■■ ■■ ■■ ■■ ■■

42

Other Binary Codes

