BINARY SYSTEMS

ATTY. MANUEL O. DIAZ JR.

What does 1,378 mean?

- 1,378 = 1,000 + 300 + 70 + 8
- 1,378 = $1 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 8 \times 10^0$
- The convention, however, is to write only the coefficients & from their position deduce the necessary powers of 10:
- $a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}a_{-4}$
- $a_j \in \{0, 1, 2, \ldots, 9\}$ and j gives the place value and hence the power of 10 by which a_j must be multiplied
- The decimal number system is said to be of *base* or *radix* 10 because it uses the 10 digits and the coefficients are multiplied by powers of 10.

Binary Number System

- Uses only two distinct values
- $11010.11 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2}$
- $11010.11_2 = 26.75_{10}$
- In general, a number expressed in *base r* system has coefficients multiplied by powers of r:
- $a_n r^n + a_{n-1} r^{n-1} + \ldots + a_1 r + a_0 + a_{-1} r^{-1} + a_{-2} r^{-2} + \ldots + a_{-m} r^{-m}$
- The coefficients a_j range in value from 0 to $r - 1$.

Arithmetic Operations

- Arithmetic operations with numbers in base *r* follow the same rules as for decimal numbers.
- One must be careful to use only the *r* allowable digits.
- The sum of two binary numbers is calculated by the same rules as in decimal, except that the digits of the sum in any significant position can be only 0 or 1.
- Any *carry* obtained in a given significant position is used by the pair of digits one significant position higher.

Arithmetic Examples

101101	Augend	101101	Minuend
+100111	Addend	-100111	Subtrahend
1010100	Sum	000110	Difference

Subtraction is slightly more complicated.

The rules are still the same, except that the *borrow* in a given significant position adds 2 to a minuend digit.

Multiplication is very simple.

The multiplier digits are always 1 or 0. Hence, the partial products are equal to the multiplicand or to 0.

Number Base Conversion

- A binary number can be converted to decimal by forming the sum of the powers of 2 of those coefficients whose value is 1:
- $1010.011_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = 10.375_{10}$
- Similarly, a number expressed in base *r* can be converted to its decimal equivalent by multiplying each coefficient with the corresponding power of *r* and adding:
- $630.4_8 = 6 \times 8^2 + 3 \times 8 + 4 \times 8^{-1} = 408.5_{10}$

Decimal-to-Base *r* Conversion

- The conversion from decimal to binary or to any other base *r* system is more convenient if the number is separated into an *integer part* and a *fraction part* and the conversion of each part done separately.
- The conversion of the integer part is done as follows:

1. Divide it by *r* to get an integer quotient a_1 and a remainder b_1.
2. Divide a_1 by *r* to get an integer quotient a_2 and a remainder b_2.
3. Repeat until the integer quotient $a_n = 0$.

Example

Convert 41_{10} to binary:

Integer	*Remainder*
41	
20	1
10	0
5	0
2	1
1	0
0	1

Answer: 101001

Octal and Hexadecimal Numbers

- $2^3 = 8$ *and* $2^4 = 16$
- Each octal digit corresponds to three binary digits
- Each hexadecimal digit corresponds to four binary digits
- The conversion from binary to octal is easily accomplished by partitioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right. The corresponding octal digit is then assigned to each group.

Example

Convert $101100011010 11.111100000110_2$ *to* octal:

10 110 001 101 011 . 111 100 000 110

2 6 1 5 3 . 7 4 0 6

Conversion from binary to hexadecimal is similar except that the binary number is divided into groups of four digits:

10 1100 0110 1011 . 1111 0010

2 C 6 B . F 2

Octal or Hexadecimal to Binary

- Conversion from octal or hexadecimal to binary is done by a reverse procedure.
- Each octal digit is converted to its 3-digit binary equivalent.
- Similarly, each hexadecimal digit is converted to its 4-digit binary equivalent.

Example

673.124_8 *to* binary:

6 7 3 . 1 2 4

110 111 011 . 001 010 100

$306.D_{16}$ *to* binary:

3 0 6 . D

0011 0000 0110 . 1101

Complements

- Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations.
- There are two types of complements for each base-*r* system: (1) the *r*'s complement and (2) the (*r*-1)'s complement.

The r's Complement

- Given a positive number N in base r with an integer part of n digits, the r's complement of N is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$.

Example:

The 10's complement of 52520_{10} is $10^5 - 52520 = 47480$

The 10's Complement

- The 10's complement of a decimal number can be formed by leaving all least significant zeros unchanged, subtracting the first nonzero least significant digit from 10, then subtracting all other higher significant digits from 9.

  Example

  The 10's complement of 25.639 is 74.361

The 2's Complement

- The 2'*s* complement can be performed by leaving all least significant zeros and the first nonzero digit unchanged, and then replacing 1'*s* by 0'*s* and 0'*s* by 1'*s* in all other higher significant digits.

  Example:

  The 2'*s* complement of 101100 *is* 010100.

  The 2'*s* complement of 0.0110 *is* 0.1010.

The $(r - 1)$'s Complement

- Given a positive number N in base r with an integer part of n digits and a fraction part of m digits, the $(r- 1)$'s complement of N is defined as

  $r^n - r^{-m} - N$.

  Example:

  The 9's complement of 52520_{10} is $10^5 - 1 - 52520$ *or* $99999 - 52520 = 47479$.

  The 9's complement of 0.3267_{10} is $1 - 10^{-4} - 0.3267$ *or* $0.9999 - 0.3267 = 0.6732$.

9's and 1's Complement

- The 9's complement of a decimal number is formed simply by subtracting every digit from 9.
- The 1's complement of a binary number is even simpler to form: the 1's are changed to 0's and the 0's to 1's.
- It follows that the r's complement can be obtained from the $(r - 1)$'s complement after the addition of r^{-m} to the least significant digit.
- It is worth mentioning that the complement of the complement restores the number to its original value.

Subtraction with *r*'s Complements

Subtract two positive numbers ($M - N$) both of base r:

1. Add the minuend M to the r's complement of the subtrahend N.
2. Inspect the result obtained in Step 1 for an end carry:

   a. If an end carry occurs, discard it.

   b. If there is no end carry, take the r's complement of the number obtained in Step 1 and place a negative sign in front.

Example 1

Subtract 72532 – 3250 using 10's complement:

The 10's complement of 3250 is 96750 (*add a prefix* 0 *to N* so that *M* and *N* will have the same number of digits).

	72532	*M*
	96750	10's complement of *N*
1	69282	

↳ *Discard the end carry.*

Example 2

Subtract 3250 – 72532 using 10's complement:

The 10's complement of 72532 is 27468.

03250	*M*
27468	10's complement of *N*
30718	

The 10's complement of 30718 *is* 69282.

Hence, the answer is –69282.

Subtraction with (r – 1)'s Complement

1. Add the minuend M to the (r – 1)'s complement of the subtrahend N.
2. Inspect the result obtained in Step 1 for an end carry.
   a. If an end carry occurs, add 1 to the LSD.
   b. If there is no end carry, take the (r – 1)'s complement of the number obtained in Step 1 and place a negative sign in front.

Example 1

Subtract 72532 – 3250 using 9's complement:

The 9's complement of 3250 is 96749 (*add a prefix* 0 *to N* so that *M* and *N* will have the same number of digits).

	72532	*M*
	96749	9's complement of *N*
1	69281	
	1	
	69282	

End around carry.

Example 2

Subtract 3250 – 72532 using 9's complement:

The 9's complement of 72532 is 27467.

03250	*M*
<u>27467</u>	9's complement of *N*
30717	

The 9's complement of 30717 *is* 69282.

Hence, the answer is –69282.

Binary Codes

- A *bit* by definition is a *binary digit*.
- To represent a group of 2^n distinct elements in a binary code requires a minimum of n bits.
- It is possible to arrange n bits in 2^n distinct ways.
- Although the *minimum* number of bits required to code 2^n distinct quantities is n, there is no *maximum* number of bits that may be used for a binary code.

Decimal Codes

DEC	BCD	Excess-3	84-2-1	2421	Biquinary
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000
			<---------------Weighted Codes----------->		

Conversion *vs* Coding

- In both, the final result is a series of bits
- The bits obtained from *conversion* are binary digits
- The bits obtained from *coding* are combinations of 1'*s* and 0'*s* arranged according to the rules of the code used.

Error-Detection Codes

- The *biquinary code* (weighted 5043210) can detect transmission errors by maintaining only two 1's per code word.
- A *parity bit* is an extra bit included with a message to make the total number of 1's either odd or even.
- The Hamming Code

The Hamming Code

ex. data is 1001

1	2	3	4	5	6	7

1	2	3	4	5	6	7

		1		**0**	**0**	**1**
1	2	3	4	5	6	7

The Hamming Code

		1		**0**	**0**	**1**
1	2	3	4	5	6	7
		1+2		1+4	2+4	1+2+4

		1		**0**	**0**	**1**
		1		0		1

0		**1**		**0**	**0**	**1**
0		1		0		1

The Hamming Code

0		**1**		**0**	**0**	**1**
1	2	3	4	5	6	7
		1+2		1+4	2+4	1+2+4

0		**1**		**0**	**0**	**1**
		1			0	1

0	**0**	**1**		**0**	**0**	**1**
	0	1			0	1

The Hamming Code

0	**0**	**1**		**0**	**0**	**1**
1	2	3	4	5	6	7
		1+2		1+4	2+4	1+2+4

0	**0**	**1**		**0**	**0**	**1**
				0	0	1

0	**0**	**1**	**1**	**0**	**0**	**1**
			1	0	0	1

The Hamming Code

- The code word is 0011001.
- Assume for instance that the received codeword is 1011001.
- First, extract the data from 1011001.
- Then, calculate the parity bits:

_ _ 1 _ 0 0 1

0 _ 1 _ 0 0 1

0 0 1 _ 0 0 1

0 0 1 1 0 0 1

The Hamming Code

- Compare the received and calculated code word:

1 0 1 1 0 0 1 Received

0 0 1 1 0 0 1 Calculated

- Obtain the vertical parity:

1 0 1 1 0 0 1 Received

<u>0 0 1 1 0 0 1</u> Calculated

1 0 0 0 0 0 0

- There is an error in bit location 1. Hence, code word should have been 0011001.

Another Example

- Received code word is 0011101.
- Extract the data word from 0011101 and recalculate:

		1		1	0	1
1		1		1		1
	0	1			0	1
			0	1	0	1
1	0	1	0	1	0	1

- The calculated code word is 1010101.

Another Example

- The calculated code word is 1010101.
- Compare the received and calculated code word by finding the vertical parity:

0	0	1	1	1	0	1
1	0	1	0	1	0	1
1	0	0	1	0	0	0
1			4			

- There is an error in the received word in its 5th bit position (1+4=5).

Reflected Code

- A number in the reflected code changes by only one bit as it proceeds from one number to the next.
- Also known as *Gray* code.

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

Alphanumeric Codes

- Internal Code
- ASCII

  (American Standard Code for Info Interchange)
- EBCDIC

  (Extended BCD Interchange Code)

Binary Storage and Registers

- A *binary cell* is a device that possesses two stable states and is capable of storing one bit of information.
- A *register* is a group of binary cells.
- The *state* of a register is an *n*-tuple number of 1's and 0's with each bit designating the state of one cell in the register.
- The *content* of a register is a function of the interpretation given to the information stored in it.

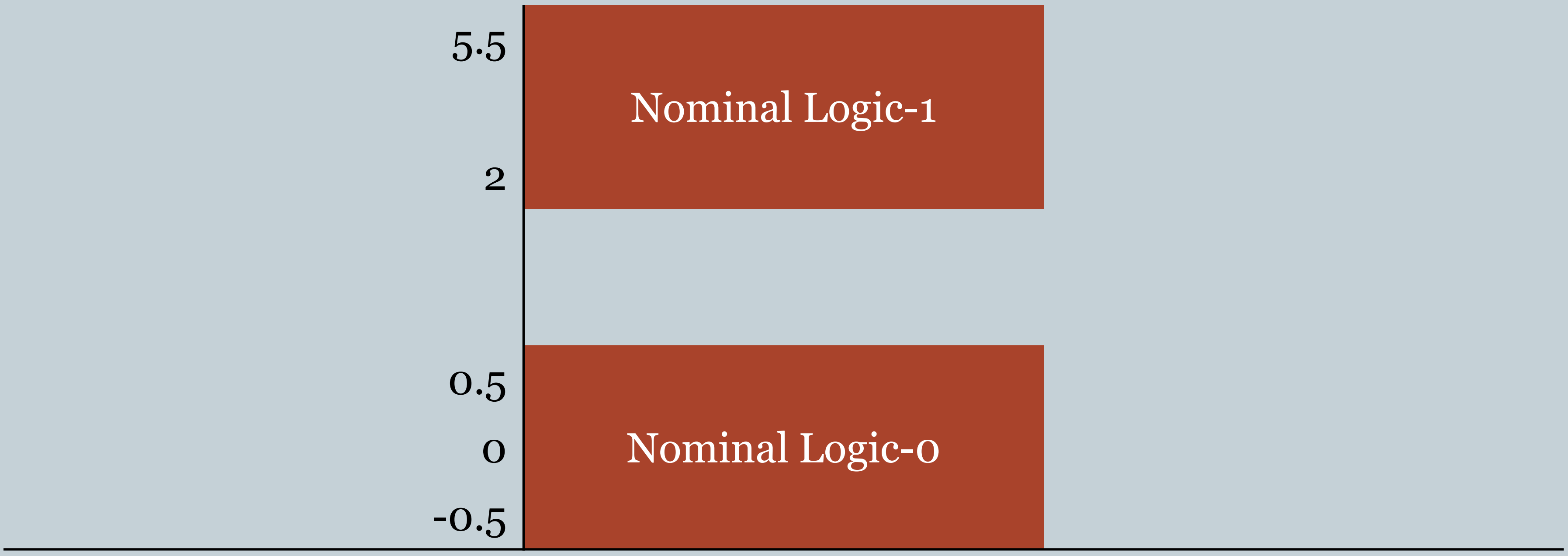
Binary Logic

AND: *the output is* 1 *if and only if all the inputs are* 1, *otherwise the output is* 0.

OR: *the output is* 0 *if and only if all the inputs are* 0, *otherwise the output is* 1.

NOT: *the output is what the input is not*.

Sample Logic Signal



Logic Gates

- Electronic digital circuits are also called *logic circuits* because with proper input, they establish logical manipulation paths.
- *Gates* are blocks of hardware that produce a logic-1 or a logic-0 output signal if input logic requirements are satisfied.
- *Digital circuits*, *switching circuits*, *logic circuits*, and *gates* are four different names used for the same type of circuits.

Integrated Circuits

- An *integrated circuit* is a small silicon semiconductor crystal, called a *chip*, containing electrical components such as transistors, diodes, resistors, and capacitors.
- The various components are interconnected inside the chip to form an electronic circuit.
- The chip is mounted on a metal or plastic package and connections are welded to external pins to form the IC.
- Packaging is either *flat* or *SIP and DIP*.

End of Chapter 1

PLEASE PREPARE FOR A LONG TEST.