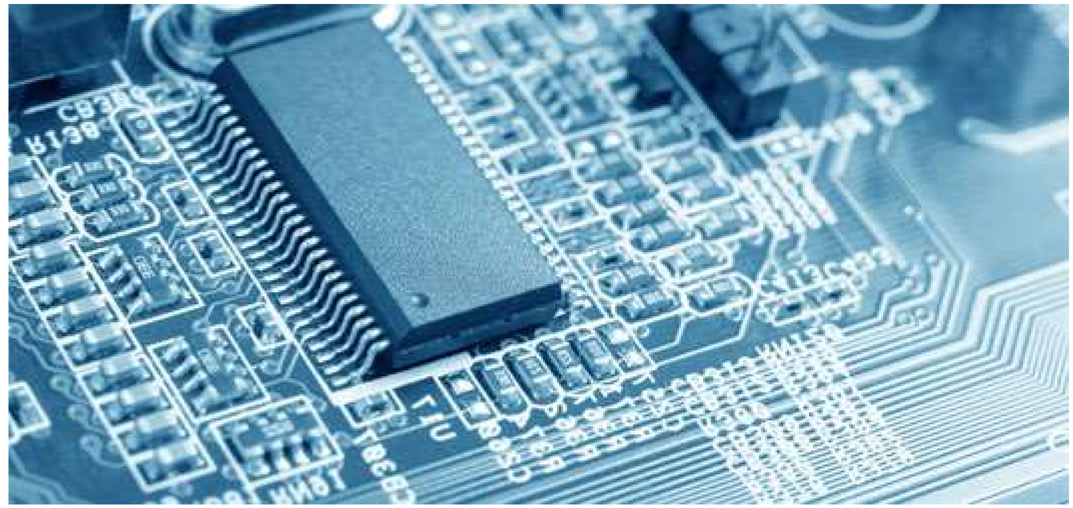
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



ROBT206 – Microcontrollers with Lab

Lectures 2-3 – Number Systems

11-16 January, 2018

Number Systems – Representation

- Positive radix, positional number systems
- A number with *radix* r is represented by a string of digits:

$$A_{n-1}A_{n-2}\ \ldots\ A_1A_0\ .\ A_{-1}\ A_{-2}\ \ldots\ A_{-m+1}\ A_{-m}$$

- in which $0 \le A_i < r$ and **.** is the *radix point.*

- The string of digits represents the power series:

$$(\text{Number})_r = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i\right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j\right)$$

(Integer Portion) + (Fraction Portion)

Number Systems – Examples

	General	Decimal	Binary
Radix (Base)	r	10	2
Digits	0 => r - 1	0 => 9	0 => 1
0	r^0	1	1
1	r^1	10	2
2	r^2	100	4
3	r^3	1000	8
Powers of Radix 4	r^4	10,000	16
5	r^5	100,000	32
-1	r^{-1}	0.1	0.5
-2	r^{-2}	0.01	0.25
-3	r^{-3}	0.001	0.125
-4	r^{-4}	0.0001	0.0625
-5	r^{-5}	0.00001	0.03125

Commonly Occurring Bases

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Numbers in Different Bases

- Good idea to memorize!

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10

Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form Σ (digit $\times$ respective power of 2).

• example: $1011.1 = (1011.1)_2$

$$\begin{array}{ccccccccccc} 1 & & 0 & & 1 & & 1 & . & 1 & & \\ \times\, 2^{3} & & \times\, 2^{2} & & \times\, 2^{1} & & \times\, 2^{0} & & \times\, 2^{-1} & & \\ \hline 8 & + & 0 & + & 2 & + & 1 & + & .5 & = & (11.5)_{10} \end{array}$$

- Convert 11010_2 to N_{10}:

Converting Octal to Decimal

- 8 digits = {0,1,2,3,4,5,6,7}
- example: $(2365.2)_8$

$$\begin{array}{ccccccccc} 2 & & 3 & & 6 & & 5 & . & 2 \\ \times\, 8^3 & & \times\, 8^2 & & \times\, 8^1 & & \times\, 8^0 & & \times\, 8^{-1} \\ \hline 1024 & + & 192 & + & 48 & + & 5 & + & .25 \end{array} = (1269.25)_{10}$$

Converting Hexadecimal to Decimal

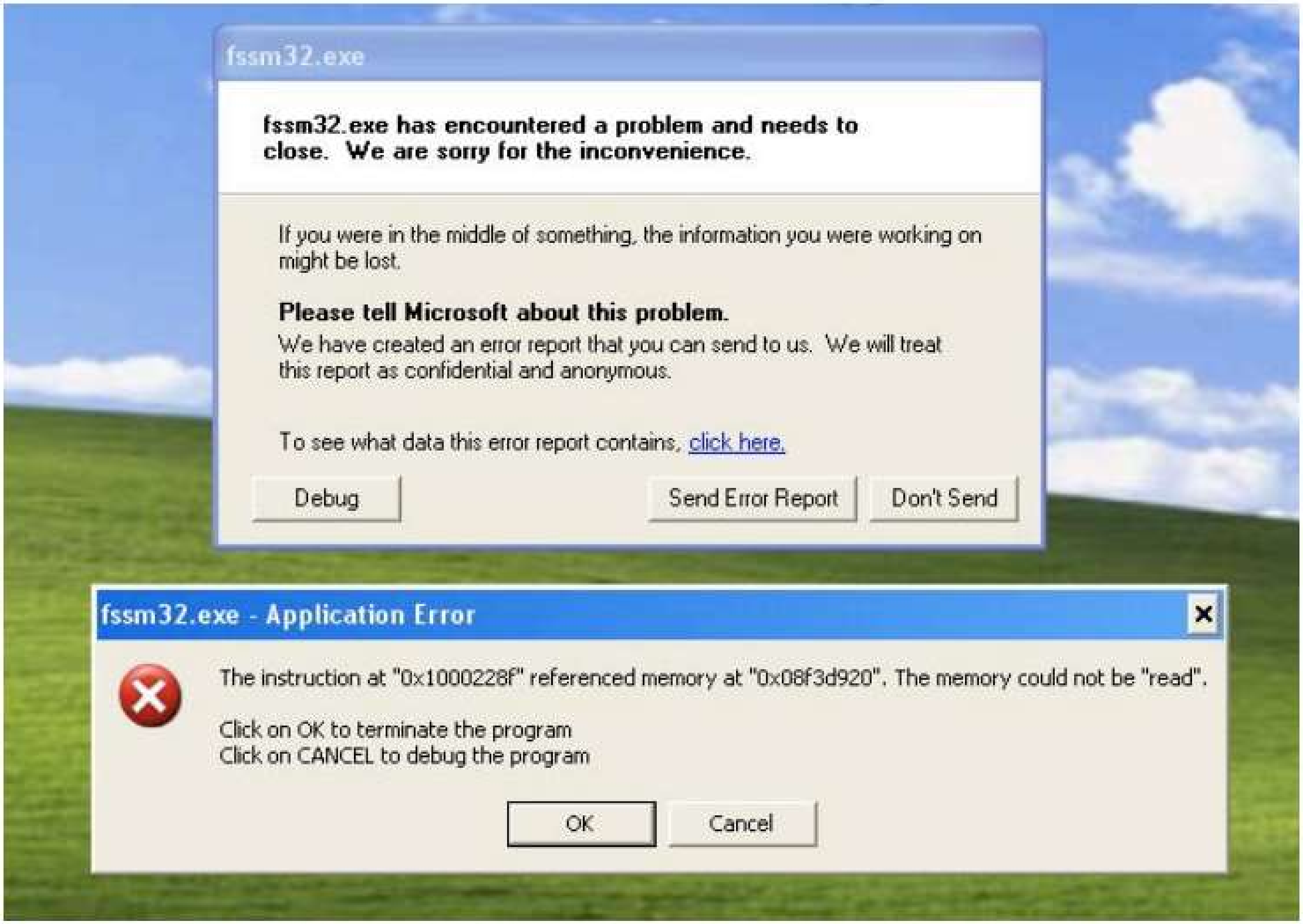
- 16 digits = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

- example: $(26BA)_{16}$ [alternate notation for hex: 0x26BA]

$$
\begin{array}{cccc}
2 & 6 & B & A \\
\times\ 16^3 & \times\ 16^2 & \times\ 16^1 & \times\ 16^0 \\
\hline
8192 + & 1536 + & 176 + & 10 = (9914)_{10}
\end{array}
$$

Why Important: More concise than binary, but related (a power of 2)

Hexadecimal (or hex) is often used for addressing



Conversion Between Bases

To convert from one base to another:

1) Convert the Integer Part

2) Convert the Fraction Part

3) Join the two results with a radix point

Conversion Details

- To Convert the Integral Part:
  - Repeatedly ***divide*** the number by the new radix and save the remainders.
  - The digits for the new radix are the remainders ***in reverse order*** of their computation.
  - If the new radix is > 10, then convert all remainders > 10 to digits A, B, …

- To Convert the Fractional Part:
  - Repeatedly ***multiply*** the fraction by the new radix and save the integer digits that result.
  - The digits for the new radix are the integer digits ***in order*** of their computation.
  - If the new radix is > 10, then convert all integers > 10 to digits A, B, …

Example: Convert 46.6875_{10} To Base 2

- Convert 46 to Base 2
- Convert 0.6875 to Base 2:
- Join the results together with the radix point:

Checking the Conversion

- To convert back, sum the digits times their respective powers of r.
- From the prior conversion of 46.6875_{10}

$$
\begin{aligned}
101110_2 &= 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 \\
&= 32 + 8 + 4 + 2 \\
&= 46 \\
0.1011_2 &= 1/2 + 1/8 + 1/16 \\
&= 0.5000 + 0.1250 + 0.0625 \\
&= 0.6875
\end{aligned}
$$

Octal (Hexadecimal) to Binary and Back

- Octal (Hexadecimal) to Binary:
  - Restate the octal (hexadecimal) as three (four) binary digits starting at the radix point and going both ways.

- Binary to Octal (Hexadecimal):
  - Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part.
  - Convert each group of three bits to an octal (hexadecimal) digit.

Octal to Hexadecimal via Binary

- Convert octal to binary.
- Use groups of <u>four bits</u> and convert as above to hexadecimal digits.
- Example: Octal to Binary to Hexadecimal

$$6 \quad 3 \quad 5 \,.\, 1 \quad 7 \quad 7_{\,8}$$

Base Conversion - Positive Powers of 2

- Useful for Base Conversion

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Exponent	Value
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152

Computer from Digital Perspective

- Information: just sequences of binary (0's and 1's)
  - True = 1, False = 0
- Numbers: converted into binary form when "viewed" by computer
  - e.g., 19 = 10011 (16 (1) + 8 (0) + 4 (0) + 2 (1) + 1 (1)) in binary
- Characters: assigned a specific numerical value (ASCII standard)
  - e.g., 'A' = 65 = 1000001, 'a' = 97 = 1100001
- Text is a sequence of characters:
  - "Hi there" = 72, 105, 32, 116, 104, 101, 114, 101

    = 1001000, 1101001, ...

Terminology: Bit, Byte, Word

- bit = a binary digit e.g., 1 or 0
- byte = 8 bits e.g., 01100100
- word = a group of bits that is **architecture dependent**

  (the number of bits that an architecture can process at once)

  a 16-bit word = 2 bytes e.g., 1001110111000101

  a 32-bit word = 4 bytes e.g., 10011101110001010111011100010 1

OBSERVATION: computers have bounds on how much input they can handle at once → limits on the sizes of numbers they can deal with

Terminology: MSB, LSB

- Bit at the left is highest order, or most significant bit (MSB)
- Bit at the right is lowest order, or least significant bit (LSB)

MSB LSB

e.g., 1001110111000101

- Common reference notation for k-bit value: $b_{k-1}b_{k-2}b_{k-3}...b_1b_0$

Single Bit Binary Addition with Carry

Given two binary digits (X,Y), a carry in (Z) we get the following sum (S) and carry (C):

Carry in (Z) of 0:

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Carry in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Multiple Bit Binary Addition

- Extending this to two multiple bit examples:

Carries

01100	10100
+10001	+10111

Sum

Note: The <u>0</u> is the default Carry-In to the least significant bit.

Single Bit Binary Subtraction with Borrow

- Given two binary digits (X,Y), a borrow in (Z) we get the following difference (S) and borrow (B):
- **Borrow in (Z) of 0:**

Z	**0**	**0**	**0**	**0**
X	**0**	**0**	**1**	**1**
-Y	**-0**	**-1**	**-0**	**-1**
BS	**0 0**	**1 1**	**0 1**	**0 0**

- **Borrow in (Z) of 1:**

Z	**1**	**1**	**1**	**1**
X	**0**	**0**	**1**	**1**
-Y	**-0**	**-1**	**-0**	**-1**
BS	**1 1**	**1 0**	**0 0**	**1 1**

Multiple Bit Binary Subtraction

- Extending this to two multiple bit examples:

Borrows

10110	10110
- 10010	- 10011

Difference

Binary Multiplication

The binary multiplication table is simple:

$$0 * 0 = 0 \ | \ 1 * 0 = 0 \ | \ 0 * 1 = 0 \ | \ 1 * 1 = 1$$

Extending multiplication to <u>multiple digits</u>:

Multiplicand	1011
Multiplier	x 101
Partial Products	1011
	0000 -
	1011 - -
Product	110111

Binary Numbers and Binary Coding

- Flexibility of representation
  - Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.
- Information Types
  - Numeric
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - Non-numeric
    - Greater flexibility since arithmetic operations not applied.
    - Not tied to binary numbers

Non-numeric Binary Codes

- Given n binary digits (called bits), a binary code is a mapping from a set of represented elements to a subset of the 2^n binary numbers.
- Example: A binary code for the seven colors of the rainbow
- Code 100 is not used

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Number of Bits Required

- Given M elements to be represented by a binary code, the minimum number of bits, *n*, needed, satisfies the following relationships:

  $2^n \geq M > 2^{(n-1)}$

  $x = \log_2 M$ where x , called the *ceiling function,*
  is the integer greater than or equal to *n*.

- Example: How many bits are required to represent <u>decimal digits</u> with a binary code?

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that can be represented.
- But, you can represent m elements, m < r^n
- Examples:
  - You can represent 4 elements in radix $r = 2$ with $n = 2$ digits: (00, 01, 10, 11).

Warning: Conversion or Coding?

- Do <u>NOT</u> mix up <u>conversion</u> of a decimal number to a binary number with <u>coding</u> a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ (This is <u>conversion</u>)
- 13 $\Leftrightarrow$ 0001|0011 (This is <u>coding</u>)

Alphanumeric codes – ASCII character codes

- American Standard Code for Information Interchange
- This code is a popular code used to represent information sent as character-based data.
  It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)
- Other non-printing characters are used for record marking and flow control (e.g., STX and ETX start and end text areas).

ASCII Properties

ASCII has some interesting properties:

Digits 0 to 9 span Hexadecimal values 30 to 39.

Upper case A - Z span 41 to 5A.

Lower case a - z span 61 to 7A.

- Lower to upper case translation (and vice versa) occurs by flipping bit 6.

Binary Coded Decimal (BCD)

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

$(185)_{10}$ =

= $(0001\ 1000\ 0101)_{BCD}$

Warning: Conversion or Coding?

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ (This is conversion)
- 13 $\Leftrightarrow$ 0001|0011 (This is coding)

BCD Arithmetic

Given a BCD code, we use binary arithmetic to add the digits:

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)

MORE THAN 9, so must be represented by two digits!

To correct the digit, subtract 10 by adding 6

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)
	+0110	so add 6
	carry = 1 0011	leaving 3 + carry
	0001 | 0011	Final answer (two digits)

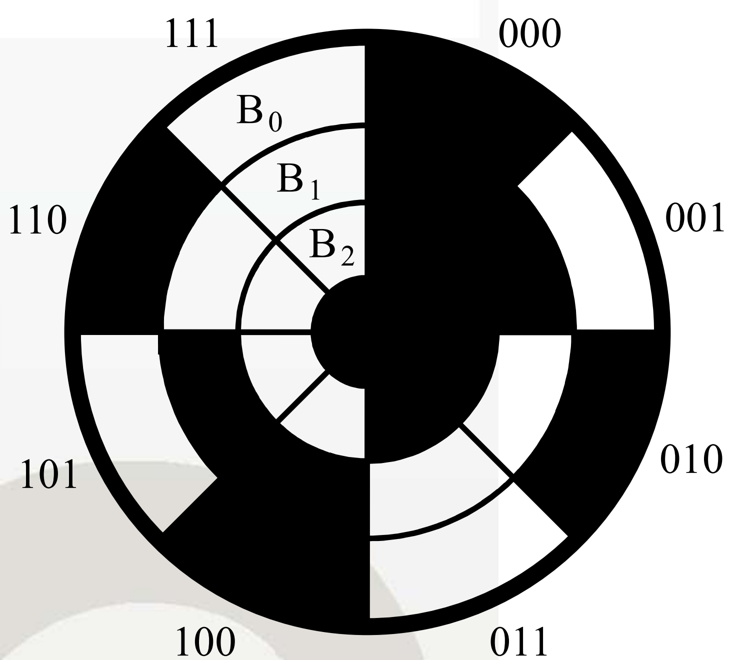
If the digit sum is > 9, add one to the next significant digit

GRAY CODE – Decimal

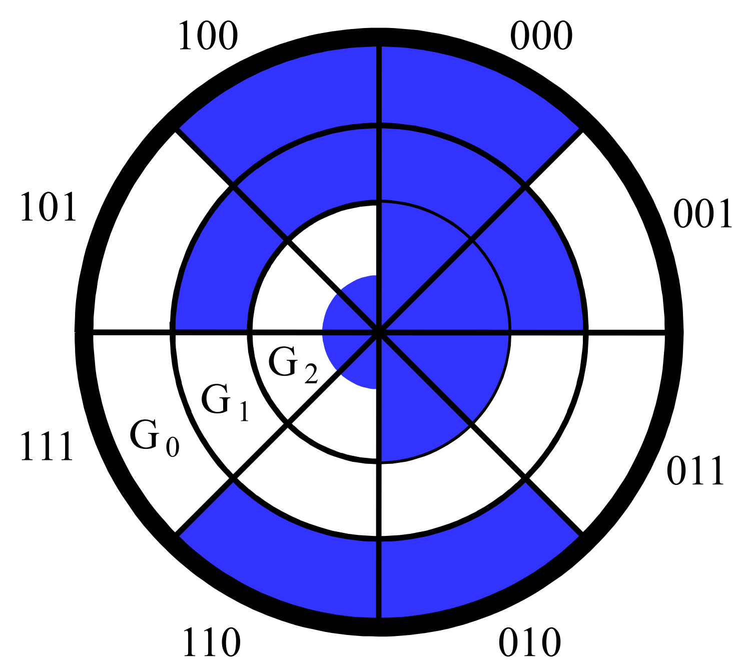
Decimal	BCD	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

Optical Shaft Encoder

- Does this special Gray code property have any value?
- An Example: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

Shaft Encoder (Continued)

- How does the shaft encoder work?

- For the binary code, what codes may be produced if the shaft position lies between codes for 3 and 4 (011 and 100)?

- Is this a problem?

Shaft Encoder (Continued)

- For the Gray code, what codes may be produced if the shaft position lies between codes for 3 and 4 (010 and 110)?
- Is this a problem?
- Does the Gray code function correctly for these borderline shaft positions for all cases encountered in octal counting?

Any Questions?