EEM 232 Digital System I

Instructor : Assist. Prof. Dr. Emin Germen

egermen@anadolu.edu.tr

Course Book : **Logic and Computer Design Fundamentals by Mano & Kime Third Ed/Fourth Ed.. Pearson**

Grading

- 1st Midterm Exam : 15%
- 2nd Midterm Exam : 20%
- Homework & Quiz(es) : 25%
- Final Exam : 40%

- Homework Policy : Must return in 1 week

(If cheating has been detected the grade will be 0)

Lecture Overview

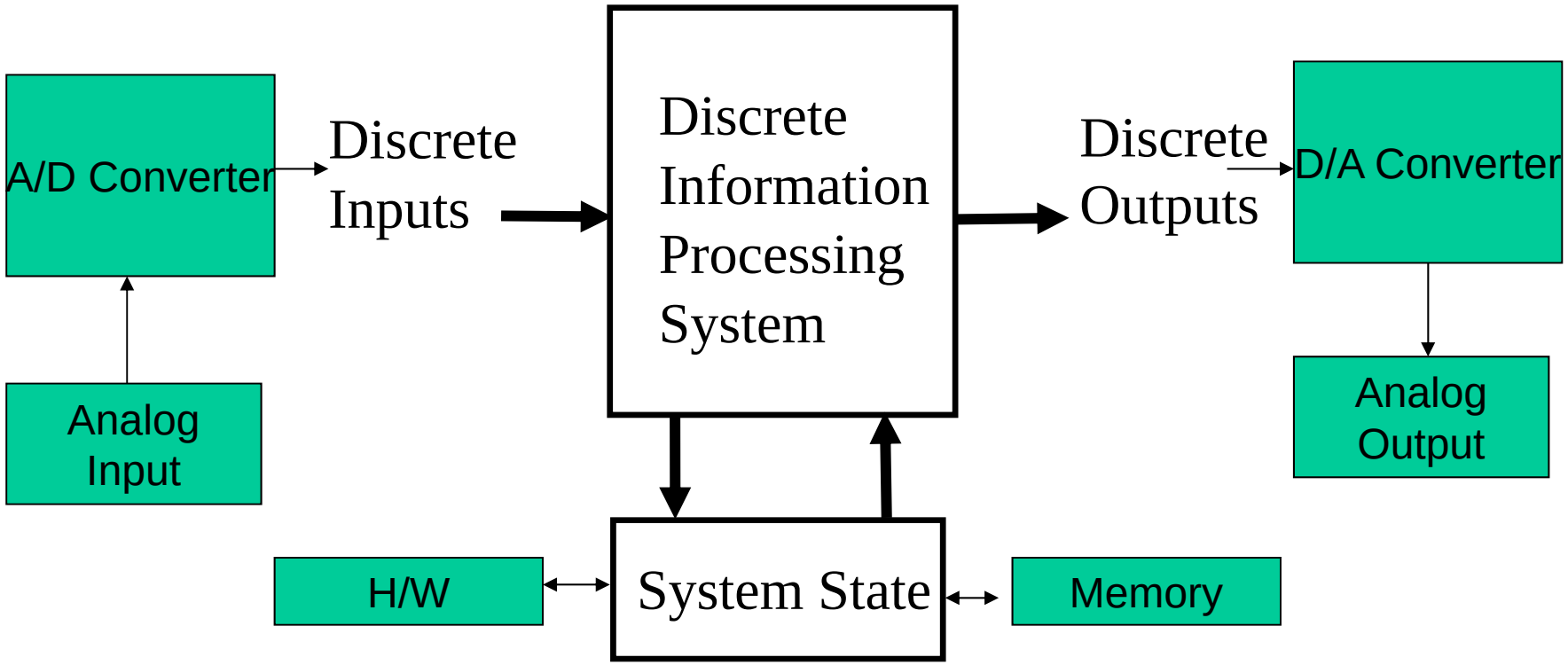
- Week 1 Introduction & Number Systems
- Week 2 Boolean Algebra
- Week 3 Karnaugh Maps
- Week 4 Combinational Logic Design
- Week 5 Combinational Logic Design (Ctd...)
- Week 6 Arithmetic Functions
- Week 7 Sequential Circuits & FlipFlops, Latches
- Week 8 Sequential Circuit Analyses
- Week 9 Sequential Circuits Counters
- Week 10 Sequential Circuit Design
- Week 11 ALU (arithmetic Logic Unit)
- Week 12 RAM

Overview

- **Digital Systems and Computer Systems**
- **Information Representation**
- **Number Systems** [binary, octal and hexadecimal]
- **Arithmetic Operations**
- **Base Conversion**
- **Decimal Codes** [BCD (binary coded decimal), parity]
- **Gray Codes**
- **Alphanumeric Codes**

Digital System

Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.

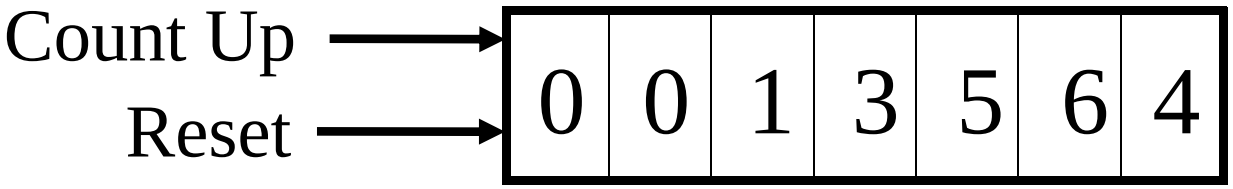


Types of Digital Systems

- No state present
  - Combinational Logic System
  - Output = Function(Input)
- State present
  - State updated at discrete times
    => Synchronous Sequential System
  - State updated at any time
    =>Asynchronous Sequential System
  - State = Function (State, Input)
  - Output = Function (State) or Function (State, Input)

Digital System Example:

A Digital Counter (e. g., odometer):

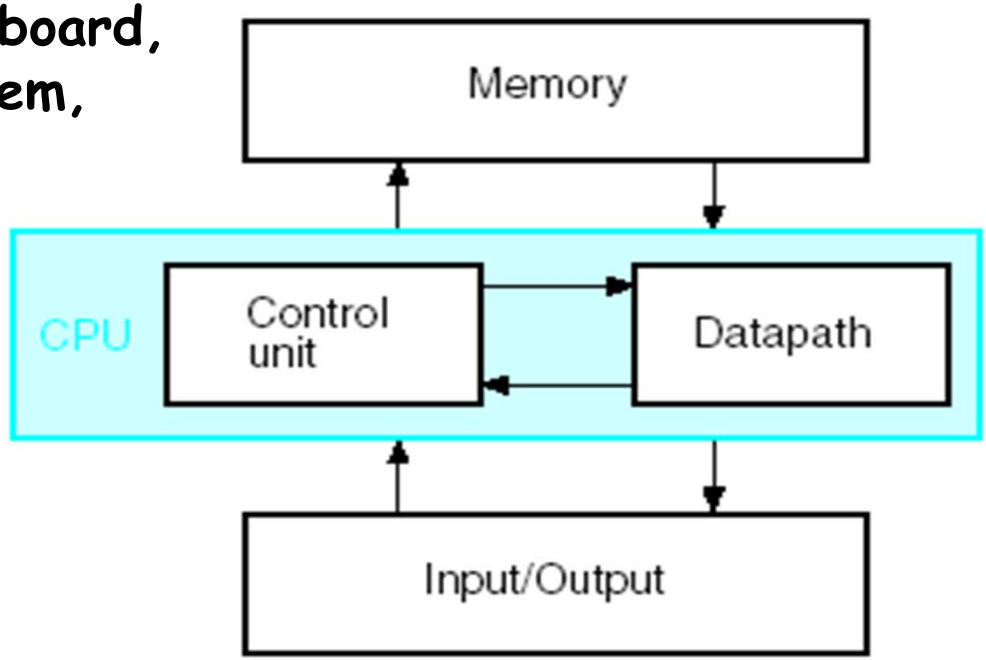


Inputs: **Count Up, Reset**
Outputs: **Visual Display**
State: **"Value" of stored digits**

Synchronous or Asynchronous?

A Digital Computer Example

Inputs: Keyboard, mouse, modem, microphone



Outputs: CRT, LCD, modem, speakers

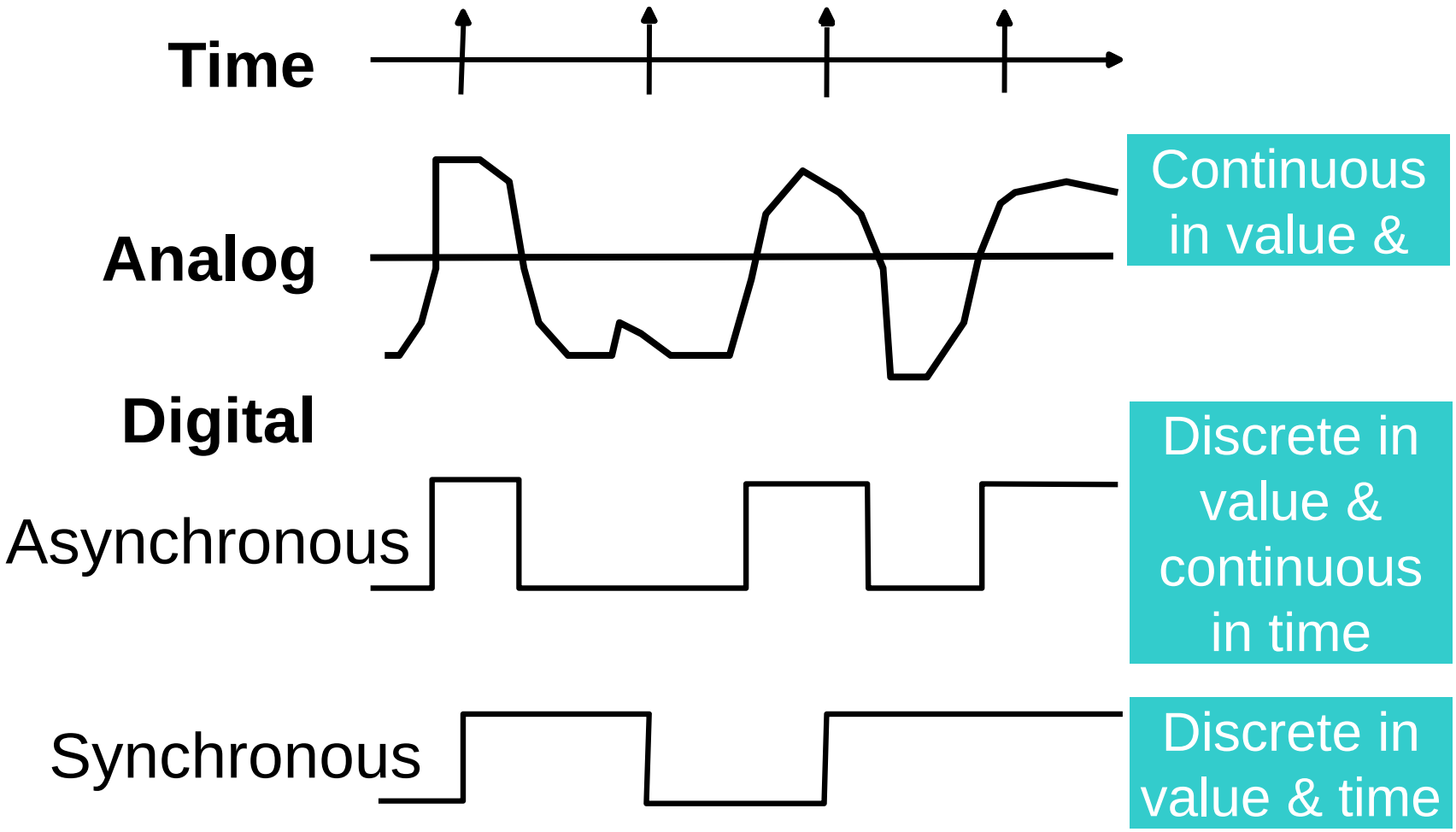
Fig. 1-2 Block Diagram of a Digital Computer

Synchronous or Asynchronous?

Signal

- An information variable represented by physical quantity.
- For digital systems, the variable takes on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities

Signal Examples Over Time



Signal Example – Physical Quantity: Voltage

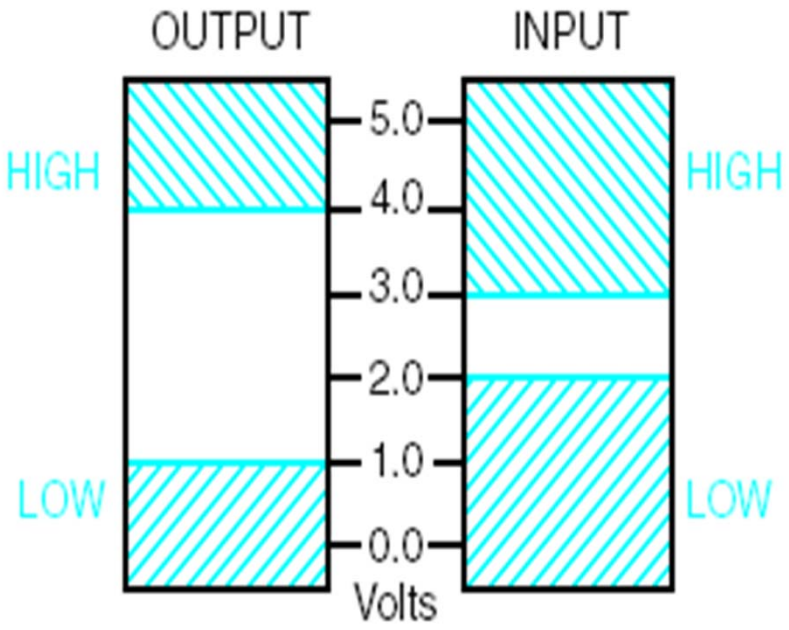


Fig. 1-1 An Example of Voltage Ranges for Binary Signals

Binary Values: Other Physical Quantities

- What are other physical quantities represent 0 and 1?
  - CPU — Voltage
  - Disk — Magnetic Field Direction
  - CD — Surface Pits/Light
  - Dynamic RAM — Electrical Charge

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:

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

$$(\text{Integer Portion}) + (\text{Fraction Portion})$$

Numbers

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

- The . is called the radix point
- A_{n-1} : most significant digit (msd)
- A_{-m} : least significant digit (lsd)

$$(724.5)_{10} = 724.5 = 7\times10^2 + 2\times10^1 + 4\times10^0 + 5\times10^{-1}$$

$$1620.375 = 1\times10^3 + 6\times10^2 + 2\times10^1 + 3\times10^{-1} + 7\times10^{-2} + 5\times10^{-3}$$

$$(312.4)_5 = 3\times5^2 + 1\times5^1 + 2\times5^0 + 4\times5^{-1} = (82.8)_{10}$$

Number Systems – Examples

		General	Decimal	Binary
Radix (Base)		r	10	2
Digits		0 => r - 1	0 => 9	0 => 1
Power of Radix	0	r^0	1	1
	1	r^1	10	2
	2	r^2	100	4
	3	r^3	1000	8
	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

Special Powers of 2

- **2^{10} (1024) is Kilo, denoted "K"**
- **2^{20} (1,048,576) is Mega, denoted "M"**
- **2^{30} (1,073, 741,824)is Giga, denoted "G"**

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

Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form Σ (digit × respective power of 2).
- Example:Convert 11010_2 to N_{10}:

- **Answer : 26**
- **How? 1*16 + 1*8 + 0*4 + 1*2 + 0*1**

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

The six letters (in addition to the 10 integers) in hexadecimal represent:

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

Conversion Between Bases

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 162.375_{10} To Base 2

- Convert 162 to Base 2

- Convert 0.375 to Base 2:

- Join the results together with the radix point:

Conversion from binary to decimal

162 / 2	= 81	rem 0
81 / 2	= 40	rem 1
40 / 2	= 20	rem 0
20 / 2	= 10	rem 0
10 / 2	= 5	rem 0
5 / 2	= 2	rem 1
2 / 2	= 1	rem 0
1 / 2	= 0	rem 1

0.375 x 2 = 0.750
0.750 x 2 = 1.500
0.500 x 2 = 1.000

Additional Issue - Fractional Part

- Note that in this conversion, the fractional part became 0 as a result of the repeated multiplications.
- In general, it may take many bits to get this to happen or it may never happen.
- Example: Convert 0.65_{10} to N_2
  - 0.65 = 0.10100011001001 …
  - The fractional part begins repeating every 4 steps yielding repeating 1001 forever!
- Solution: Specify number of bits to right of radix point and round or truncate to this number.

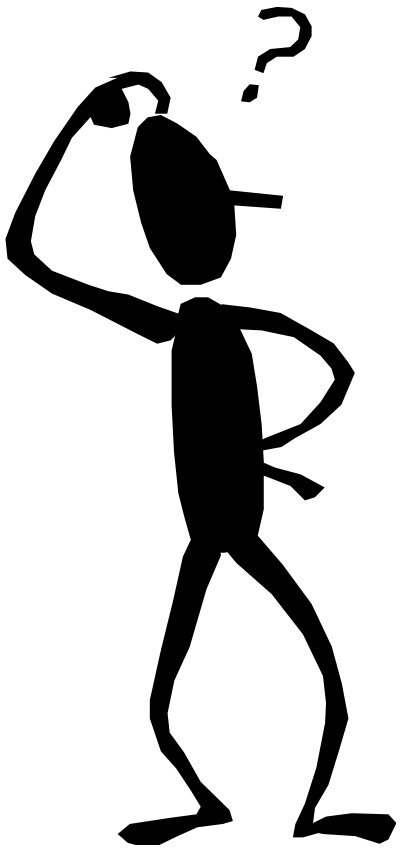
Checking the Conversion

To convert back, sum the digits times their respective powers of r.

- From the prior conversion of 162.375_{10}

$$10100010_2 = 1\cdot128 + 0\cdot64 + 1\cdot32 + 0\cdot16 + 0\cdot8 + 0\cdot4 + 1\cdot2 + 0\cdot1$$
$$= 128 + 32 + 2$$
$$= 162$$
$$0.011_2 = 1/4 + 1/8$$
$$= 0.250 + 0.1250$$
$$= 0.375$$

Why does this work?



- This works for converting from decimal to *any* base
- Why? Think about converting 162.375 from decimal to decimal

$$\begin{aligned} 162 / 10 &= 16 \quad \text{rem } 2 \\ 16 / 10 &= 1 \quad \text{rem } 6 \\ 1 / 10 &= 0 \quad \text{rem } 1 \end{aligned}$$

- Each division strips off the rightmost digit (the remainder). The quotient represents the remaining digits in the number

- Similarly, to convert fractions, each multiplication strips off the leftmost digit (the integer part). The fraction represents the remaining digits

$$\begin{aligned} 0.375 \times 10 &= 3.750 \\ 0.750 \times 10 &= 7.500 \\ 0.500 \times 10 &= 5.000 \end{aligned}$$

Octal and Hexadecimal Numbers

- The octal number system: Base-8
- Eight digits: 0,1,2,3,4,5,6,7

$$(127.4)_8 = 1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1} = (87.5)_{10}$$

- The hexadecimal number system: Base-16
- Sixteen digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

$$(B65F)_{16} = 11\times 16^3 + 6\times 16^2 + 5\times 16^1 + 15\times 16^{-0} = (46687)_{10}$$

- For our purposes, base-8 and base-16 are most useful as a "shorthand"
- notation for binary numbers

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 four bits and convert as above to hexadecimal digits.
- Example: Octal to Binary to Hexadecimal

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

- Why do these conversions work?

A Final Conversion Note

- You can use arithmetic in other bases if you are careful:
- Example: Convert 101110_2 to Base 10 using binary arithmetic:

  Step 1 101110 / 1010 = 100 r 0110

  Step 2 100 / 1010 = 0 r 0100

  Converted Digits are 0100_2 | 0110_2

  or 4 6 $_{10}$

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	100
Indigo	101
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:

$$2^n <= M < 2^{n+1}$$

$$\lceil n \rceil = \log_2 M$$

where $\lceil x \rceil$ called the *ceiling function,* which is the integer greater than or equal to *x*.

- 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).
  - You can represent 4 elements in radix $r = 2$ with $n = 4$ digits: (0001, 0010, 0100, 1000).
  - This second code is called a "one hot" code.

Binary Codes for Decimal Digits

- There are over 8,000 ways that you can chose 10 elements from the 16 binary numbers of 4 bits. A few are useful:

Decimal	8,4,2,1	Excess3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
- This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.
- Example: 1001 (9) = 1000 (8) + 0001 (1)
- How many "invalid" code words are there?
- What are the "invalid" code words?

Excess 3 Code and 8, 4, –2, –1 Code

Decimal	Excess 3	8, 4, –2, –1
0	0011	0000
1	0100	0111
2	0101	0110
3	0110	0101
4	0111	0100
5	1000	1011
6	1001	1010
7	1010	1001
8	1011	1000
9	1100	1111

- What interesting property is common to these two codes?

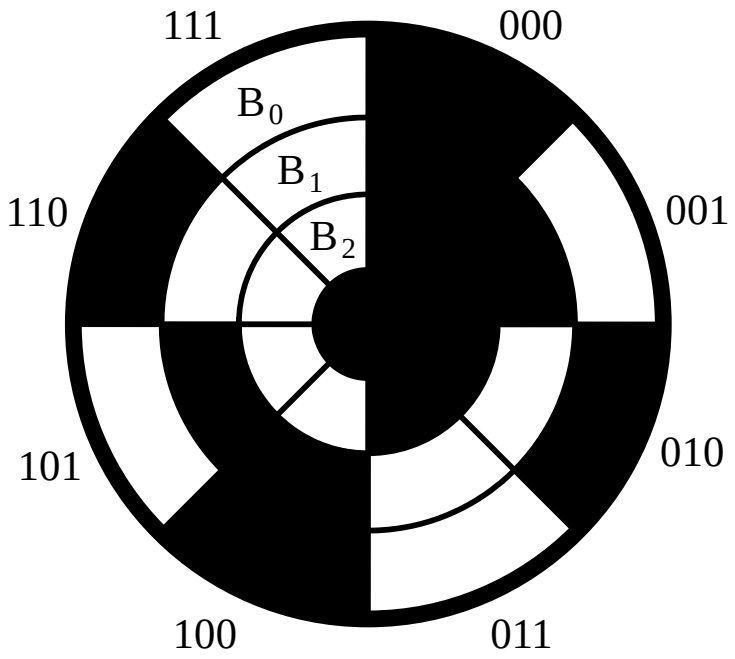
Gray Code

Decimal	8,4,2,1	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

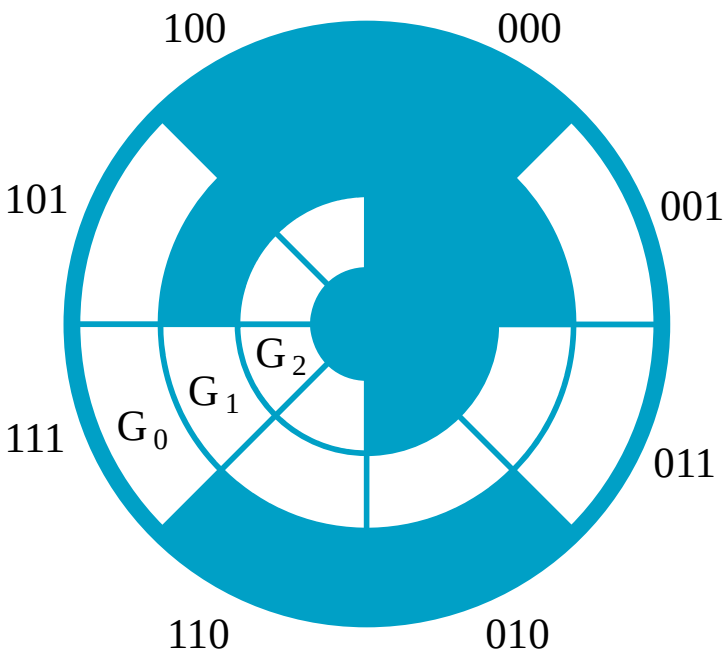
- What special property does the Gray code have in relation to adjacent decimal digits?

Gray Code (Continued)

- 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

Gray Code (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?

Gray Code (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?

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)