Chapter 1

Professor Brendan Morris, SEB 3216, brendan.morris@unlv.edu
http://www.ee.unlv.edu/~b1morris/cpe100/

CPE100: Digital Logic Design I

Section 1004: Dr. Morris

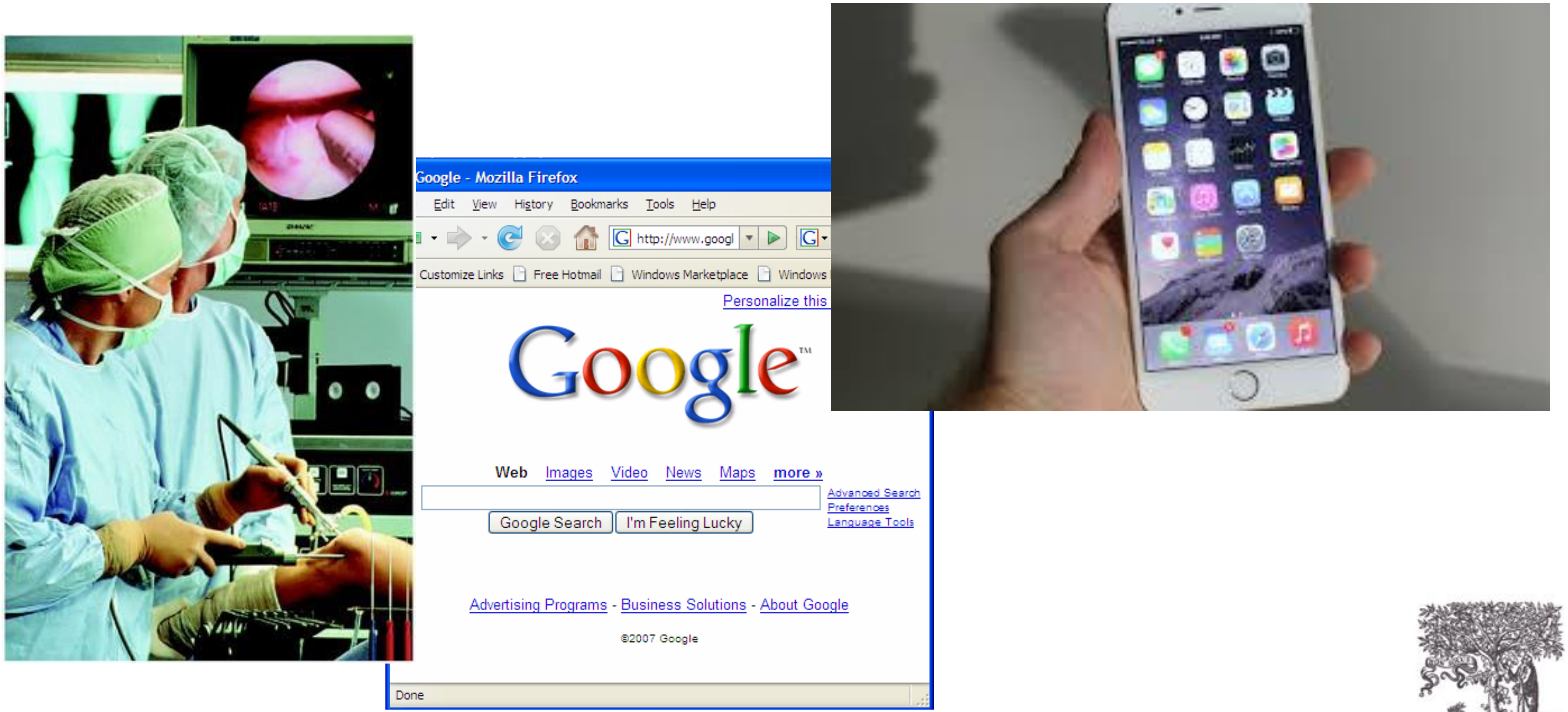
From Zero to One

Background: Digital Logic Design

- How have digital devices changed the world?
- How have digital devices changed *your* life?

Background

- Digital Devices have revolutionized our world
  - Internet, cell phones, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to over $300 billion in 2015



Google - Mozilla Firefox

Edit View History Bookmarks Tools Help

http://www.googl

Customize Links Free Hotmail Windows Marketplace Windows

Personalize this

Google™

Web Images Video News Maps more »

Advanced Search
Preferences
Language Tools

Google Search I'm Feeling Lucky

Advertising Programs - Business Solutions - About Google

©2007 Google

Done

The Game Plan

- Purpose of course:
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs

Chapter 1: Topics

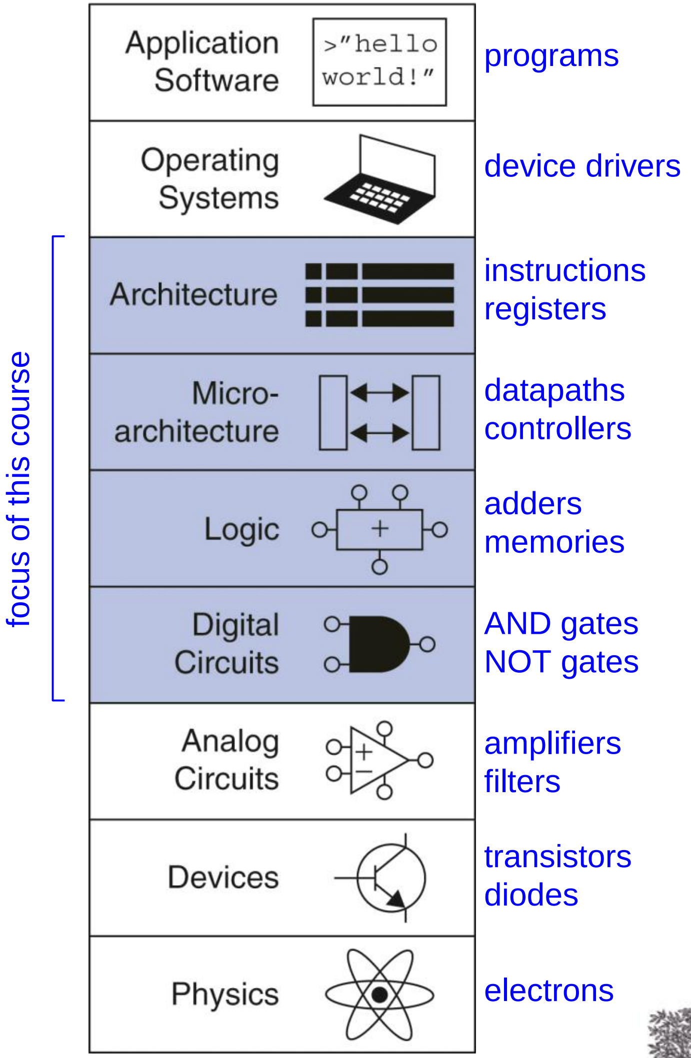
- The Art of Managing Complexity
- The Digital Abstraction
- Number Systems
- Addition
- Binary Codes
- Signed Numbers
- Logic Gates
- Logic Levels
- CMOS Transistors
- Power Consumption

The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –y's
  - Hierarch**y**
  - Modularit**y**
  - Regularit**y**

Abstraction

- What is abstraction?
  - Hiding details when they are not important
- Electronic computer abstraction
  - Different levels with different building blocks



Discipline

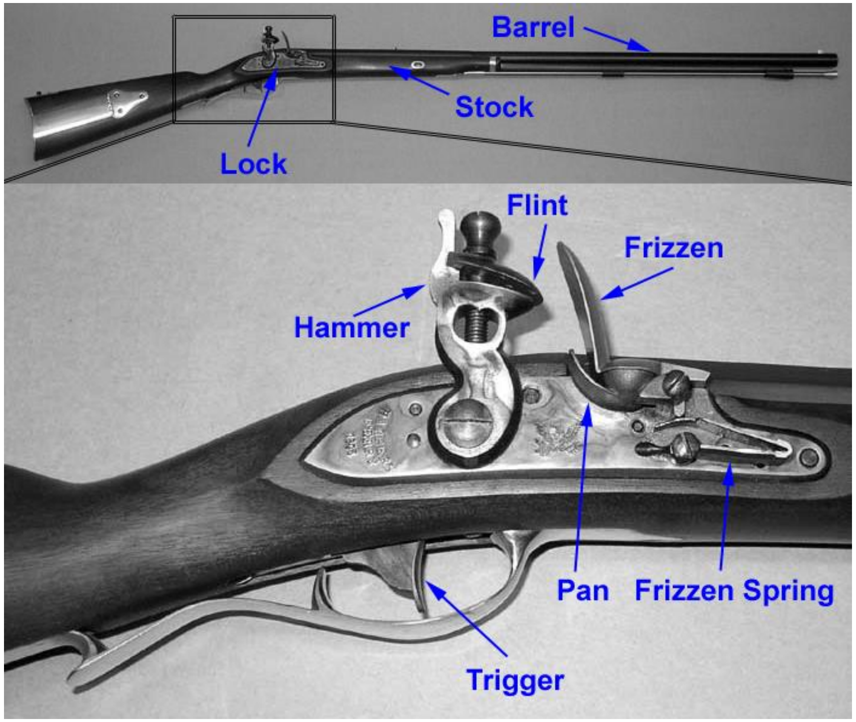
- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages (0 V, 5 V) instead of continuous (0V – 5V)
  - Simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors:
    - i.e., digital cameras, digital television, cell phones, CDs

The Three –y's

- Hierarchy
  - A system divided into modules and submodules
- Modularity
  - Having well-defined functions and interfaces
- Regularity
  - Encouraging uniformity, so modules can be easily reused

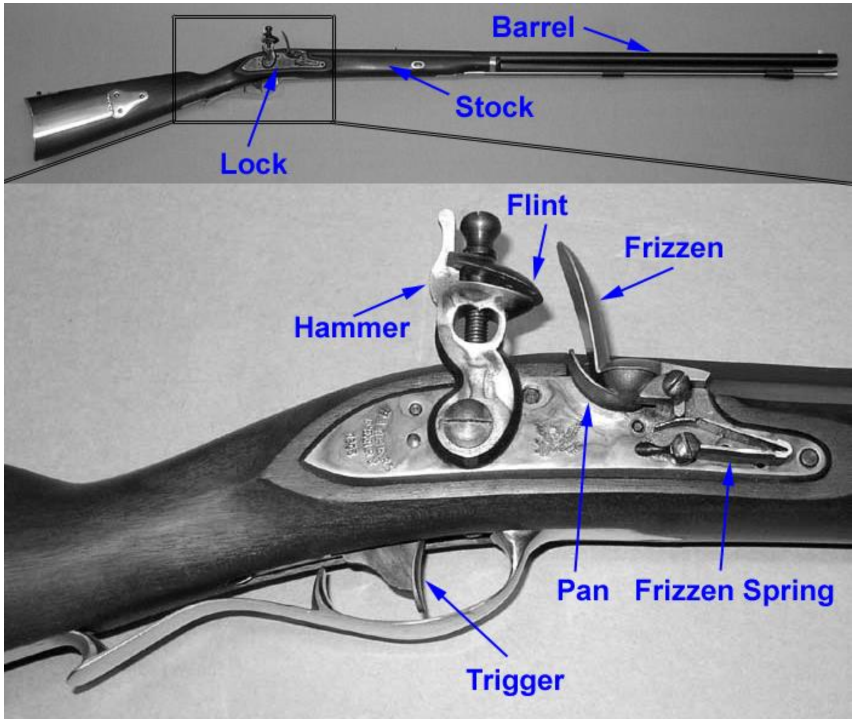
Example: Flintlock Rifle

- Hierarchy
  - Three main modules: Lock, stock, and barrel
  - Submodules of lock: Hammer, flint, frizzen, etc.



Example Flintlock Rifle

- Modularity
  - Function of stock: mount barrel and lock
  - Interface of stock: length and location of mounting pins
- Regularity
  - Interchangeable parts



The Art of Managing Complexity

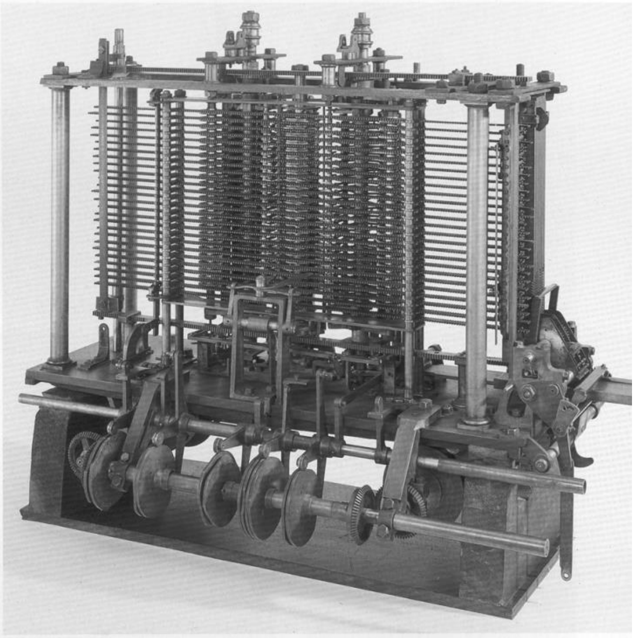
- Abstraction
- Discipline
- The Three –y's
  - Hierarch**y**
  - Modularit**y**
  - Regularit**y**

The Digital Abstraction

- Most physical variables are **continuous**
  - Voltage on a wire (1.33 V, 9 V, 12.2 V)
  - Frequency of an oscillation (60 Hz, 33.3 Hz, 44.1 kHz)
  - Position of mass (0.25 m, 3.2 m)
- Digital abstraction considers **discrete subset** of values
  - 0 V, 5 V
  - “0”, “1”

The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)
- Babbage died before it was finished



Digital Discipline: Binary Values

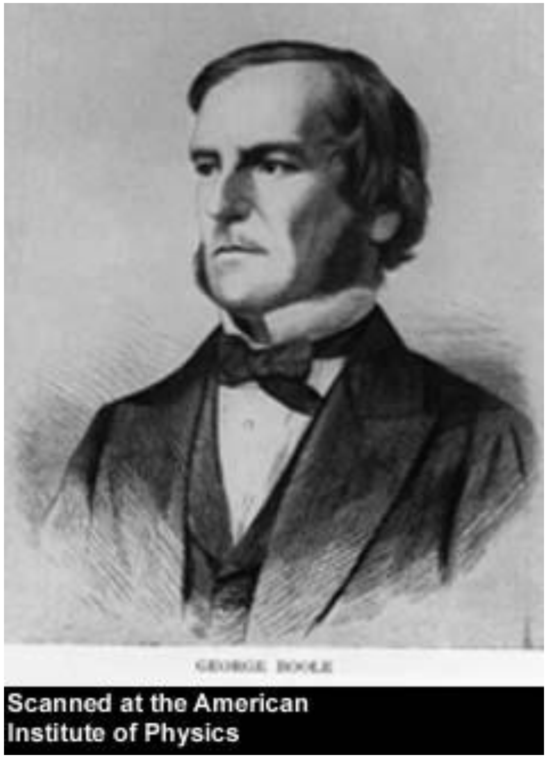
- Two discrete values
  - 1 and 0
    - 1 = TRUE = HIGH = ON
    - 0 = FALSE = LOW = OFF
- How to represent 1 and 0
  - Voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use voltage levels to represent 1 and 0
  - Bit = binary digit
    - Represents the status of a digital signal (2 values)

Why Digital Systems?

- Easier to design
- Fast
- Can overcome noise
- Error detection/correction

George Boole, 1815-1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen’s College in Ireland
- Wrote An Investigation of the Laws of Thought (1854)
- Introduced binary variables
- Introduced the three fundamental logic operations: AND, OR, and NOT



Scanned at the American Institute of Physics

Number Systems

- Decimal
  - Base 10
- Binary
  - Base 2
- Hexadecimal
  - Base 16

Decimal Numbers

- Base 10 (our everyday number system)

1000's Column

100's Column

10's Column

1's Column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

Five Thousand

Three Hundred

Seven Tens

Four Ones

Base 10

Binary Numbers

- Base 2 (computer number system)

8's Column
4's Column
2's Column
1's Column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

One Eight

One Four

Zero Two

One One

Base 2

Powers of Two

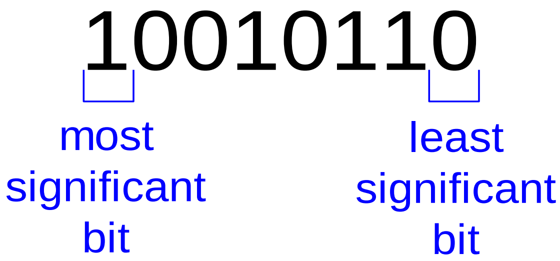
- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

Powers of Two

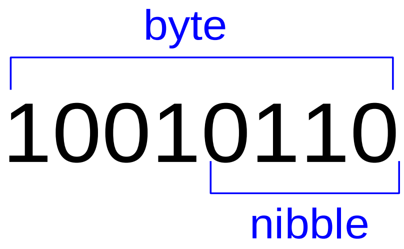
- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- Handy to memorize up to 2^{10}

Bits, Bytes, Nibbles …

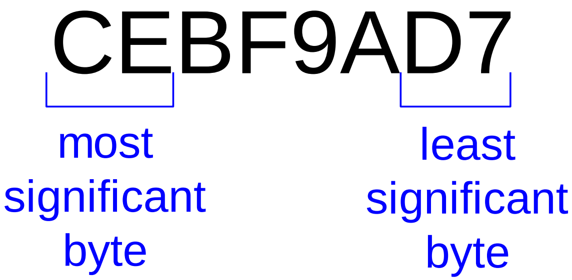
- Bits



- Bytes = 8 bits
- Nibble = 4 bits



- Words = 32 bits
  - Hex digit to represent nibble



Decimal to Binary Conversion

- Two Methods:
- Method 1: Find largest power of 2 that fits, subtract and repeat
- Method 2: Repeatedly divide by 2, remainder goes in next most significant bit

D2B: Method 1

- Find largest power of 2 that fits, subtract, repeat

53_{10}

D2B: Method 1

- Find largest power of 2 that fits, subtract, repeat

53_{10}	32×1
$53\text{-}32 = 21$	16×1
$21\text{-}16 = 5$	4×1
$5\text{-}4 = 1$	1×1

$$= 110101_2$$

D2B: Method 2

- Repeatedly divide by 2, remainder goes in next most significant bit

$53_{10} =$

D2B: Method 2

- Repeatedly divide by 2, remainder goes in next most significant bit

$53_{10} =$

$53/2 = 26 \text{ R}1$ LSB

$26/2 = 13 \text{ R}0$

$13/2 = 6 \quad \text{R}1$

$6/2 \quad = 3 \quad \text{R}0$

$3/2 \quad = 1 \quad \text{R}1$

$1/2 \quad = 0 \quad \text{R}1$ MSB

$= 110101_2$

Number Conversion

- Binary to decimal conversion
  - Convert 10011_2 to decimal

$$16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}$$

- Decimal to binary conversion
  - Convert 47_{10} to binary

$$32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2$$

D2B Example

- Convert 75_{10} to binary

D2B Example

- Convert 75_{10} to binary

$$75_{10} = 64 + 8 + 2 + 1 = 1001011_2$$

- Or

$$\begin{array}{lll}
75/2 & = 37 & \mathrm{R}1 \\
37/2 & = 18 & \mathrm{R}1 \\
18/2 & = 9 & \mathrm{R}0 \\
9/2 & = 4 & \mathrm{R}1 \\
4/2 & = 2 & \mathrm{R}0 \\
2/2 & = 1 & \mathrm{R}0 \\
1/2 & = 0 & \mathrm{R}1
\end{array}$$

Binary Values and Range

- N-digit decimal number
  - How many values?
  - Range?
- Example:
  3-digit decimal number
  - Possible values
  - Range

Binary Values and Range

- N-digit decimal number
  - How many values?
    - 10^N
  - Range?
    - $[0, 10^N - 1]$

- Example:
  3-digit decimal number
  - Possible values
    - $10^3 = 1000$
  - Range
    - $[0, 999]$

Binary Values and Range

- N-bit binary number
  - How many values?
  - Range?
- Example:
  3-bit binary number
  - Possible values
  - Range

Binary Values and Range

- N-bit binary number
  - How many values?
    - 2^N
  - Range?
    - $[0, 2^N - 1]$

- Example:
  3-bit binary number
  - Possible values
    - $2^3 = 8$
  - Range
    - $[0, 7] = [000_2, 111_2]$

Binary Values and Range

- N-digit decimal number
  - How many values?
    - 10^N
  - Range?
    - $[0, 10^N - 1]$
- Example:
  3-digit decimal number
  - Possible values
    - $10^3 = 1000$
  - Range
    - $[0, 999]$

- N-bit binary number
  - How many values?
    - 2^N
  - Range?
    - $[0, 2^N - 1]$
- Example:
  3-bit binary number
  - Possible values
    - $2^3 = 8$
  - Range
    - $[0, 7] = [000_2, 111_2]$

Hexadecimal Numbers

- Base 16 number system
- Shorthand for binary
  - Four binary digits (4-bit binary number) is a single hex digit

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary

- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal

Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
    - 0x4AF $= 0100\ 1010\ 1111_2$
- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal
    - $4 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 119910_{10}$

Number Systems

- Popular

Decimal	Base 10
Binary	Base 2
Hexadecimal	Base 16

- Others

Octal	Base 8
Any other base	

Octal Numbers

- Same as hex with one less binary digit

Octal Digit	Decimal Equivalent	Binary Equivalent
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

Number Systems

- In general, an N-digit number $\{a_{N-1}a_{N-2}\ldots a_1a_0\}$ of base R in decimal equals
  - $a_{N-1}R^{N-1} + a_{N-2}R^{N-2} + \cdots + a_1R^1 + a_0R^0$

- Example: 4-digit $\{5173\}$ of base 8 (octal)

Number Systems

- In general, an N-digit number $\{a_{N-1}a_{N-2} \ldots a_1 a_0\}$ of base R in decimal equals
  - $a_{N-1}R^{N-1} + a_{N-2}R^{N-2} + \cdots + a_1 R^1 + a_0 R^0$
- Example: 4-digit $\{5173\}$ of base 8 (octal)
  - $5 \times 8^3 + 1 \times 8^2 + 7 \times 8^1 + 3 \times 8^0 = 2683_{10}$

Decimal to Octal Conversion

- Remember two methods for D2B conversion
  - 1: remove largest multiple; 2: repeated divide
- Convert 29_{10} to octal

Decimal to Octal Conversion

- Remember two methods for D2B conversion
    - 1: remove largest multiple; 2: repeated divide
- Convert 29_{10} to octal
- Method 2

29/8	=3	R5	lsb
3/8	=0	R3	msb

$$29_{10} = 35_8$$

Decimal to Octal Conversion

- Remember two methods for D2B conversion
  - 1: remove largest multiple; 2: repeated divide
- Convert 29_{10} to octal
- Method 1

29 8×3=24

29-24=5

$$29_{10} = 24 + 5 = 3 \times 8^1 + 5 \times 8^0 = 35_8$$

  - Or (better scalability)

$$29_{10} = 16 + 8 + 4 + 1 = 11101_2 = 35_8$$

Octal to Decimal Conversion

- Convert 163_8 to decimal

Octal to Decimal Conversion

- Convert 163_8 to decimal

  - $163_8 = 1 \times 8^2 + 6 \times 8^1 + 3$
  - $163_8 = 64 + 48 + 3$
  - $163_8 = 115_{10}$

Recap: Binary and Hex Numbers

- Example 1: Convert 83_{10} to hex

- Example 2: Convert 01101011_2 to hex and decimal

- Example 3: Convert 0xCA3 to binary and decimal

Recap: Binary and Hex Numbers

- Example 1: Convert 83_{10} to hex
  - $83_{10} = 64 + 16 + 2 + 1 = 1010011_2$
  - $1010011_2 = 101\ 0011_2 = 53_{16}$
- Example 2: Convert 01101011_2 to hex and decimal
  - $01101011_2 = 0110\ 1011_2 = 6B_{16}$
  - 0x6B $= 6 \times 16^1 + 11 \times 16^0 = 96 + 11 = 107$
- Example 3: Convert 0xCA3 to binary and decimal
  - 0xCA3 $= 1100\ 1010\ 0011_2$
  - 0xCA3 $= 12 \times 16^2 + 10 \times 16^1 + 3 \times 16^0 = 3235_{10}$

Large Powers of Two

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)
- 2^{40} = 1 tera ≈ 1 trillion (1,099,511,627,776)

Large Powers of Two: Abbreviations

- 2^{10} = 1 kilo ≈ 1000 (1024)

  for example: 1 kB = 1024 Bytes

  1 kb = 1024 bits

- 2^{20} = 1 mega ≈ 1 million (1,048,576)

  for example: 1 MiB, 1 Mib (1 megabit)

- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)

  for example: 1 GiB, 1 Gib

Estimating Powers of Two

- What is the value of 2^{24}?

- How many values can a 32-bit variable represent?

Estimating Powers of Two

- What is the value of 2^{24}?
  - **$2^4 \times 2^{20} \approx 16$ million**
- How many values can a 32-bit variable represent?
  - **$2^2 \times 2^{30} \approx 4$ billion**

Binary Codes

Another way of representing decimal numbers

Example binary codes:

- Weighted codes
  - Binary Coded Decimal (BCD) (8-4-2-1 code)
  - 6-3-1-1 code
  - 8-4-2-1 code (simple binary)
- Gray codes
- Excess-3 code
- 2-out-of-5 code

Binary Codes

Decimal #	8-4-2-1 (BCD)	6-3-1-1	Excess-3	2-out-of-5	Gray
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

Each code combination represents **a single decimal digit.**

Weighted Codes

- Weighted codes: each bit position has a given weight
  - Binary Coded Decimal (BCD) (8-4-2-1 code)
    - **Example:** 726_{10} = $0111\ 0010\ 0110_{BCD}$
  - 6-3-1-1 code
    - **Example:** 1001 (6-3-1-1 code) = 1×6 + 0×3 + 0×1 + 1×1
    - **Example:** 726_{10} = $1001\ 0011\ 1000_{6311}$
- BCD numbers are used to represent fractional numbers exactly (vs. floating point numbers – which can't - see Chapter 5)

Weighted Codes

Decimal #	8-4-2-1 (BCD)	6-3-1-1
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0100
4	0100	0101
5	0101	0111
6	0110	1000
7	0111	1001
8	1000	1011
9	1001	1100

- **BCD Example:**

726_{10} = 0111 0010 0110_{BCD}

- **6-3-1-1 code Example:**

726_{10} = 1001 0011 1000_{6311}

Excess-3 Code

Decimal #	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

- Add 3 to number, then represent in binary
  - **Example:** $5_{10} = 5+3 = 8 = \mathbf{1000_2}$
- Also called a **biased** number
- Excess-3 codes (also called XS-3) were used in the 1970’s to ease arithmetic

- **Excess-3 Example:**

$\mathbf{726_{10} = 1010\ 0101\ 1001_{xs3}}$

2-out-of-5 Code

Decimal #	2-out-of-5
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

- 2 out of the 5 bits are 1
- Used for error detection:
  - If more or less than 2 of 5 bits are 1, error

Gray Codes

Decimal #	Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	1110
6	1010
7	1011
8	1001
9	1000

- Next number differs in only one bit position
  - **Example:** 000, 001, 011, 010, 110, 111, 101, 100
- **Example use:** Analog-to-Digital (A/D) converters. Changing 2 bits at a time (i.e., 011 →100) could cause large inaccuracies.

Addition

- Decimal

$$\begin{array}{r} 3734 \\ +\ \ 5168 \\ \hline \end{array}$$

- Binary

$$\begin{array}{r} 1011 \\ +\ \ 0011 \\ \hline \end{array}$$

Addition

- Decimal

$$\begin{array}{rl} 11 & \leftarrow \text{carries} \\ 3734 & \\ +\ \ 5168 & \\ \hline 8902 & \end{array}$$

- Binary

$$\begin{array}{r} 1011 \\ +\ \ 0011 \\ \hline \end{array}$$

Addition

- Decimal

$$\begin{array}{r} 11 \\ 3734 \\ +\;\; 5168 \\ \hline 8902 \end{array}$$

← carries

- Binary

$$\begin{array}{r} 11 \\ 1011 \\ +\;\; 0011 \\ \hline 1110 \end{array}$$

← carries

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1001 \\ +\ \ 0101 \\ \hline \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1011 \\ +\ \ 0110 \\ \hline \end{array}$$

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} \color{blue}{1} \\ 1001 \\ +\quad 0101 \\ \hline \color{blue}{1110} \end{array}$$

- Add the following 4-bit binary numbers

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

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1 \\ 1001 \\ +\ \ 0101 \\ \hline 1110 \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 111 \\ 1011 \\ +\ \ 0110 \\ \hline 10001 \end{array}$$

Overflow!

Overflow

- Digital systems operate on a **fixed number of bits**
- Overflow: when result is too big to fit in the available number of bits
- See previous example of 11 + 6

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two’s Complement Numbers

Sign/Magnitude

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - **Positive number:** sign bit = 0
  - **Negative number:** sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Example, 4-bit sign/magnitude representations of ± 6:
  - +6 =
  - -6 =
- Range of an *N*-bit sign/magnitude number:
  -

Sign/Magnitude

- 1 sign bit, *N*-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - **Positive number:** sign bit = 0
  - **Negative number:** sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Example, 4-bit sign/magnitude representations of ± 6:
  - +6 = **0110**
  - -6 = **1110**
- Range of an *N*-bit sign/magnitude number:
  - **$[-(2^{N-1}-1), 2^{N-1}-1]$**

Sign/Magnitude Numbers

- Problems:
  - Addition doesn't work, for example -6 + 6:

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline \end{array}$$

  - Two representations of 0 (± 0):
    - (+0) =
    - (−0) =

Sign/Magnitude Numbers

- Problems:
  - Addition doesn't work, for example -6 + 6:

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline 10100 \end{array}$$

(wrong!)

  - Two representations of 0 (± 0):
    - $(+0) = 0000$
    - $(-0) = 1000$

Two’s Complement Numbers

- Don’t have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

- Range of representable numbers not symmetric
  - One extra negative number

Two’s Complement Numbers

- msb has value of -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two’s comp number?

- Most positive 4-bit number?
- Most negative 4-bit number?

Two's Complement Numbers

- msb has value of -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's comp number?
  - $[-(2^{N-1}), 2^{\mathrm{N}-1} - 1]$
- Most positive 4-bit number? 0111
- Most negative 4-bit number? 1000

"Taking the Two's Complement"

- **Flips the sign** of a two's complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$

“Taking the Two’s Complement”

- **Flips the sign** of a two’s complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$
  1. **1100**
  2. **+ 1**

  $1101 = -3_{10}$

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$

- What is the decimal value of the two's complement number 1001_2?

Two’s Complement Examples

- Take the two’s complement of $6_{10} = 0110_2$
  1. 1001
  2. $+ \quad 1$

  $1010_2 = -6_{10}$

- What is the decimal value of the two’s complement number 1001_2?
  1. 0110
  2. $+ \quad 1$

  $0111_2 = 7_{10}$, so $1001_2 = -7_{10}$

- Add 6 + (-6) using two’s complement numbers

$$\begin{array}{rr} & 0110 \\ + & 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two’s complement numbers

$$\begin{array}{rr} & 1110 \\ + & 0011 \\ \hline \end{array}$$

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

```
 111
   0110
+  1010
-------
  10000
```

- Add -2 + 3 using two's complement numbers

```
   1110
+  0011
-------
```

Two’s Complement Addition

- Add 6 + (-6) using two’s complement numbers

$$\begin{array}{r} 111 \\ 0110 \\ +\ \ 1010 \\ \hline 10000 \end{array}$$

- Add -2 + 3 using two’s complement numbers

$$\begin{array}{r} 111 \\ 1110 \\ +\ \ 0011 \\ \hline 10001 \end{array}$$

Increasing Bit Width

- Extend number from N to M bits (M > N) :
    - Sign-extension
    - Zero-extension

Sign-Extension

- Sign bit copied to msb's
- Number value is same

- Example 1
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value:
- Example 2
  - 4-bit representation of -7 = 1001
  - 8-bit sign-extended value:

Sign-Extension

- Sign bit copied to msb's
- Number value is same

- Example 1
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- Example 2
  - 4-bit representation of -7 = 1001
  - 8-bit sign-extended value: 11111001

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers

- Example 1
  - 4-bit value = 0011_2
  - 8-bit zero-extended value:
- Example 2
  - 4-bit value = 1001
  - 8-bit zero-extended value:

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers

- Example 1
  - 4-bit value = 0011_2
  - 8-bit zero-extended value: 00000011
- Example 2
  - 4-bit value = 1001
  - 8-bit zero-extended value: 00001001

Zero-Extension

- Zeros copied to msb's
- Value changes for negative numbers

- Example 1
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: $00000011 = 3_{10}$
- Example 2
  - 4-bit value = $1001 = -7_{10}$
  - 8-bit zero-extended value: $00001001 = 9_{10}$

Number System Comparison

Number System	**Range**
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:

	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unsigned									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Two's Complement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Sign/Magnitude		1111	1110	1101	1100	1011	1010	1001	0000 1000	0001	0010	0011	0100	0101	0110	0111								