Chapter 1 :: From Zero to One

•***Digital Design and Computer Architecture*, 2nd Edition**

David Money Harris and Sarah L. Harris 저

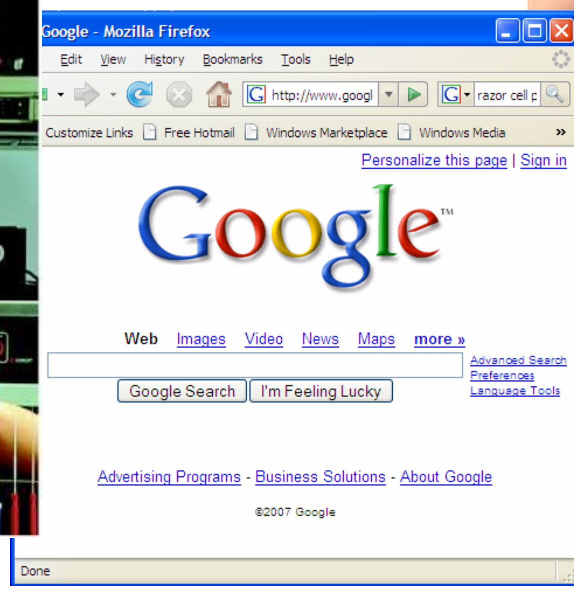
논리설계 및 실험, 한림대학교 양은샘.

Chapter 1 :: Topics

- Background
- The Game Plan
- The Art of Managing Complexity
- The Digital Abstraction
- Number Systems
- Logic Gates
- Logic Levels
- CMOS Transistors
- Power Consumption

Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to $213 billion in 2004



The Game Plan

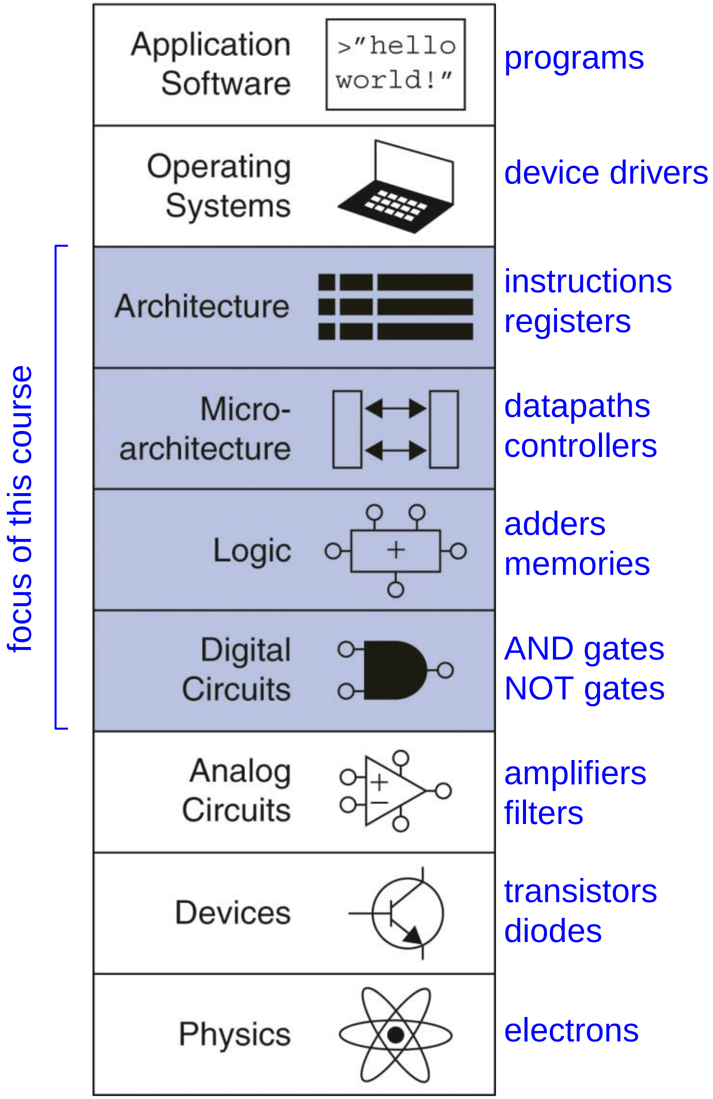
- The purpose of this course is that you:
  - Understand what's under the hood of a computer
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs
  - Design and build a microprocessor

The Art of Managing Complexity

- Abstraction (개념화)
- Discipline (규약)
- The Three –Y's
  - Hierarchy (계층화)
  - Modularity (모듈화)
  - Regularity (균일화)

Abstraction

- Hiding details when they aren't important



Discipline

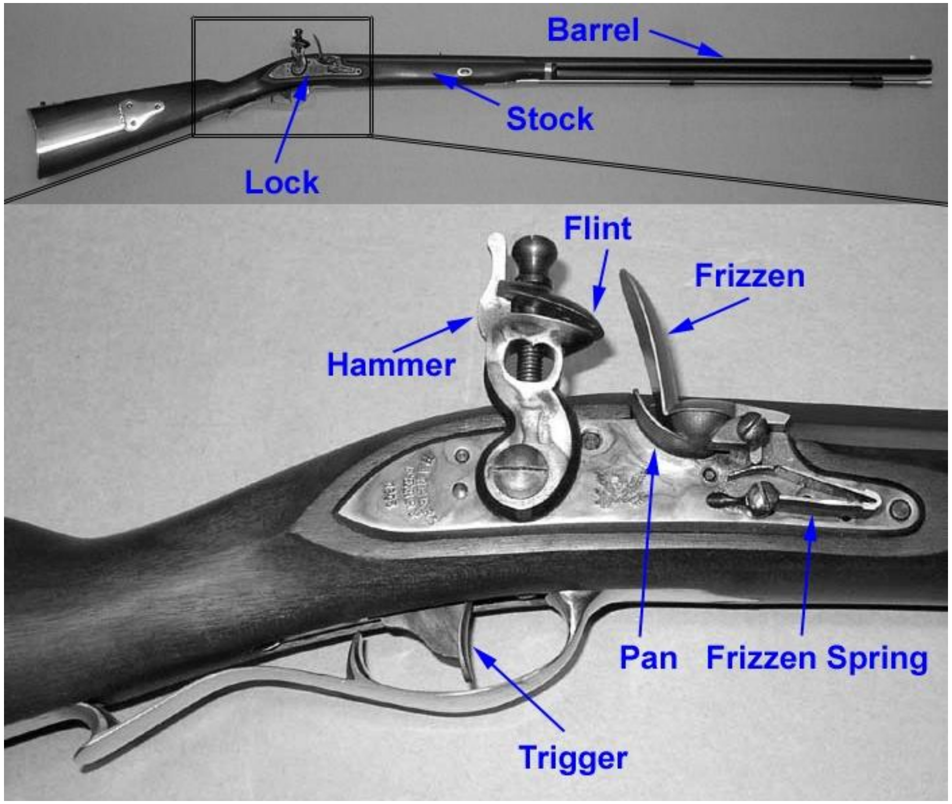
- Intentionally restricting your design choices
  - to work more productively at a higher level of abstraction
- Example: Digital discipline
  - Considering discrete voltages instead of continuous voltages used by analog circuits
  - Digital circuits are simpler to design than analog circuits – can build 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

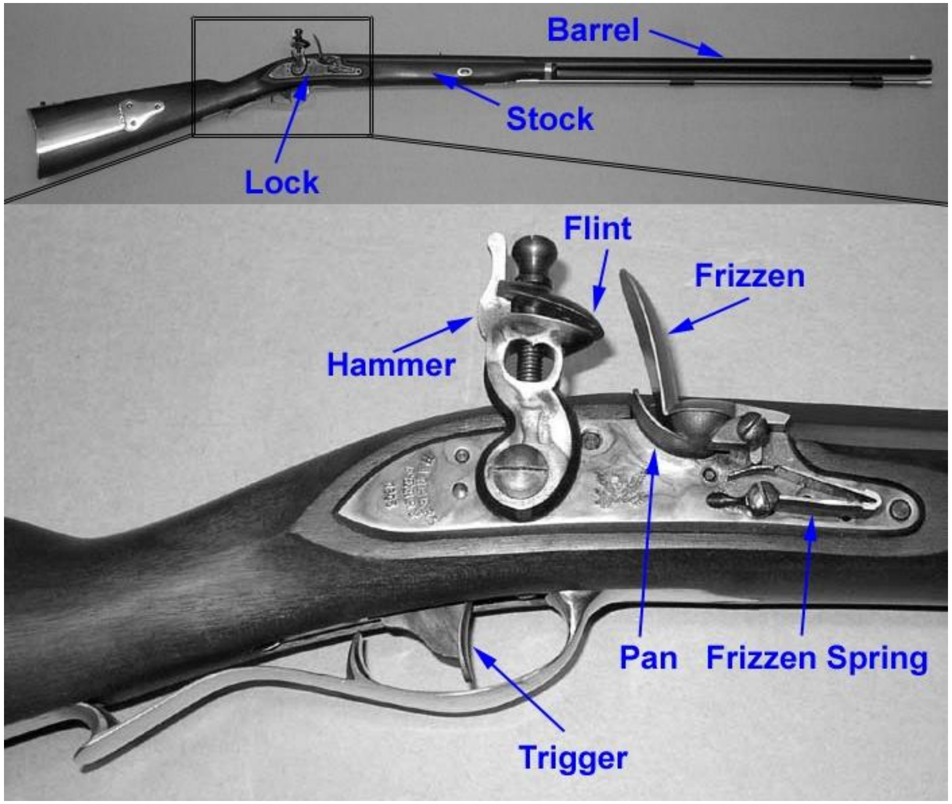
Example: Flintlock Rifle

- 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



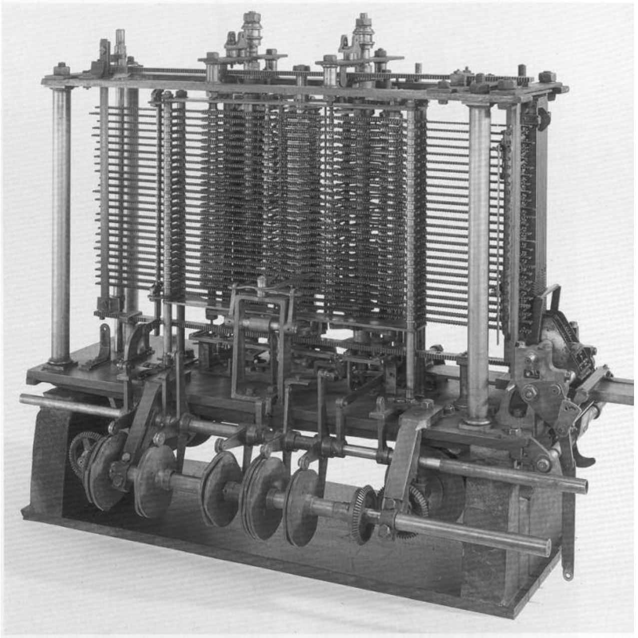
The Digital Abstraction

- Most physical variables are **continuous**, for example
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass

- Digital abstraction considers **discrete subset** of values

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

- Typically consider only **two discrete values**:
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW

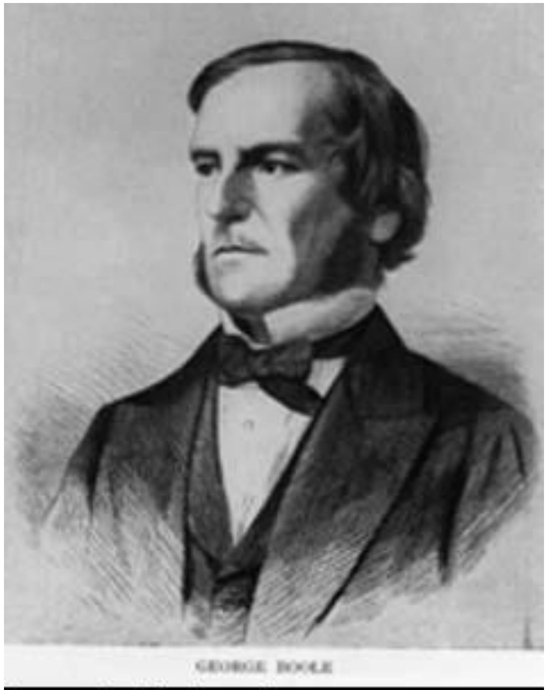
- **1 and 0** can be represented by specific voltage levels, rotating gears, fluid levels, etc.

- Digital circuits use **voltage** levels to represent 1 and 0

- ***Bit***: *B*inary dig*it*

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

1.4 Number Systems(수의 체계)

- Decimal numbers (10진수)

1의 자리
10의 자리
100의 자리
1000의 자리

$$9742_{10} = 9 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$$

9	7	4	2
천	백	십	일

- Binary numbers (2진수)

1의 자리
2의 자리
4의 자리
8의 자리
16의 자리

$$10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$$

1	0	1	1	0
16	8	4	2	1

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 2^9

Number Conversion

- Decimal to binary conversion:
  - Convert 10011_2 to decimal (2진수를 10진수로)
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$

- Decimal to binary conversion:
  - Convert 47_{10} to binary (10진수를 2진수로)
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

Binary Values and Range

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

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

Binary Values and Range

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

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

Hexadecimal Numbers (16진수)

- Base 16
- Shorthand for Binary

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
	16	

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
  - **$0100\ \ 1010\ \ 1111_2$**

- **Hexadecimal to decimal conversion:**
  - Convert 0x4AF to decimal
  - $010010101111_2 = 1\times2^{10} + 1\times2^{7} + 1\times2^{5} + 1\times2^{3} + 1\times2^{2} + 1\times2^{1} + 1\times2^{0}$
    $= 1024 + 128 + 32 + 8 + 4 + 2 + 1$
    $= \mathbf{1199_{10}}$
  - $0x4AF = (4 \times 16^2) + (10 \times 16^1) + (15 \times 16^0)$
    $= \mathbf{1199_{10}}$

Bits, Bytes, Nibbles...

- Bits

10010110

most significant bit

least significant bit

- Bytes & Nibbles

byte

10010110

nibble

- Bytes

CEBF9AD7

most significant byte

least significant byte

최상위바이트

최하위바이트

Large Powers of Two (거듭제곱의 계산)

- $2^{10} = 1$ kilo ≈ 1000 (1024)
- $2^{20} = 1$ mega ≈ 1 million (1,048,576)
- $2^{30} = 1$ giga ≈ 1 billion (1,073,741,824)

Estimating Powers of Two

- What is the value of 2^{24}?
  - $2^4 \times 2^{20} \approx$ **16 million** = 16 Mega

- How many values can a 32-bit variable represent?
  - $2^2 \times 2^{30} \approx$ **4 billion** = 4 Giga

Signed Binary Numbers (부호화된 2진수)

- 부호화된 2진수의 표현법 2가지
  - Sign/Magnitude Numbers (부호 및 크기의 2진수 표현법)
  - Two’s Complement Numbers (2의 보수)

Sign/Magnitude Numbers (부호/크기 표기법)

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

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

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

- Example, 4-bit representations of ± 5:

  $-5 = 1101_2$

  $+5 = 0101_2$

- Range of an *N*-bit sign/magnitude number:

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Sign/Magnitude Numbers (부호/크기 표기법)

- Problems (문제점):
  - Addition doesn't work, for example -5 + 5:

$$
\begin{array}{r}
1101 \\
+\ 0101 \\
\hline
10010
\end{array}
$$

(wrong!)

  - Two representations of 0 (± 0): (두 가지의 zero표현)

1000 (-0)

0000 (+0)

Two’s Complement Numbers

- Don’t have same problems as sign/magnitude numbers: (부호 및 크기의 2진수 표현법에서 발생한 문제점 해결)
  - Addition works
  - Single representation for 0 (zero 표현 한 가지로 가능)

"Taking the Two's Complement"

- Reversing the sign of a two's complement number
- Method (2의 보수로 표현된 음수 만들기) :
  1. Invert the bits (모든 비트를 반전 시킨 값에)
  2. Add 1 (1을 더한다)
- Example: Reverse the sign of (+7) = 0111_2
  1. 1000
  2. \+ 1

     1001 = (-8+1= -7)

Two's Complement Examples

- Take the two's complement of (+6) = 0110_2
  1. 1001
  2. + 1

  1010 = (-8+2 = -6)

- Take the two's complement of (-3) = 1101_2
  1. 0010
  2. + 1

  0011 = (+3)

Addition (2진수의 덧셈)

- Decimal

$$\begin{array}{rr} & \color{blue}{11} \\ & 3734 \\ + & 5168 \\ \hline & 8902 \end{array}$$

← carries

- Binary

$$\begin{array}{rr} & \color{blue}{11} \\ & 1011 \\ + & 0011 \\ \hline & 1110 \end{array}$$

← 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

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

Two's Complement Addition

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

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

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

```
  111
   1110
+  0011
-------
  10001
```

Overflow (오버플로우)

- Digital systems operate on a **fixed number of bits**
- overflows : when the result is too big to fit in the available number of bits (계산과정에서 결과 값이 데이터의 용량에 비해 크거나 작을 때)
- Example: add 13 and 5 using 4-bit numbers

$$\begin{array}{r} 11\ 1 \\ 1101 \\ +\quad 0101 \\ \hline 10010 \end{array}$$

Overflow (오버플로우)

- Unsigned - 최상위 비트를 넘어가는 carry가 발생할 때
- Sign/Magnitude

- Two’s Complement

  ①두 개의 숫자가 양수일 때 결과 값이 음수거나

  ②두 개의 숫자가 음수일 때 결과 값이 양수일 때

  => 최상위비트 carry_in != carry_out

Two's Complement Numbers

- Same as unsigned binary, but the most significant bit (msb) has value of -2^{N-1}
- Most positive 4-bit number: 0111_2 (7_{10})
- Most negative 4-bit number: 1000_2 ($-2^3 = -8_{10}$)

- The most significant bit still indicates the sign
  (1 = negative, 0 = positive)
  (첫 비트가 0이면 양수, 1이면 음수)
- Range of an N-bit two's complement number (2의 보수로 표현할 수 있는 범위) : $[-2^{N-1}, 2^{N-1}-1]$

Increasing Bit Width (비트확장)

- A value can be **extend**ed from N bits to M bits (where $M > N$) by using:
  - Sign-extension (부호비트로 확장)
  - Zero-extension (zero로 확장)

Sign-Extension(부호 확장)

- Sign bit is copied into **most significant bits**.
- Number value remains the same.

- **Example 1:**
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- **Example 2:**
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011

Zero-Extension(zero 확장)

- Zeros are copied into most significant bits.
- Number value may change.

- **Example 1:**
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: 00000011 = 3_{10}

- **Example 2:**
  - 4-bit value = 1011 = -5_{10}
  - 8-bit zero-extended value: 00001011 = 11_{10}

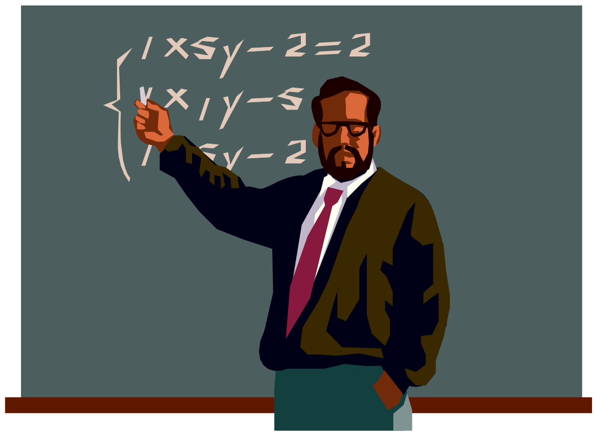
Number System Comparison

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								

Q & A



[실습] Chapter 1.1~1.4 :: 연습문제(p40~)

- **수의 표현 범위 : 7, 8, 9, 10, 11, 12, 41**

- **진법 : 13(a,b), 15(a,b), 17(a,b), 21(a,c), 23(a,c)**

- **data의 표현 : 43, 45, 46, 48, 49**

- **2의 보수의 수 : 68**

- **오버플로우 : 52(b), 54(b), 56(c), 61(c)**

- **Extention(부호 확장) : 33, 35,**