w09-Lec1

Data Representation Part I Integer

Assembled for 204111
by Kittipitch Kuptavanich
Ratsameetip Wita

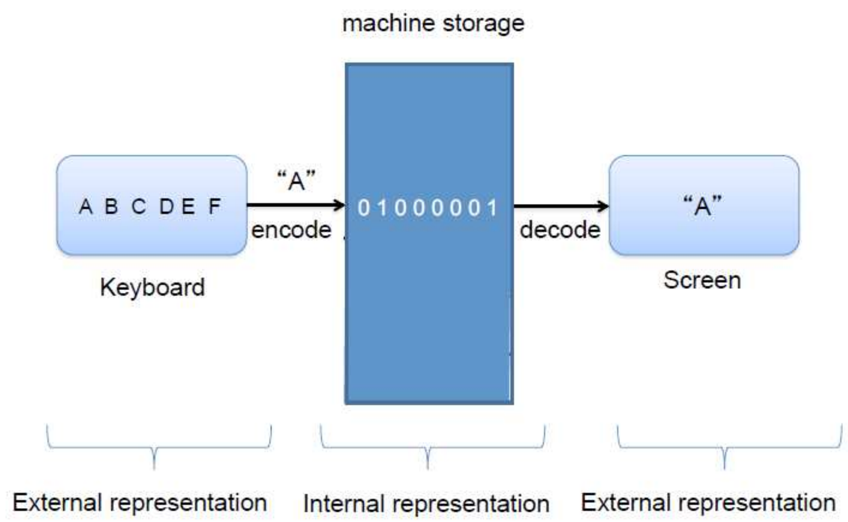
Integer and Real Number

- **Digital Data**
- **Integer Representation**
- **Binary Arithmetic**
- **Negative Integer Encoding**
- **Real Number and Floating Point Representation**
- **Rounding**

Representing Data

- **เราสามารถเก็บข้อมูลต่าง ๆ ในชีวิตประจำวันไว้ในคอมพิวเตอร์ได้ เช่น**
  - **เอกสาร รูปภาพ เสียง ตัวเลข ฯลฯ**
- **คอมพิวเตอร์จะมีวิธีการเก็บในรูปแบบของเลขฐาน 2**
  - **เข้ารหัสข้อมูล (Encode) แปลงข้อมูล ->เลขฐาน 2**
  - **ถอดรหัสข้อมูล (Decode) เลขฐาน 2 -> ข้อมูล**

Representing Data



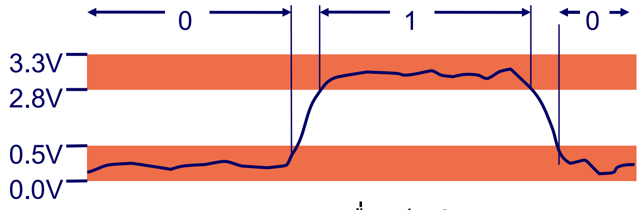
from Carnegie Mellon University's 15213 course slide by Greg Kesden

Why Base 2?

- **รูปแบบของเลขฐาน 2 ประกอบด้วยเลข 0 และ 1**
  - **Represent 15213_{10} as 11101101101101_2**
  - **Represent 1.20_{10} as $1.0011001100110011[0011]..._2$**
- **เลข 0 และ 1 สามารถตีความได้เป็นสถานะ เปิด/ปิด ของกระแสไฟฟ้า**
  - **การส่งสัญญาณในคอมพิวเตอร์เป็นการไหลของกระแสไฟฟ้า**

Digital Data

- ข้อมูลต่างๆในระบบคอมพิวเตอร์ มีลักษณะดังนี้
  - ระดับไฟฟ้า **binary** physical states (high or low voltages, etc.)
  - สามารถแปลความหมายได้ในรูปแบบบิต (bit -1s and 0s)



- เราสามารถแปลความหมายบิต เพื่อเก็บข้อมูลประเภทต่างๆ เช่น integers, real numbers, text, …

Bits, Bytes and Words

- **หน่วยของข้อมูลที่เล็กที่สุดในคอมพิวเตอร์คือ Bits มีค่า 0 หรือ 1 (Binary Digits)**
- **กลุ่มของ Bits ที่ใช้ในการแทนค่าข้อมูลใด ๆ จะเรียกว่า Bytes ซึ่ง 1 Bytes จะประกอบด้วย 8 Bits (เช่น 1 ตัวอักษร)**
- **ในการอ่านข้อมูลของคอมพิวเตอร์ จะมีการอ่านกลุ่มของ Bytes ในลักษณะเป็นก้อน ๆ หรือ fixed-sized chunks เรียกว่า Word**
- **Word จะมีขนาด 4 bytes (32 bits) หรือ 8 bytes (64 bits) ขึ้นกับรูปแบบสถาปัตยกรรม**
- **In this class we assume a word size of <u>*4*</u> bytes**

Types interpret bits

- **a 32-bit "word" might be**
- **1100 1100 1011 0111 0000 0000 0000 0000**
- **what this means depends on the machinery to interpret it, could be (explore with 0xED)**

Type	Interpret
bits	1100 1100 1011 0111 0000 0000 0000 0000
Floating point number	6.59339 X 10-41
String (Unicode UTF-16)	첷
RGB pixel color	
Little-endian integer	47052

Information Capacity

จำนวน Bits	ค่าที่เป็นไปได้	จำนวนของข้อมูลที่เป็นไปได้ทั้งหมด
1	0 1	2 (2^1)
2	00 01 10 11	4 (2^2)
3	000 001 010 011 100 101 110 111	8 (2^3)
4	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	16 (2^4)

k bits สามารถแสดงผลข้อมูลที่ต่างกันได้ทั้งหมดจำนวน 2^k ค่า

Place-value numerals (base 10)

- **The numeral we write: 15627**
- **What it means:**
  - $7 \times 10^0 + 2 \times 10^1 + 6 \times 10^2 + 5 \times 10^3 + 1 \times 10^4$
- **Problem: electronic circuitry for base-10 arithmetic is slow.**
- **Solution: use place-value numerals, but in base 2–binary notation**

Place-value numerals in general

- **กำหนด b เป็นเลขฐาน (base หรือ radix)**
- **เลขที่เป็นไปได้ในการแสดงผล จะมีจำนวน b ตัว**
  - **base 10 : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
  - **base 2 : 0, 1**
  - **base 8 : 0, 1, 2, 3, 4, 5, 6, 7**
  - **base 16 : 0, 1, …, 9, A, B, C, D, E, F**
- **การแสดงผล จำนวน n ในฐาน b**
  - **ทำการหาร n ด้วย b เพื่อจำเศษที่เหลือจากการหาร r (a digit)**
  - **หารไปเรื่อย ๆจนกว่า $n_i = 0$**
  - **เศษของการหารที่ได้ เรียงลำดับแบบ reverse order จะเป็นค่าของ n ในฐาน b**

Binary place-value example

- **Base 2 ตัวเลขแสดงผลที่ใช้ได้คือ.....................**
- **ต้องการแสดง 6 ในฐาน 2:**
  - **6 // 2 = 3 เศษ ....**
  - **3 // 2 = 1 เศษ ....**
  - **1 // 2 = 0 เศษ ....**
- **เลข 6 สามารถแสดงในรูปแบบฐาน 2 = ...**
- **......... $\times 2^0$ + ... $\times 2^1$ + ... $\times 2^2 = 6_{10}$**

Binary place-value example

- **Base 2 ตัวเลขแสดงผลที่ใช้ได้คือ 0,1**
- **ต้องการแสดง 6 ในฐาน 2:**
  - **6 // 2 = 3 เศษ 0**
  - **3 // 2 = 1 เศษ 1**
  - **1 // 2 = 0 เศษ 1**
- **เลข 6 สามารถแสดงในรูปแบบฐาน 2 = 110_2**
- **$0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 6_{10}$**

Base Number in Computer

- เลขฐานที่เกี่ยวข้องกับคอมพิวเตอร์ ประกอบด้วย
  - **Base 2 (Binary): แทนค่าในระดับบิท (0 1)**
  - **Base 8 (Octal) : มีความสัมพันธ์กับเลขฐาน 2 โดย เลขฐาน 2 จำนวน 3 บิท สามารถแทนได้ด้วยเลขฐาน 8 จำนวน 1 บิท ทำให้การเขียนตัวเลขสั้นลง**
    เช่น $(110111)_2 = (67)_8$
  - **Base 16 (Hexadecimal) : เลขฐาน 2 จำนวน 4 บิท สามารถแทนได้ด้วยเลขฐาน 16 จำนวน 1 บิท ทำให้การเขียนตัวเลขสั้นลง**
    เช่น $(11010101)_2 = (D5)_{16}$

from Carnegie Mellon University's 15213 course slide by Greg Kesden

Encoding Byte Values

- **Byte = 8 bits**
  - **Binary 00000000_2 to 11111111_2**
  - **Decimal: 0_{10} to 255_{10}**
  - **Hexadecimal 00_{16} to FF_{16}**
- **Base 16 (Hexadecimal)**
  - **ใช้สัญลักษณ์ '0' to '9' และ 'A' to 'F'**
  - **Write $FA1D37B_{16}$ as**
    - `0xFA1D37B`
    - `0xfa1d37b`

Hex	Decimal	Binary
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

Conversion Exercise

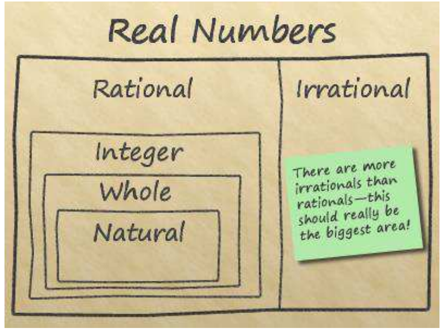
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Practice Problem 2.3: Fill in the missing entries

Decimal	Binary	Hexadecimal
0	0000 0000	0x00
167	________	________
62	________	________
188	________	________
________	0011 0111	________
________	1000 1000	________
________	1111 0011	________
________	________	0x52
________	________	0xAC
________	________	0xE7

Number System (Revisited)

- **Integer จำนวนเต็มใด ๆ สามารถเป็นได้ทั้งจำนวนบวกและลบ**
- **Rational จำนวนตรรกยะ (สามารถแทนได้ด้วยเศษส่วนของจำนวนเต็ม)**
- **Irrational จำนวนอตรรกยะ (ไม่สามารถแทนด้วยเศษส่วนของจำนวนเต็มได้)**
- **Real จำนวนจริง หมายถึงระบบตัวเลขทั้งหมด**



Integer Representation

- **เลขจำนวนเต็ม Integer ประกอบด้วย**
  - **จำนวนเต็มบวก Positive Integer**
  - **ศูนย์ Zero**
  - **จำนวนเต็มลบ Negative Integer**
- **ในการเก็บข้อมูลจำนวนเต็ม จะเป็นการเก็บในรูปแบบเลขฐาน 2 ขนาดตามกำหนดโดย Data Type**

Integer Representation

- **จากความรู้ในเรื่องเลขฐาน Bit Byte Word จะเห็นว่าการเก็บข้อมูลในคอมพิวเตอร์นั้นมีจำนวนจำกัด**
- **การแทนค่าข้อมูลตัวเลข จึงมีจำนวนจำกัด ขึ้นอยู่กับความจุ (Capacity) ของหน่วยความจำ**

Bits	Minimum	Maximum
8	0	$2^8 - 1$ (255)
16	0	$2^{16} - 1$ (65,535)
32	0	$2^{32} - 1$ (4,294,967,295)
64	0	$2^{64} - 1$ (18,446,744,073,709,551,615)

Ranges for typical computer "word" sizes

Binary Arithmetic

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

+	0	1
0	0	1
1	1	10

x	0	1
0	0	0
1	0	1

Binary Arithmetic

- **ในการบวกจะทำงานปกติ และแสดงผลด้วยเลข 0 หรือ 1 เท่านั้น**
  - 1 + 1 = 10, 10 - 1 = 1, 10 + 1 = 11
- **สมมติ ขนาดของข้อมูล = 4 bits**

```
   1010
+  1010
  _____

  _____
```


Binary Arithmetic

- **ในการบวกจะทำงานปกติ และแสดงผลด้วยเลข 0 หรือ 1 เท่านั้น**
  - 1 + 1 = 10, 10 - 1 = 1, 10 + 1 = 11
- **สมมติ ขนาดของข้อมูล = 4 bits**

```
  1 1
   1010
+  1010
  _____
  10100
```

ผลลัพธ์มีขนาด 5 bits
ทำให้เกินขนาดของข้อมูล

Overflow

- **หากกำหนดให้ขนาดของข้อมูลมีขนาด k bits ใด ๆ แล้ว ข้อมูลที่บรรจุก็จะมีได้ไม่เกิน k หลัก เท่านั้น**

```
   |1|0|1|0|
+  |1|0|1|0|
   ---------
 1 |0|1|0|0|
```

- **ข้อมูลที่เก็บจริงในหน่วยความจำจะมีค่า 0100 เท่านั้น (20 mod 16 = 4)**

Overflow Concept

- **Base 10: 2 digits**

```
   |6|4|    6x10^1+4x10^0
+  |4|8|    4x10^1+8x10^0
    ---
 1 |0|2|
```

- **102 mod 10^2 = 02**

- **Base 2: 4 digits**

```
   |1|0|1|0|    1x2^3+0x2^2+1x2^1+0x2^0
+  |1|0|1|0|
   -------
 1 |0|1|0|0|
```

- **10100 mod 2^4 = 0100**

Negative Integer

- จาก **Integer Representation** เราทราบว่าการเก็บตัวเลขมีการแปลงและเก็บในรูปแบบฐาน 2
- วิธีที่ง่ายที่สุด ใช้ Bit ที่สำคัญที่สุด (most significant bit: MSB) เป็นตัวบอกเครื่องหมาย ถ้า MSB (ซ้ายสุด) เป็น 1 แสดงว่าเป็นจำนวนลบ ถ้าเป็น 0 แสดงว่าเป็นจำนวน<u>บวก</u> วิธีนี้เรียกว่า sign and magnitude

$8 = 1000_2$,

$-8 = 11000_2$

  - ปัญหา จะทำให้เกิด 0, -0 (0000_2,10000_2)
  - เมื่อทำมาบวกลบกัน จะทำให้ค่าผิดเพี้ยน

Signed Bit

- Let's try some operation
- 5 + 2

```
    0 1 0 1        5
+   0 0 1 0        2
=   0 1 1 1        7 (correct)
```

- 5 - 2 = 5 + -2

```
    0 1 0 1        5
+   1 0 1 0       -2
=   1 1 1 1       -7 (wrong)
```


Two's Complement for Negative Integer [1]

- สิ่งที่ต้องคำนึงถึงในการแทนข้อมูลคือต้องสามารถทำ operation ทางคณิตศาสตร์ได้และให้ผลลัพธ์ถูกต้อง
- ใช้แนวคิดของการทำ Modular Arithmetic (mod)
- เมื่อมีจำนวน k bits จะทำการ mod ด้วย 2^k
- We define negative numbers as *additive inverse*
  - ให้ –x = y แล้ว จะได้ x + y = 0 mod 2^k เรียกว่าเป็น two's complement ของ X

Two's Complement for Negative Integer [2]

- ตัวอย่าง Base 2: 4 bits   1= 0001, -1 =?

```
Carry Bits            1              1 1            1 1 1
  0 0 0 1        0 0 0 1        0 0 0 1        0 0 0 1
+ ? ? ? ?      + ? ? ? 1      + ? ? 1 1      + ? 1 1 1
  0 0 0 0        0 0 0 0        0 0 0 0        0 0 0 0

           1 1 1 1
             0 0 0 1            0 0 0 1
           + 1 1 1 1          + 1 1 1 1
             0 0 0 0          1 0 0 0 0
```

Representation of -1

Unsigned and Signed Integer

- **Unsigned Integer (3 bits)**

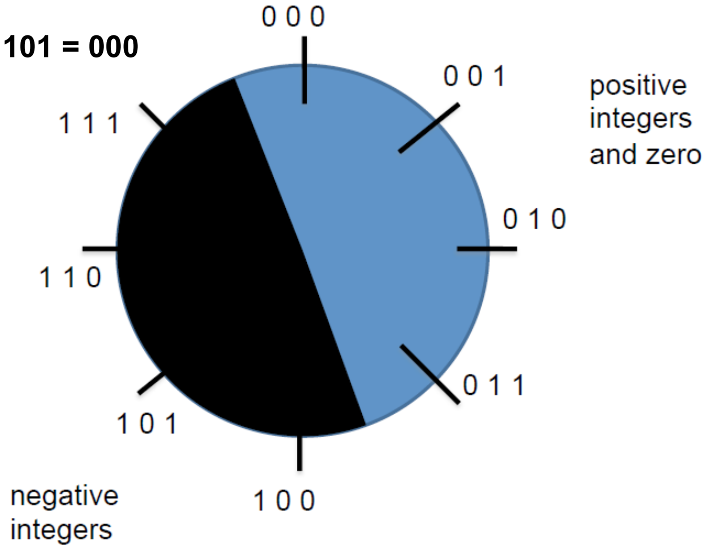
Bit Represent	Decimal Value
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

- **Signed Integer (3 bits)**

Bit Represent	Decimal Value
000	0
001	+1
010	+2
011	+3
100	- 4
101	- 3
110	- 2
111	- 1

Two's Complement Integer

- **เมื่อนำ n+(-n) = 0 เสมอ**
- **เช่น 011 + 101 = 000**



Two's-Complement Encodings [1]

- **มองหาบิตซ้ายสุด (MSB- Most Significant Bit)**
  - **1 คือ..................... 0 คือ..........................**
- **วิธีที่ 1: complement the bits and then increment กลับบิตแล้วบวก 1**
  - **( ) 0101 → 1010+1 → 1011 ( )**
- **วิธีที่ 2: Let k be the position of the rightmost 1, we complement each bit to the left of bit position k (หา 1 ตัวขวาสุด แล้วกลับ bit เฉพาะทางซ้ายของ 1 ตัวนั้น)**
  - **( ) 1100 → 1100 → 0100 ( )**

Two's-Complement Encodings [1]

- **มองหาบิตซ้ายสุด (MSB- Most Significant Bit)**
  - **1 คือ..จำนวนเต็มลบ..... 0 คือ.....จำนวนเต็มบวก......**
- **วิธีที่ 1: complement the bits and then increment กลับบิตแล้วบวก 1**
  - **(5) 0101 → 1010+1 → 1011 (-5)**
- **วิธีที่ 2: Let k be the position of the rightmost 1, we complement each bit to the left of bit position k (หา 1 ตัวขวาสุด แล้วกลับ bit เฉพาะทางซ้ายของ 1 ตัวนั้น)**
  - **(-4) 1100 → 1100 → 0100 (4)**

Two's-Complement Encodings [2]

- แนวคิดคือ ในกรณี **MSB** เป็น 1 นอกจาก จะหมายความว่า จำนวนดังกล่าวเป็นจำนวนลบแล้ว bit นี้ยังมีน้ำหนักค่าเท่ากับ -2^{n-1}
- Bit อื่น ๆ มีค่าปรกติ เช่นกรณี 4 bit (-2^3 to 2^3-1)

$$
\begin{aligned}
0001 &= -0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 &&= 0+0+0+1 &&= 1\\
0101 &= -0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 &&= 0+4+0+1 &&= 5\\
1011 &= -1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 &&= -8+0+2+1 &&= -5\\
1111 &= -1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 &&= -8+4+2+1 &&= -1
\end{aligned}
$$

Two's Complement Exercise

- ให้ทำการหาค่า **Two's Complement** ของตัวเลขต่อไปนี้

X		-X	
Bit Represent	Decimal Value	Bit Represent	Decimal Value
1010			
		1001	
			3
		0100	
	5		
1000			

Range Calculation

- **Consider an unsigned environment (0 and positive integer) For 4 bits**
  - **We can represent** 2^4 **different numbers :** 0000, 0001, 0010, …… 1111
  - **The maximum is**
    - $1111_2 = 15_{10} = 2^4 - 1$
  - **In signed representations, about half of the range will be assigned to represent the negative numbers for** w **bits the range will be** -2^{w-1} **to** $2^{w-1} - 1$

Bits	unsigned data range	value
2	0 - (2² - 1)	0 - 3
4	0 – (2⁴ – 1)	0 – 15
…	…	…
w	0 – (2ʷ – 1)	

Range Calculation Exercise

- Fill in the rest of the entries

Bits	Unsigned range	Unsigned value	Signed range	Signed value
2	0 to 2^2 – 1	0 to 3		
4	0 to 2^4 – 1	0 to 15		
8	0 to 2^8 – 1	0 to 255		
16				
32				
w	0 to 2^w – 1			

Two's Complement Exercise (Ans.)

- ให้ทำการหาค่า **Two's Complement** ของตัวเลขต่อไปนี้

X		-X	
Bit Represent	**Decimal Value**	**Bit Represent**	**Decimal Value**
1010	-6	0110	6
0111	7	1001	-7
1101	-3	0011	3
1100	-4	0100	4
0101	5	1011	-5
1000	-8	1000	-8

Range Calculation Exercise (Ans.)

- Fill in the rest of the entries

Bits	Unsigned range	unsigned value	Signed range	Signed value
2	0 to 2^2 – 1	0 to 3	-2^1 to 2^1-1	-2 to 1
4	0 to 2^4 – 1	0 to 15	-2^3 to 2^3-1	-8 to 7
8	0 to 2^8 – 1	0 to 255	-2^7 to 2^7-1	-128 to 127
16	0 to 2^{16} – 1	0 to 65535	-2^{15} to $2^{15}-1$	-32768 to 32767
32	0 to 2^{32} – 1	0 to 4294967295	-2^{31} to $2^{31}-1$	-2147483648 to - 2147483647
w	0 to 2^w – 1		-2^{w-1} to $2^{w-1}-1$	

Conversion Exercise (Ans.)

Practice Problem 2.3: Fill in the missing entries

Decimal	Binary	Hexadecimal
0	0000 0000	0x00
167	1010 0111	0xA7
62	0011 1110	0x3E
188	1011 1100	0xBC
55	0011 0111	0x37
136	1000 1000	0x88
243	1111 0011	0xF3
82	0101 0010	0x52
172	1010 1100	0xAC
231	1110 0111	0xE7