Data and Numbers – Introduction to Computer (計算機概論)

Winston H. Hsu (徐宏民)
National Taiwan University, Taipei

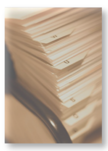
September 12, 2022

Office: R512, CSIE Building
Communication and Multimedia Lab (通訊與多媒體實驗室)
http://winstonhsu.info

Some of the slides are from the two textbooks and the reference

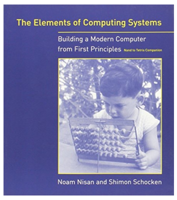
Administrative Issues

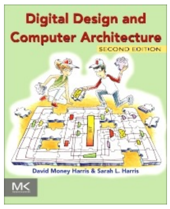
- Homework (#2) available now
  - **DUE: at 12pm, September 19, 2022 (Monday); end of the lecture time; hard copy (online for those in quarantine)**
- TAs
  - 趙雋同 <r11922109@ntu.edu.tw>, R506
  - 許雅晴 <r10922192@ntu.edu.tw>, R506
  - Office hours (mail inquiry first): 2-3pm, Tuesday & Wednesday
- Course information
  - **Course outline**
    https://winstonhsu.info/2022f-comp-intro/
  - **Readings, homework, slides, etc.**
    https://cool.ntu.edu.tw/courses/19517



Textbook and References

- Textbook:
  - The Elements of Computing Systems, Noam Nisan and Shimon Schocken
    - The first 6 chapters (to be used) are available online; you probably need not buy the hard copy
    - **http://www1.idc.ac.il/tecs/plan.html**
  - Digital Design and Computer Architecture, 2nd Edition, David Harris and Sarah Harris.
  - **Digital copy can be easily found online**
- References
  - Computer Science: An Overview, 12th Edition, by J. Glenn Brookshear
    - To be used in the second half
    - 前幾屆的單班用書





Topics to Be Covered

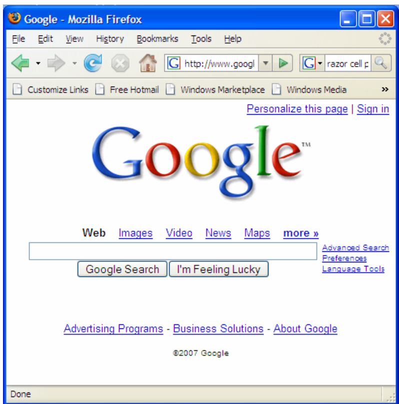
- Harris: Chap. 1.1-1.5
- Brookshear: Chap. 1.1, 1.2, 1.4-1.7
- Nisan: Chap. 2.1
- Some sections are overlapped across books.
- Today's slides are mainly from [Harris] and [Brookshear]

Outline

- Background
- The Art of Managing Complexity
- The Digital Abstraction
- Number Systems
- Data Storage/Representations
- Logic Gates

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 $300 billion in 2011
- Also national security matters

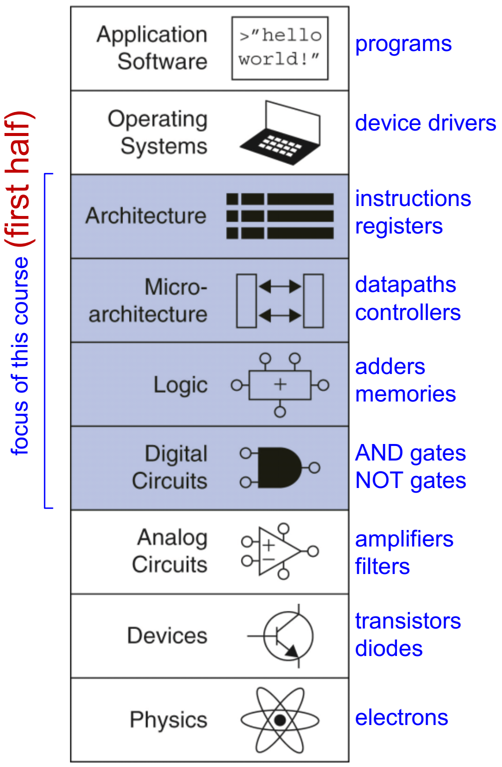



The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –Y's
  - Hierarchy
  - Modularity
  - Regularity

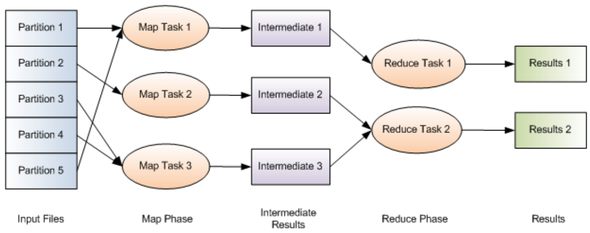
Abstraction

- Hiding details when they are not important
- Advancing a complex project for many details



Discipline

- Intentionally "restrict" design choices (for some advantages)
- Hardware example: Digital discipline
  - Discrete voltages instead of continuous
  - Simpler to design than analog circuits – can build more sophisticated systems; how to convert the analog to
  - Digital systems replacing analog predecessors:
    - i.e., digital cameras, digital television, cell phones, CDs; or further e-car, e-com, etc.
- Software example: Map/Reduce (or functional programming)
  - Breaking data/computing units for parallel processing
- CISC vs. RISC



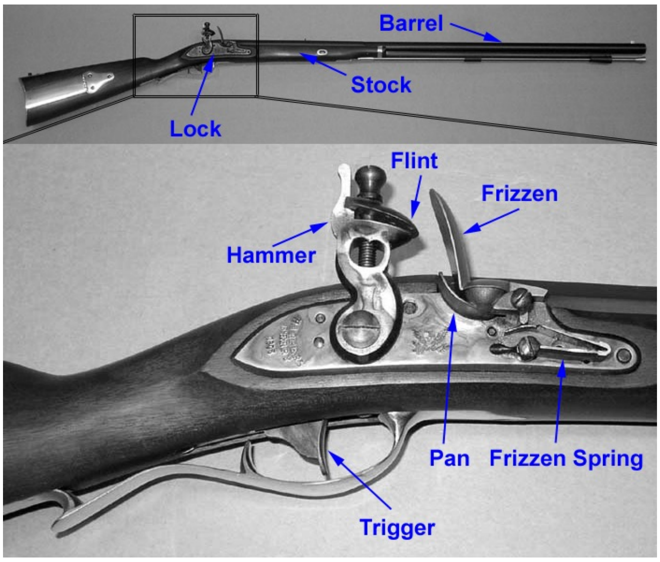
http://cdmh.co.uk/papers/software_scalability_mapreduce/

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
  - Recently MCU/ECU shortage due to the matters

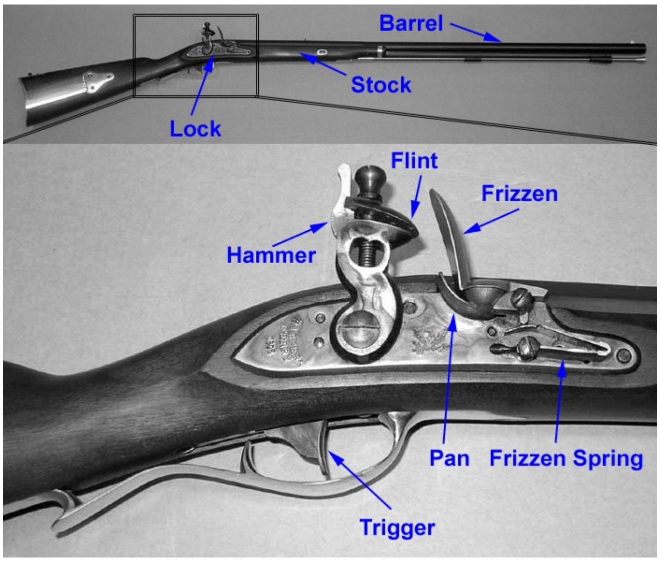
Example: The Flintlock Rifle

- Hierarchy
  - Three main modules: lock, stock, and barrel
  - Submodules of lock: hammer, flint, frizzen, etc.
- Can be modelled as a graph for many computing models e.g., GNN



Example: The Flintlock Rifle

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

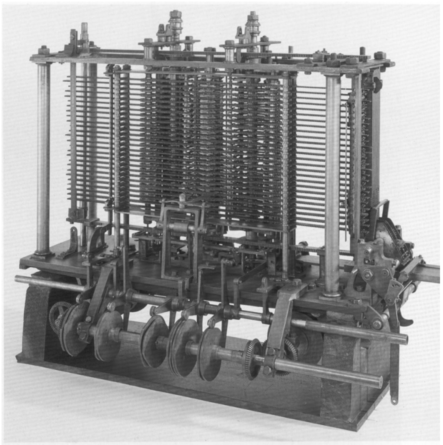


The Digital Abstraction

- Most physical variables are **continuous (analog)**
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
  - Image/audio capturing, written documents, etc.
- 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

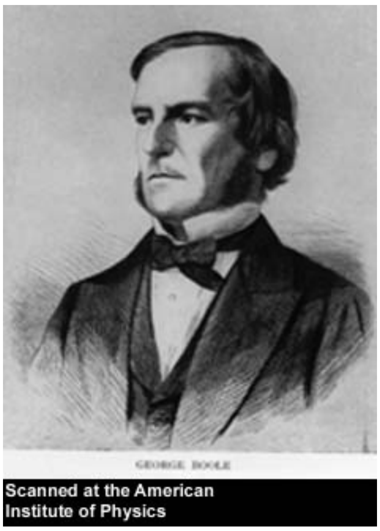


Digital Discipline: Binary Values

- **Two discrete values:**
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- **1 and 0:** voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use **voltage** levels to represent 1 and 0
- ***Bit***: Binary digit
- The future → Quantum Computing

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.



Number Systems

- Decimal numbers

1000's column
100's column
10's column
1's column

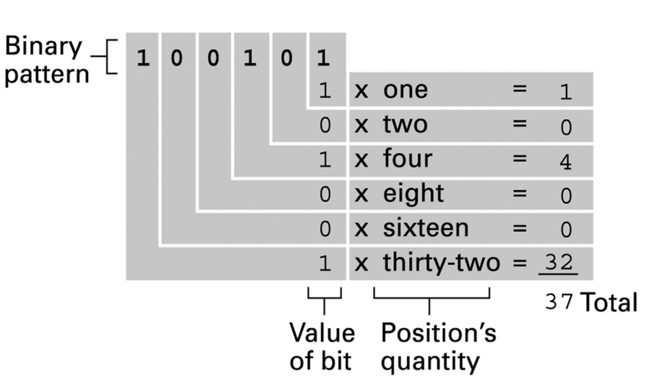
$$5374_{10} =$$

- Binary numbers

8's column
4's column
2's column
1's column

$$1101_2 =$$

Number Systems



- Decimal numbers

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 thousands, three hundreds, seven tens, four ones

- Binary numbers

8's column
4's column
2's column
1's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight, one four, no two, one one

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

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

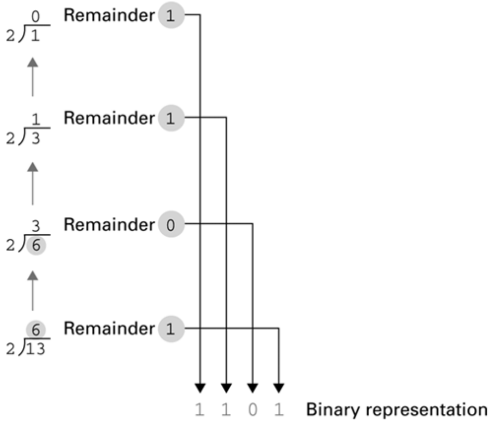
Number Conversion

- Decimal to binary conversion:
  - Convert 10011_2 to decimal

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

Number Conversion

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



- Decimal to binary conversion:
  - Convert 47_{10} to binary
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$
  - Another way is to iteratively substract the less and closest number (power of 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

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	

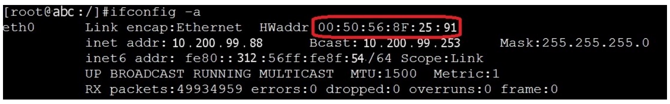
Hexadecimal Numbers

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

- Base 16
- Shorthand for binary (readable)

Error for debugging (memory address)



Mac address

Hexadecimal Numbers

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

Hexadecimal Numbers

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  - $0100\ 1010\ 1111_2$
- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$

Bits, Bytes, Nibbles…

- Bits

10010110

most significant bit

least significant bit

- Bytes & Nibbles

byte

10010110

nibble

- Bytes

CEBF9AD7

most significant byte

least significant byte

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)

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

Addition

- Decimal

```
   11  ← carries
   3734
 + 5168
 ------
   8902
```

- Binary

```
   11  ← carries
   1011
 + 0011
 ------
   1110
```

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

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

- 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
  - Some are problems and some are not
  - Common coding errors (e.g., illegal memory access, etc.)
- See previous example of 11 + 6

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two’s Complement Numbers

Sign/Magnitude Numbers

- 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/mag representations of ± 6:
  - +6 =
  - \- 6 =
- Range of an *N*-bit sign/magnitude number:

Sign/Magnitude Numbers

- 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/mag 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:

```
   1110
 + 0110
 ------
  10100 (wrong!)
```

- Two representations of 0 (± 0):

  1000

  0000

Two's Complement Numbers

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

Two’s Complement Numbers

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

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

- Most positive 4-bit number:
- Most negative 4-bit number:
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two’s comp number:

Two’s Complement Numbers

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

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

- Most positive 4-bit number: 0111
- Most negative 4-bit number: 1000
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two’s comp number:

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

Decimal	Bit Pattern
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

16 bit range
-32,768 to 32,767

Taking the Two's Complement

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

Taking the Two's Complement

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

$$\begin{array}{r} 1100 \\ +\quad 1 \\ \hline 1101 = -3_{10} \end{array}$$

Another View of Tow's Complement

$$\bar{x} = \begin{cases} 2^n - x & if x \neq 0 \\ 0 & otherwise \end{cases}$$

- If n = 5, -2's two's complement can be
  - $2^5 - 2 = 30 = (11110)$

Two's Complement

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

- What is the decimal value of 1001_2?

Two's Complement

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

$$\begin{array}{r} 1001 \\ + \quad 1 \\ \hline 1010_2 = -6_{10} \end{array}$$

- What is the decimal value of 1001_2?

$$\begin{array}{l} 1001 \\ - \quad 1 \\ \hline 1000_2 \text{ (invert bits)} \rightarrow 0111_2 = 7_{10} \\ \text{so } 1001_2 = -7 \end{array}$$

Two's Complement Addition

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

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

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

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

Two's Complement Addition

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

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

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

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


More Practices (n=4)

Problem in base ten		Problem in two's complement		Answer in base ten
3 + 2	→	0011 + 0010 = 0101	→	5
−3 + −2	→	1101 + 1110 = 1011	→	−5
7 + −5	→	0111 + 1011 = 0010	→	2

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

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 = 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								

Representing Text

- Each character (letter, punctuation, etc.) is assigned a unique bit pattern.
  - **ASCII**: Uses patterns of 7-bits to represent most symbols used in written English text
  - ISO developed a number of 8 bit extensions to ASCII, each designed to accommodate a major language group
  - **Unicode**: Uses patterns up to 21-bits to represent the symbols used in languages world wide, 16-bits for world's commonly used languages
- Example,

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

ASCII Table (& Extended)

- You don't need to memorize them!
- Useful when doing regular expression, terminal I/O, programming, etc.

Regular ASCII Chart (character codes 0 – 127)

000 (nul)	016 ► (dle)	032 sp	048 0	064 @	080 P	096 `	112 p
001 ☺ (soh)	017 ◄ (dc1)	033 !	049 1	065 A	081 Q	097 a	113 q
002 ☻ (stx)	018 ↕ (dc2)	034 "	050 2	066 B	082 R	098 b	114 r
003 ♥ (etx)	019 ‼ (dc3)	035 #	051 3	067 C	083 S	099 c	115 s
004 ♦ (eot)	020 ¶ (dc4)	036 $	052 4	068 D	084 T	100 d	116 t
005 ♣ (enq)	021 § (nak)	037 %	053 5	069 E	085 U	101 e	117 u
006 ♠ (ack)	022 ▬ (syn)	038 &	054 6	070 F	086 V	102 f	118 v
007 • (bel)	023 ↨ (etb)	039 '	055 7	071 G	087 W	103 g	119 w
008 ◘ (bs)	024 ↑ (can)	040 (	056 8	072 H	088 X	104 h	120 x
009 (tab)	025 ↓ (em)	041)	057 9	073 I	089 Y	105 i	121 y
010 (lf)	026 (eof)	042 *	058 :	074 J	090 Z	106 j	122 z
011 ♂ (vt)	027 ← (esc)	043 +	059 ;	075 K	091 [	107 k	123 {
012 ♀ (np)	028 ∟ (fs)	044 ,	060 <	076 L	092 \	108 l	124 |
013 (cr)	029 ↔ (gs)	045 -	061 =	077 M	093]	109 m	125 }
014 ♫ (so)	030 ▲ (rs)	046 .	062 >	078 N	094 ^	110 n	126 ~
015 ☼ (si)	031 ▼ (us)	047 /	063 ?	079 O	095 _	111 o	127 ⌂

http://dochome.programbl.com/rs232-cps-plus/tools-ascii-table-standard-and-extended-ascii-table.html

Extended ASCII Table

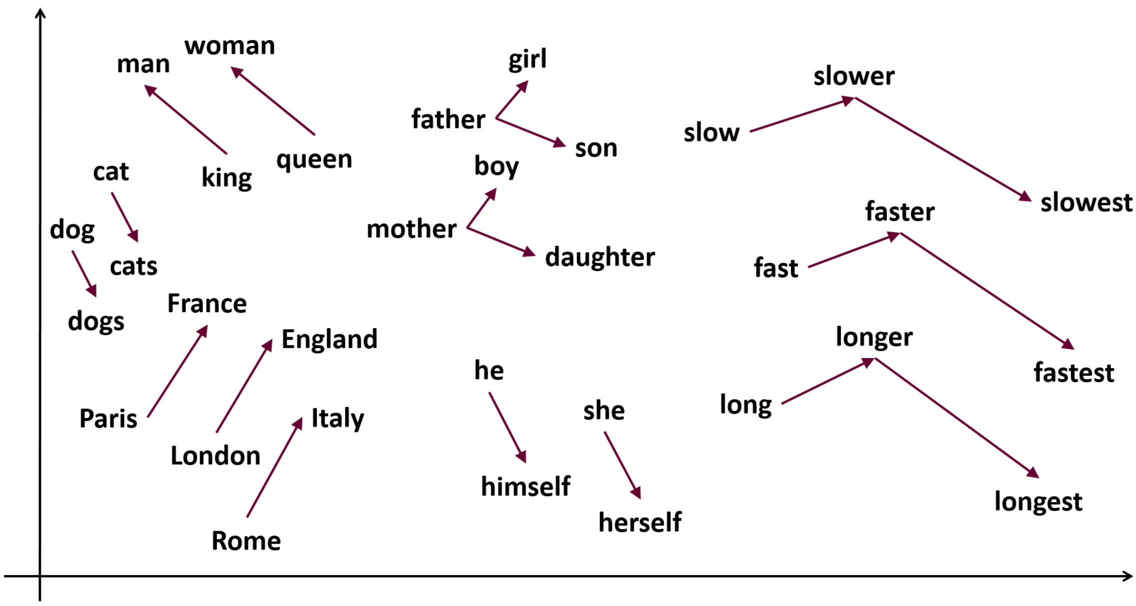
- You don’t need to memorize them!
- Useful when doing regular expression, terminal I/O, programming, etc.

Extended ASCII Chart (character codes 128 - 255)

128	Ç	143	Å	158	₧	172	¼	186	║	200	╚	214	╓	228	Σ	242	≥
129	ü	144	É	159	ƒ	173	¡	187	╗	201	╔	215	╫	229	σ	243	≤
130	é	145	æ	160	á	174	«	188	╝	202	╩	216	╪	230	µ	244	⌠
131	â	146	Æ	161	í	175	»	189	╜	203	╦	217	┘	231	τ	245	⌡
132	ä	147	ô	162	ó	176	░	190	╛	204	╠	218	┌	232	Φ	246	÷
133	à	148	ö	163	ú	177	▒	191	┐	205	═	219	█	233	Θ	247	≈
134	å	149	ò	164	ñ	178	▓	192	└	206	╬	220	▄	234	Ω	248	°
135	ç	150	û	165	Ñ	179	│	193	┴	207	╧	221	▌	235	δ	249	∙
136	ê	151	ù	166	ª	180	┤	194	┬	208	╨	222	▐	236	∞	250	·
137	ë	152	ÿ	167	º	181	╡	195	├	209	╤	223	▀	237	φ	251	√
138	è	153	Ö	168	¿	182	╢	196	─	210	╥	224	α	238	ε	252	ⁿ
139	ï	154	Ü	169	⌐	183	╖	197	┼	211	╙	225	ß	239	∩	253	²
140	î	155	¢	170	¬	184	╕	198	╞	212	╘	226	Γ	240	≡	254	■
141	ì	156	£	171	½	185	╣	199	╟	213	╒	227	π	241	±	255	
142	Ä	157	¥														

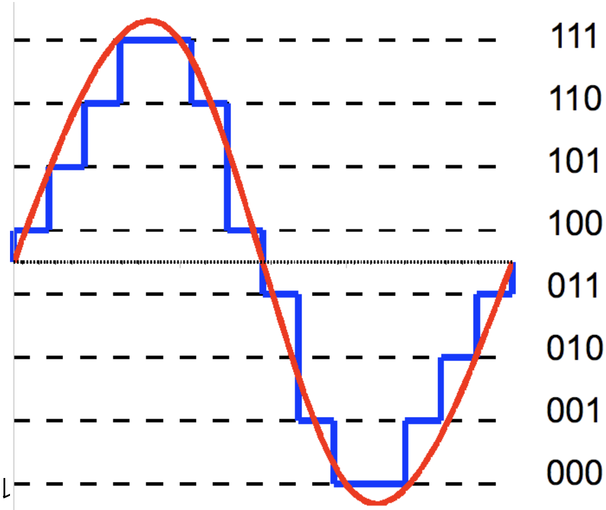
Representing Text (More Semantic Rich Representations)

- Distributed word embedding (e.g., word2vec); a word in a high-dimensional feature; strong representations with **BERT**



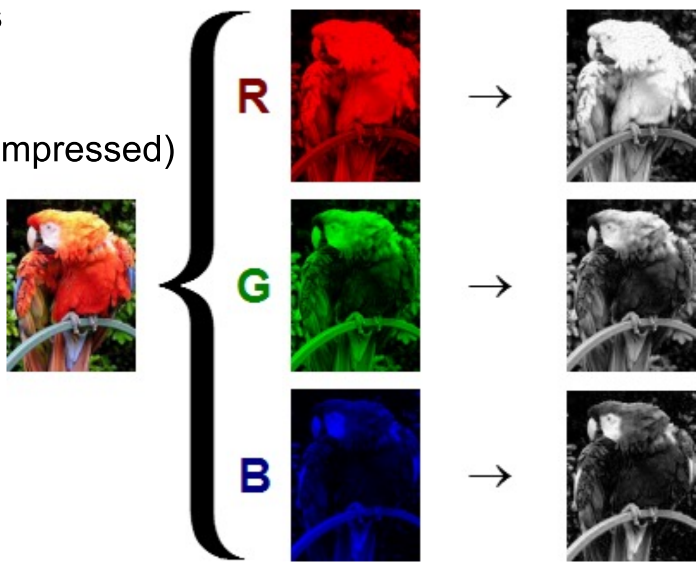
Representing Sound

- Sampling techniques
  - Nyquist theorem to ensure quality
  - Used for high quality recordings
  - Records actual audio
- Quantization is the key
  - Analog amplitudes -> bits
  - Factors
    - Bits per sample
    - Sampling frequency (times/sec)
    - 電話音質(11,025Hz, 8bits, Mono)、
      CD音質(44,100Hz, 16bits, Stereo)、
    - DVD音質(96kHz, 24bits, Stereo)。
  - How to calculate music size? 5 minute audio recording

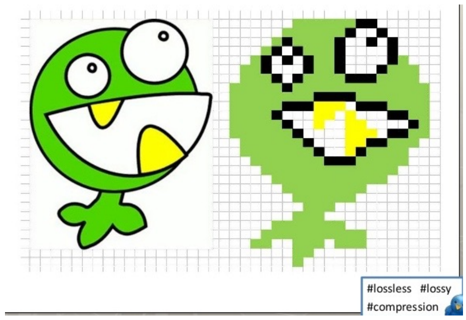


Representing Images

- Color channels in RGB (also HSV, Lab, YUV, etc.)
- Each pixel is represented as the value in [0, 255]; 8 bits
- True color: 2^{24}
- What is the size for the (uncompressed) image
- The image size is bulky → requiring compression



Data Compression



- Lossy versus lossless
  - Sometimes you do not notice
- Run-length encoding
  - AAAABBBCCDEEEE → 4A3B2C1D4E
- Frequency-dependent encoding (Huffman codes)
  - High freq. words → shorter codes; information theory
- Relative encoding (e.g., DPCM)
  - (91,91,99,98,91,93,92,92) → (91,0,8,-1,-7,2,-1,0)
- Dictionary encoding (Includes adaptive dictionary encoding such as LZW encoding.)

Color Subsampling (Lossy) by Optimizing Human Perception


Compressed


Y Channel


Cb Channel


Cr Channel

- Intuition – our eyes are much better at detecting spatial changes in the luminance (intensity) than that in the chrominance (color)
- Compression artifacts, best seen by looking for the jagged diagonal lines in the Cb and Cr images.
- The image after compression, far-left image, looks “perfect” even though there are visible artifacts in the Cb and Cr subimages.

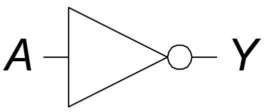
* From Ponceleon and Slaney

Logic Gates

- **Perform logic functions:**
  - inversion (NOT), AND, OR, NAND, NOR, etc.
- **Single-input:**
  - NOT gate, buffer
- **Two-input:**
  - AND, OR, XOR, NAND, NOR, XNOR
- **Multiple-input**

Single-Input Logic Gates

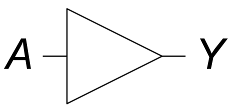
NOT



$$Y = \overline{A}$$

A	Y
0	
1	

BUF

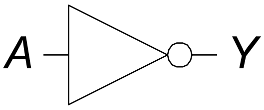


$$Y = A$$

A	Y
0	
1	

Single-Input Logic Gates

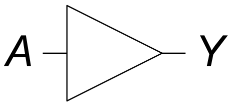
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

BUF

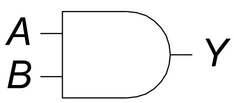


$$Y = A$$

A	Y
0	0
1	1

Two-Input Logic Gates

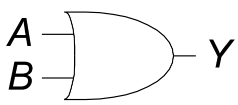
AND



$$Y = AB$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR

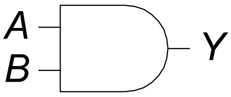


$$Y = A + B$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Two-Input Logic Gates

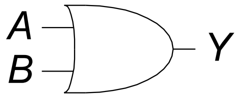
AND



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

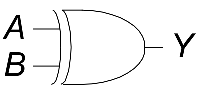


$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

More Two-Input Logic Gates

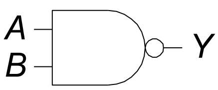
XOR



$Y = A \oplus B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

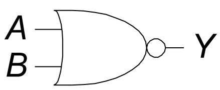
NAND



$Y = \overline{AB}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

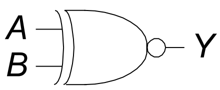
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR

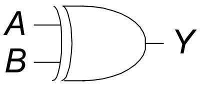


$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

More Two-Input Logic Gates

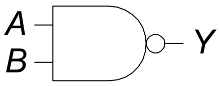
XOR



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

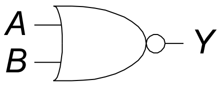
NAND



$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

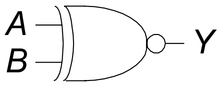
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

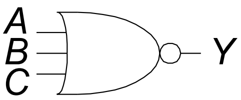


$Y = \overline{A \oplus B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Multiple-Input Logic Gates

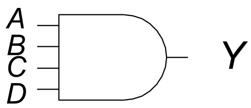
NOR3



$Y = \overline{A+B+C}$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

AND4

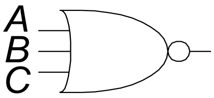


$Y = ABCD$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiple-Input Logic Gates

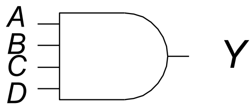
NOR3



$Y = \overline{A+B+C}$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

AND4



$Y = ABCD$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- Multi-input XOR: Odd parity

To-Do Items

- Today’s slides are available in the NTU COOL webpage
- Homework (#1) CV due on coming Monday (Sept. 19; **online sub.)**
- Homework (#2) available now
  - **DUE: at 12pm, Sept. 26, 2022 (Monday); end of the lecture time; hard copy**
- Next week: Boolean Logic (Sept. 19)
  - Chapter 2 (parts) from Digital Design and Computer Architecture, 2nd Edition, David Harris and Sarah Harris.
  - Chapter 1 and 2 (parts) from The Elements of Computing Systems, Nisan, et al.
  - Or Chapter 1.1-1.2 from Computer Science: An Overview, 12th Edition, by J. Glenn Brookshear