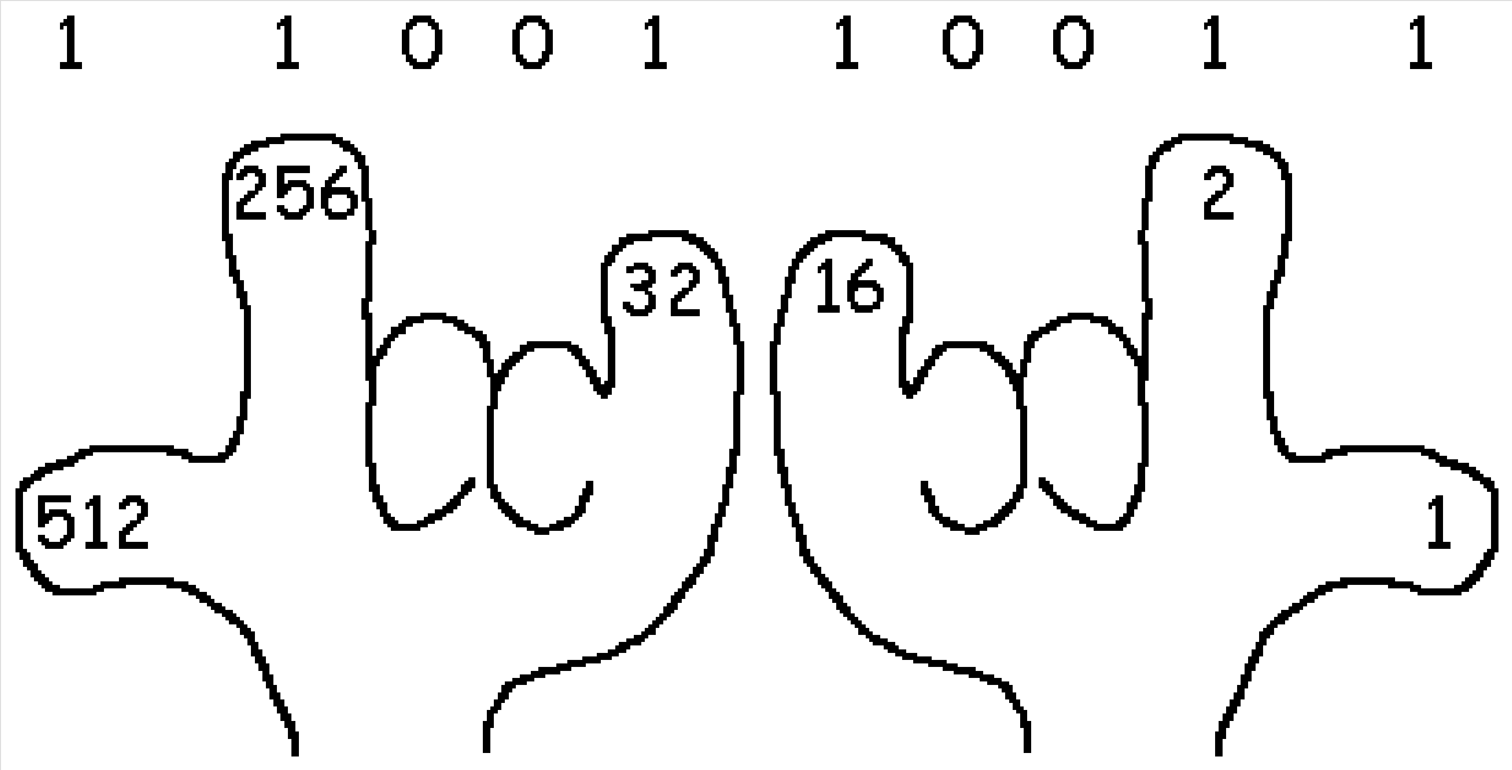
Part 2 of CMSC 11

- **More CS concepts**
  - How data and programs are internally represented, including **number systems**
  - How computer hardware is organized
  - How programs are compiled, loaded and executed
  - Overview of other programming languages
- **More algorithms and C programming**
  - Using pre-defined functions: commonly used functions involving math and strings
  - Modular programming: top-down design of bigger programs using functions
  - Basic data structures: arrays, arrays of arrays, and their applications

Number systems

- Most people prefer the decimal number system (mainly because we have 10 fingers!)
  - 234.56 means 2*100 + 3*10 + 4*1 + 5/10 + 6/100
  - We have 10 digits {0..9} and we use powers of 10
- Computers have “on-off switches” instead of fingers and so they use the binary system
  - They use 2 binary digits {0,1} and they use powers of 2
  - 1101 means 1*8 + 1*4 + 0*2 + 1*1
  - 110.101 means 1*4 + 1*2 + 0*1 + 1/2 + 0/4 + 1/8

How high can you count with your ten fingers?



www.mathmaniacs.org/lessons/01-binary/fingers.gif

Converting binary to decimal

- Converting a binary number to its decimal equivalent is performed by adding the successive powers of 2 where the bits (binary digits) are on.

1 0 0 0 1 1 0 1

128 64 32 16 8 4 2 1

128+8+4+1 = 141 (in decimal)

Binary to decimal conversion
powers-of-two algorithm

```
input binary integer;
decimal = 0;
power = 1;
for (each binary digit from right to left) {
   if (current bit == 1)
          decimal = decimal + power;
   power = power*2;
}
output decimal;
```

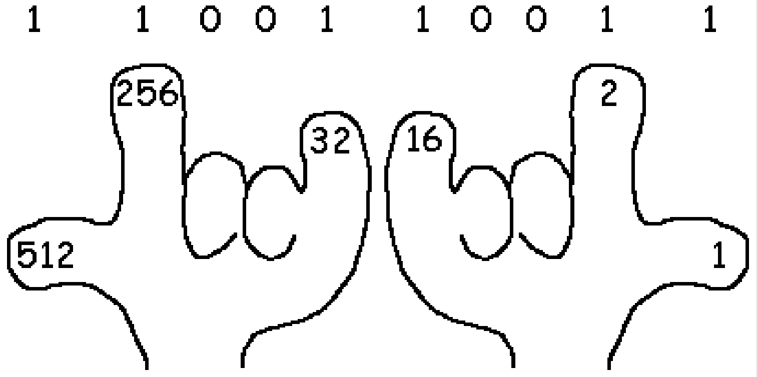
1101	power = 1, decimal = 1
1101	power = 2, decimal = 1
1101	power = 4, decimal = 5
1101	power = 8, decimal = 13

Binary to decimal conversion
another algorithm

```
input binary integer;
decimal = 0;
for (each binary digit from left to right) {
   if (current bit == 0)
      decimal = decimal*2;
  else
      decimal = decimal*2 + 1;
}
output decimal;
------------------------------------------------------------
1101     decimal = 2*0+1 = 1
1101     decimal = 2*1+1 = 3
1101     decimal = 2*3 = 6
1101     decimal = 2*6+1 = 13
```

How high can you count with your ten fingers?

With 1 bit, there are only 2 possible values: 0 and 1
With 2 bits, 4 possible values: 00, 01, 10, 11 (0 to 3)
With 3 bits, 8 possible values: 000, 001, 010, 011, 100, 101, 110, 111
With 4 bits, 16 possible values
With 5 bits, 32 possible values
With 6 bits, 64 possible values
With 7 bits, 128 possible values
With 8 bits, 256 possible values
With 9 bits, 512 possible values
With 10 bits or fingers, there are 1024 possible values from 00000 00000 to 11111 11111 (0 to 1023)



What about decimal to binary?

Example: What is 300 in binary?

One way is by repeated subtraction of powers of two:

300 – 256 =44	256
44 – 32 = 12	32
12 – 8 = 4	8
4 – 4 = 0	4

300 in decimal is equivalent to 1 0010 1100 in binary

What about decimal to binary?

Example: What is 300 (decimal) in binary?

Another algorithm is by repeated integer division by 2

300/2 = 150 r 0

150/2 = 75 r 0

75/2 = 37 r 1

37/2 = 18 r 1

18/2 = 9 r 0

9/2 = 4 r 1

4/2 = 2 r 0

2/2 = 1 r 0

1/2 = 0 r 1 .. **1 0010 1100 (sequence of remainders)**

(check by converting back to decimal...)

Base 8 (octal) and Base 16 (hexadecimal)

- 8 octal digits are 0, 1, 2, 3, 4, 5, 6, 7
- 16 hexadecimal digits are
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- What's special about base 8 and base 16?
- Being powers of 2, conversions between binary, octal, and hex number systems are fairly easy
- Idea is to group the bits by 3's for octal numbers, or to group the bits by 4's for hex

100 110 111 (binary) is equivalent to 467 (octal)
1 0011 0111 (binary) is equivalent to 137 (hex)
... check that these are also equivalent 311 (decimal)

Basic conversions

Dec	Bin	Oct	Hex
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Dec	Bin	Oct	Hex
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

Example

- Convert 34 (octal) into the other number systems
- Octal to binary: form the binary digits by 3's.....
  011 100 (binary)
- Binary to hex: regroup the bits into 4's, then translate to hex: 0001 1100 = 1C (hex)
- Hex to decimal: use powers of 16's
  1C (hex) = 1*16 + 12 = 28 (decimal)
- Check: 34 (octal) = 3*8 + 4 = 28 (decimal)

Number systems in C programming

- printf() number formats
  “%d”   print as a decimal number
  “%o”   print as an octal number
  “%x”   print as a hexadecimal number

```
int x = 28;
printf("%d decimal = %o octal = %x hex\n", x, x, x);
```

output:
28 decimal = 34 octal = 1c hex

Number systems in C programming

- Constant numbers prefix
  0x for hex,   example: 0x1a (hex) = 26 (decimal)
  0   for octal,   017 (oct) = 15 (decimal)

```
int  p = 0x1a, q = 017;
printf("%d decimal = %o octal = %x hex\n", p, p, p);
printf("%d decimal = %o octal = %x hex\n", q, q, q);
```

output:
26 decimal = 32 octal = 1a hex
15 decimal = 17 octal = f hex

Arithmetic operations

- Binary arithmetic is easy

+	0	1
0	0	1
1	1	10

x	0	1
0	0	0
1	0	1

- Exercise: Write C programs for similar addition and multiplication tables for the octal and hex number systems (create 8x8 and 16x16 tables .... your chance to practice programming using nested loops)

Octal Addition Table

+:	0	1	2	3	4	5	6	7
0:	0	1	2	3	4	5	6	7
1:	1	2	3	4	5	6	7	10
2:	2	3	4	5	6	7	10	11
3:	3	4	5	6	7	10	11	12
4:	4	5	6	7	10	11	12	13
5:	5	6	7	10	11	12	13	14
6:	6	7	10	11	12	13	14	15
7:	7	10	11	12	13	14	15	16

Addition examples

10 101 (bin)	25 (oct)	21 (dec)
+ 10 100	+ 24	+ 20
------------	----------	----------
101 001	51	41

1 0110 (bin)	16 (hex)	22 (dec)
+ 1 0111	+ 17	+ 23
------------	----------	----------
10 1101	2D	45

Bitwise logical operators

- Bitwise 1's complement ~ Ex: ~1100 = 0011
- Bitwise AND & Ex: 1100 & 1010 = 1000
- Bitwise OR | Ex: 1100 | 1010 = 1110
- Bitwise eXclusive OR ^ Ex: 1100 ^ 1010 = 0110
- Left shift << Ex: 1001 << 1 = 10010
- Right shift >> Ex: 11000 >> 2 = 110

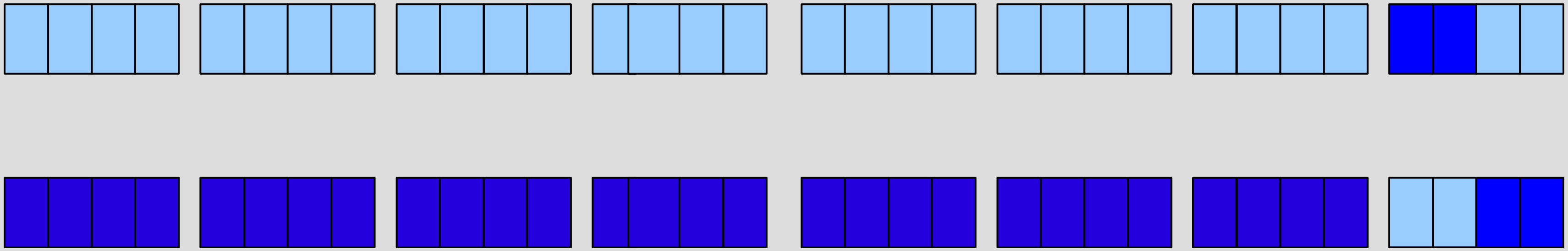
Run the ff. code fragment and try to explain its output

```
int j, p = 1;
for (j=0; j<32; j++) {
    printf("%d %d\n", j, p);
    p = p << 1;
}
```

Bits, nibbles and bytes

- Many processors now represent integers as 32-bit numbers. This is equivalent to 4 groups of 8-bit bytes, or 8 groups of 4-bit nibbles. Can you now explain the ff. code and its output?

- `printf("complement of %x is %x", 0xC, ~0xC);`
- `complement of c is fffffff3`



Representation of negative numbers

- Negative numbers are often stored in the computer's memory using the so-called 2's complement representation
- This is obtained by taking the 1's complement (flip all the bits) and then adding 1.

Example:	0001 1011	(under an 8-bit system)
1's complement	1110 0100	
2's complement	1110 0101	

Representation of negative numbers

Positive and negative numbers under a 4-bit system

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

- Note that the left-most bit acts a sign bit (1 = negative), and that n bits can represent all integers in [-2^{n-1}....$2^{n-1}-1$]

Addition examples

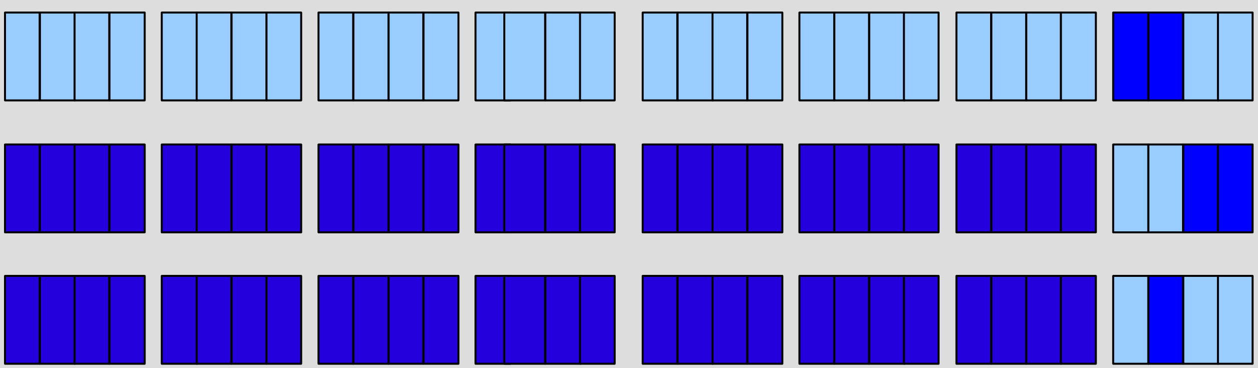
```
  0011          3
+ 1101       + -3
---------    -----
 10000          0
(with overflow)

  0010          2
+ 1011       + -5
---------    -----
  1101         -3
```

Representation of negative numbers

Can you now explain the ff. code and its output?

```
printf("negative of %d is %d\n", 0xc, ~0xc + 1);
negative of 12 is -12
```



C (hex) = 12 (decimal)

1's complement of C

2's complement of C

sizeof(int) = 4 bytes = 32 bits for many PCs

Representing chars

- **Plain ASCII code is a 7-bit code** to represent the most common characters
- **Extended ASCII uses 8 bits** to include certain additional chars like ñ, arrows, and lines
- **Unicode uses 16 bits** in order to represent practically all character sets (including Japanese, Korean, Arabic, etc)

```
int c;
for (c=0; c<128; c++)  {
   printf("char %c = decimal %d = hex %x\n", c, c, c);
}
```

Part of the ASCII character set

```
Chr Ctrl  Dec Hex    Chr  Dec Hex    Chr  Dec Hex    Chr   Dec Hex

NUL  ^@    0   0     SP   32  20      @   64  40      `     96  60
SOH  ^A    1   1      !   33  21      A   65  41      a     97  61
STX  ^B    2   2      "   34  22      B   66  42      b     98  62
ETX  ^C    3   3      #   35  23      C   67  43      c     99  63
EOT  ^D    4   4      $   36  24      D   68  44      d    100  64
ENQ  ^E    5   5      %   37  25      E   69  45      e    101  65
ACK  ^F    6   6      &   38  26      F   70  46      f    102  66
BEL  ^G    7   7      '   39  27      G   71  47      g    103  67

BS   ^H    8   8      (   40  28      H   72  48      h    104  68
HT   ^I    9   9      )   41  29      I   73  49      i    105  69
LF   ^J   10   A      *   42  2A      J   74  4A      j    106  6A
.....                 .....           .....           .....

CAN  ^X   24  18      8   56  38      X   88  58      x    120  78
EM   ^Y   25  19      9   57  39      Y   89  59      y    121  79
SUB  ^Z   26  1A      :   58  3A      Z   90  5A      z    122  7A
ESC  ^[   27  1B      ;   59  3B      [   91  5B      {    123  7B
FS   ^\   28  1C      <   60  3C      \   92  5C      |    124  7C
GS   ^]   29  1D      =   61  3D      ]   93  5D      }    125  7D
RS   ^^   30  1E      >   62  3E      ^   94  5E      ~    126  7E
US   ^_   31  1F      ?   63  3F      _   95  5F     DEL   127  7F
```

http://www.mhuffman.com/notes/numbers/numrep.htm

Code Charts - Scripts - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.unicode.org/charts/

Mandriva Mandriva Store

Code Charts - Scripts

Many other character sets are available in Unicode

Cyrillic	Vai	Ol Chiki	(see also **Unihan Database**)	***Cuneiform***
Cyrillic Supplement	**Middle Eastern Scripts**	Oriya	***Radicals and Strokes***	Cuneiform
Cyrillic Extended A	***Arabic***	Saurashtra	CJK Radicals	Cuneiform Numbers
Cyrillic Extended B	Arabic	Sinhala	KangXi Radicals	Old Persian
Georgian	Arabic Supplement	Syloti Nagri	CJK Strokes	Ugaritic
Georgian	Arabic Present. Forms A	Tamil	Ideographic Description	***Linear B***
Georgian Supplement	Arabic Present. Forms B	Telugu	***Chinese-specific***	Linear B Syllabary
Greek	***Hebrew***	**South East Asian**	Bopomofo	Linear B Ideograms
Greek	Hebrew	Balinese	Bopomofo Extended	***Other Ancient Scripts***
Greek Extended	*Hebrew Present. Forms*	Buginese	***Japanese-specific***	Aegean Numbers
(see also ***Ancient Greek***)	***Syriac***	Cham	Hiragana	Ancient Symbols
Latin	Syriac	Kayah Li	Katakana	Carian
Basic Latin	***Thaana***	Khmer	Katakana Phonetic Ext.	Counting Rod Numerals
Latin-1	Thaana	Khmer Symbols	*Halfwidth Katakana*	Cypriot Syllabary
Latin Extended A	**American scripts**	Lao	***Korean-specific***	Glagolitic
Latin Extended B	Canadian Syllabics	Myanmar	Hangul Syllables (4MB)	Gothic
Latin Extended C	Cherokee	New Tai Lue	Hangul Jamo	Lycian
Latin Extended D	Deseret	Rejang	Hangul Compatibility Jamo	Lydian
Latin Extended Additional	**Philippine Scripts**	Sundanese	Halfwidth Jamo	Ogham
Latin Ligatures	Buhid	Tai Le	***Yi***	Old Italic
Fullwidth Latin Letters	Hanunoo	Thai	Yi (.6MB)	Phaistos Disc
Small Forms	Tagalog	**Other Scripts**	Yi Radicals	Phoenician
*(see also **Phonetic Symbols**)*	Tagbanwa	Shavian		Runic

To get a list of code charts for a character, enter its code in the search box at the top. To access a chart for a given block, click on its entry in the table. The charts are PDF files, and some of them may be very large. For frequent access to the same chart, right-click

http://www.unicode.org/charts/PDF/U3190.pdf file:///home/jms/tmp/U1700-1.pdf

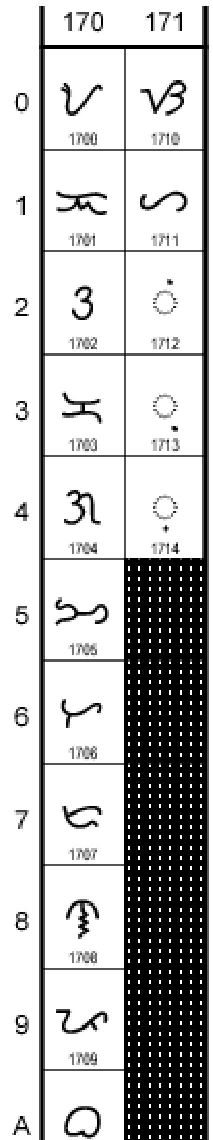
File Edit View Settings Help

100%

Page

1

2



Independent vowels

1700 TAGALOG LETTER A
1701 TAGALOG LETTER I
1702 TAGALOG LETTER U

Consonants

1703 TAGALOG LETTER KA
1704 TAGALOG LETTER GA
1705 TAGALOG LETTER NGA
1706 TAGALOG LETTER TA
1707 TAGALOG LETTER DA
1708 TAGALOG LETTER NA
1709 TAGALOG LETTER PA
170A TAGALOG LETTER BA
170B TAGALOG LETTER MA
170C TAGALOG LETTER YA
170D <reserved>
170E TAGALOG LETTER LA
170F TAGALOG LETTER WA
1710 TAGALOG LETTER SA
1711 TAGALOG LETTER HA

Dependent vowel signs

1712 TAGALOG VOWEL SIGN I
1713 TAGALOG VOWEL SIGN U

Virama

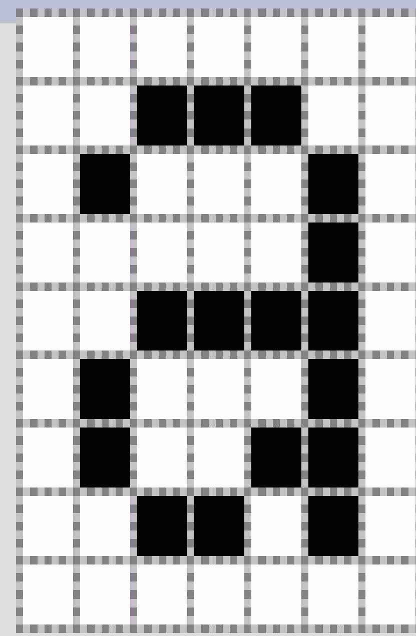
1714 TAGALOG SIGN VIRAMA

Unicode for Ancient Tagalog Scripts 1700-171F

www.unicode.org

Page 2 of 2

Representing images



0000000
0011100
0100010
0000010
0011110
0100010
0100110
0011010
0000000

A **black & white** image 7 pixels wide and 9 pixels high can be represented as a sequence of 63 bits.
How many bits per pixel do we need if we want 16 **shades of gray**? Or if we want 256 different **colors**?

http://www.mhuffman.com/notes/numbers/numrep.htm