

10011110000011001101001011110001100001110000001111100000000110100111000011100001000011

TECH 3233 Lecture 2

Dan Kohn
University of Memphis

Modified from: <http://resources.dpsu.org/ASA2/Computing/NumberBases.cw>
References: <http://homepage.mac.com/stephudejebater.cs.ucr.edu/www.stofarm.com/DOS/6x4Desktopx.html>

10011110000011001101001011110001100001110000001111100000000110100111000011100001000011

Binary and Decimal Recap

1. Decimal Numbers
2. Binary Numbers
3. Converting to Decimal
4. Converting to Binary
5. Questions

Decimal

0010011110000011001101001011110001100001110000011111000000001101001110000111000010000

- Decimal is the number system we use, base 10.
- We have 10 possible digits, 0-9.
- Each digit in a number has a different place value, working from the right hand side place value increases in powers of 10.
- For example... with the Decimal number 5106

10^3	10^2	10^1	10^0
1000	100	10	1
5	1	0	6

- And so is made up of
 - $(5 \times 1000) + (1 \times 100) + (0 \times 10) + (6 \times 1) = 5106$

100100111100000110011010010111100110011000111000001111100000001101001110000111

Binary

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- Binary is the system used by electronics, i.e. a circuit is either on or off.
- Binary has just two possible digits, 1 or 0.
- Each digit in a number has a different place value, working from the right hand side place value increases in powers of 2.
- For example, the binary 1011 is...

2^3	2^2	2^1	2^0
8	4	2	1
1	0	1	1

- And so is made up of
 - $(1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1) = 11$

100100111100000110011010010111100110011000110000110000011111000000011010011100001110

Binary to Decimal

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- So to convert from binary to Decimal you add up the value of each digit.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Dec.
128	64	32	16	8	4	2	1	
1	1	0	0	1	0	0	1	<u>201</u>

- The value in Decimal is...
 - $(1 \times 128) = 128$
 - $(1 \times 64) = 64$
 - $(0 \times 32) = 0$
 - $(0 \times 16) = 0$
 - $(1 \times 8) = 8$
 - $(0 \times 4) = 0$
 - $(0 \times 2) = 0$
 - $(1 \times 1) = 1$
- Total is $128 + 64 + 0 + 0 + 8 + 0 + 0 + 1 = 201$

100100111100000110011010010111100110011000110000110000011111000000011010011100001110

Decimal to Binary

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- Generate the same table, but keep going until the last column is greater than the value to be converted

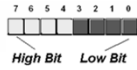
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	0	1	0	0

- Now, starting at the left, find the first value that is less than the number to convert. Put a one in that column and, off to the side subtract that number from the original value. Then find the first value that is smaller than the new value. Keep repeating until the value of zero is reach.
- Example: Convert 100 dec to binary
 - $100 - 64 = 36$
 - $36 - 32 = 4$
 - $4 - 4 = 0$

100100111100000110011010010111100110011000110000110000011111000000011010011100001110

Binary Numbers (Definitions)

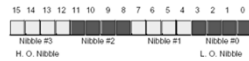
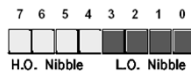
- The bit to the left is the High Bit
- The bit to the right is the Low Bit



1/23/20 Number Systems 7

Binary Numbers (Definitions)

- Similarly the nibble to the left is the High Nibble
- The nibble to the right is the Low Nibble



1/23/20 Number Systems 8

Bitwise Functions

0010011111000001110011010010111100011000011100001110000000111110000000110100111000011100001000011

Hexadecimal

00100111110000011100110100101111000110000111000000111110000000110100111000011100001000011

1. Hexadecimal Numbers
2. Why Use Hexadecimal
3. Converting to Binary
4. Converting from Binary
5. Reference Table
6. Questions

00100111110000011100110100101111000110000111000000111110000000110100111000011100001000011

Hexadecimal Numbers

00100111110000011100110100101111000110000111000000111110000000110100111000011100001000011

- In hexadecimal (hex for short) we have 16 different digits, 0-9 (like Decimal) and then an additional 6 digits. We use letters of the alphabet to represent these additional digits.
- In hex place value increases in powers of 16.
- For example, the hex number C3...

16^2	16^1	16^0
256	16	1
0	C	3

- Is calculated as
 - $=(0 \times 256) + (C \times 16) + (3 \times 1)$
 - $=(0 \times 256) + (12 \times 16) + (3 \times 1)$
 - $= 195$

1001001111100000111001101001011111001100110001100001110000111000000111111000000011010011100001111

00100111110000011100110100101111000110000111000000111110000000110100111000011100001000011

Why use Hexadecimal?

00100111110000011100110100101111000110000111000000111110000000110100111000011100001000011

- Hexadecimal numbers allow for larger numbers to be stored in the same number of bits.
- This makes hex more efficient as well as easier to read than binary.
- For example, to store the number 255 in binary requires eight digits (1111111), in Decimal it requires three digits, whilst in hex it requires only two (FF)
- Each hex digits requires one nibble (four bits) to store in the computer's binary memory.

100100111110000011100110100101111100110011000110000111000000111111000000011010011100001111

Converting to Binary

- To convert to binary...
 - Each hex digit must be individually converted into a Decimal number.
 - That Decimal number is then converted into a four bit binary number.
- For example... To convert the number 5B into binary
 - 5 → 5 in Decimal → 0101 in binary
 - B → 11 in Decimal → 1011 in binary
- Therefore, 5B is shown as 0101 1011 in binary.

Converting from Binary

- To convert back from binary into hex you can split the binary number into nibbles (blocks of four bits), starting from the right hand side.
- You then convert each nibble into its Decimal equivalent, and then into a single hex digit.
- For example, to convert the number 1101 1011 into hex...
 - 1101 → 13 in Decimal → "D"
 - 1011 → 11 in Decimal → "B"
 - Therefore, the hex conversion is "DB"

Reference Table...

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

Binary Coded Decimal

001000111100000110011010010111100011000011000001111100000000110100111000011000010000

- Each Decimal Digit is represented by its 4 bit binary equivalent.
- EG:

		145		
	1	4	5	
	0001	0100	0101	
- NOTE: This is not BINARY (145 dec to binary would be 10010001)

100100111100000110011010010111100110011000110000110000011111000000001101001100001110

Time

00100011110000011001101001011110001100001100001111100000000110100111000011000010000

- We can use a techniques like bit fields and packed data to store information like the date:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	M	M	M	D	D	D	D	Y	Y	Y	Y	Y	Y	Y	Y

- We know the month can be represented by values 1-12 we use 4 bits (which can represent values 0 - 15).
- The largest month has 31 days so we use 5 bits (which can represent values 0 - 31)
- Here we represent the year using the last two digits of the year so 7 bits are used (which can represent values 0 - 127)

100100111100000110011010010111100110011000110000110000011111000000001101001100001110

ASCII

00100011110000011001101001011110001100001100001111100000000110100111000011000010000

- To represent letters we use the American Standard Code for Information Interchange (ASCII)
- Standard was adopted in 1963
- <http://asciitable.com>

100100111100000110011010010111100110011000110000110000011111000000001101001100001110

An Easier Method...

0010011111000001100110100101111000110000111000001111100000000110100111000011000010000

- A simpler way of converting a positive to a negative is to follow this method...
 - Starting from the right (least-significant bit, LSB)
 - Leave all the digits up to and including the first '1' alone.
 - Invert all the remaining digits.
- For example, negative 6 in binary is

10110	-0110	→	1010
↑ ↑ ↑ ↑			
The ones			
are			

10010011111000001100110100101111001100110001100001110000011111000000001101001100001100001110

Binary Subtraction

0010011111000001100110100101111000110000111000001111100000000110100111000011000010000

- All a processor is capable of is binary addition, all other operations can be expressed in terms of addition.
- Subtraction is the result of adding a negative number, so to do the sum 6 - 4, you do 6 PLUS negative 4.
- For example...
 - = 108 - 53
 - = 01101100 - 00110101
 - = 01101100 + 11001011
 - = 00110111

10010011111000001100110100101111001100110001100001110000011111000000001101001100001100001110

01001111100000110011010010111100011000011100000111110000000011010011100001100001000011

Binary Fractions

0010011111000001100110100101111000110000111000001111100000000110100111000011000010000

1. Fixed Point
2. Floating Point in Decimal
3. Floating Point in Binary
4. Converting from Floating Point
5. Negative Floating Point
6. Normalization
7. Normalized Binary
8. Questions

Fixed Point Fractions

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- With binary fractions place value still varies by powers of two.
- In fixed point binary the decimal place is fixed as being after a certain amount of digits.
- For example, where the binary point is fixed after four digits, the number 10011010 has a value of..

2 ³	2 ²	2 ¹	2 ⁰	.	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴
8	4	2	1	.	1/2	1/4	1/8	1/16
1	0	0	1	.	1	0	1	0

- = (1×8) + (1×1) + (1×0.5) + (1×0.0625)
- = 8 + 1 + 0.5 + 0.0625
- = 9.5625

10010011111000000110011010010111100110011000110000110000011111000000001101001100001110

Understanding Floating Point Fractions in Decimal

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- Floating point fractions are the equivalent of Scientific Notation in Decimal.
- In Scientific Notation you have a sign, a mantissa and an exponent.
- For example, - 0.65 × 10⁴
 - '-' is the sign
 - '0.65' is the mantissa
 - '4' is the exponent
 - The number is raised to the power 10, since we are in Decimal.

10010011111000000110011010010111100110011000110000110000011111000000001101001100001110

Understanding Floating Point Fractions In Binary

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- Typically 16 bits might represent a binary floating point value.
- The first 10 bits represent the mantissa, and the last 6 the exponent.
- For example 0 100000000 000111
 - '0' is the sign of the mantissa
 - '100000000' is the mantissa
 - '000111' is the exponent
 - The number is raised to the power 2, since we are in binary.

10010011111000000110011010010111100110011000110000110000011111000000001101001100001110

Converting From Floating Point

00100111110000011001101001011110001100001100000111110000000010100111000011000010000

- To convert 0 100000010 000111 into Decimal...
 1. Convert the exponent into its decimal form.
 - $000111 \rightarrow 4 + 2 + 1 = 7$
 2. Consider that the mantissa is preceded by a decimal point, move that decimal point exponent times to the right.
 - $100000010 \rightarrow 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0$
 3. Now convert that value into Decimal, remembering that bits after the decimal point are negative powers and therefore fractions. Don't forget the sign bit!
 - $1000000.10 \rightarrow +\ 64.5$

100100111110000011001101001011110011001100011000011000001111100000001101001100001110

Negative Floating Point

00100111110000011001101001011110001100001100000111110000000010100111000011000010000

- Where the sign bit is negative, the overall number is negative.
 - For example... 1 110000000 00001
 - = - $.110000000 \times 2^{00001}$
 - = - 1.10000000
 - = - 1.5
 - Where the exponent is negative, rather than moving the decimal point to the right, you move it to the left.
 - For example... 0 010000000 11111
 - = + $.010000000 \times 2^{11111}$
 - = + $.010000000 \times 2^{-00001}$
 - = + .0010000000
 - = + 0.125

100100111110000011001101001011110011001100011000011000001111100000001101001100001110

Normalization

00100111110000011001101001011110001100001100000111110000000010100111000011000010000

- To be memory efficient all floating point numbers should be normalised.
- A normalised number can only be displayed in one way.
 - For example in strict scientific notation 300 can only be shown as 0.3×10^3
 - However using a looser notation it could be shown as 300×10^0 , 30×10^1 , etc.
- In normalized binary, the first bit of the mantissa after the sign bit should always be a 1.

100100111110000011001101001011110011001100011000011000001111100000001101001100001110

Normalized Binary

001001111100000110011010010111100011000011000001111100000000110100111000011000010000

- To Normalize a binary number...
 1. Write the mantissa with the assumed decimal point in place.
 2. Shift the decimal point until it reaches the first 1.
 3. Subtract the number of places moved from the exponent.
- For example... Normalize 0 000110101 000010
 1. .000110101
 2. .0 0 0 1 1 0 1 0 1
 3. We moved the point 3 places, therefore we subtract 3 from the exponent of 2 ($2 - 3 = -1$)
 - 0 110101000 111111

100100111110000011001101001011110011001100011000011100000011111000000011010011100001100001110

IEEE-754

00100111110000011001101001011110001100001100001111100000000110100111000011000010000

- <http://babbage.cs.qc.edu/IEEE-754/Decimal.html>

100100111110000011001101001011110011001100011000011100000011111000000011010011100001110

CONTEXT

00100111110000011001101001011110001100001100001111100000000110100111000011000010000

- What does the value 10101000 binary represent?

100100111110000011001101001011110011001100011000011100000011111000000011010011100001110
