COMP 222
Number Systems

Convert from base 2, 8, 16 to base 10

- Each position represents a power of the base, starting at 0 on the right
- Each digit is multiplied by its power of the base
- Terms are summed to get the base 10 value

Examples:

- $10110101_2 = 1 \text{ X } 2^7 + 0 \text{ X } 2^6 + 1 \text{ X } 2^5 + 1 \text{ X } 2^4 + 0 \text{ X } 2^3 + 1 \text{ X } 2^2 + 0 \text{ X } 2^1 + 1 \text{ X } 2^0$ in base 10
- $F3EA7_{16} = 15 \text{ X } 16^4 + 3 \text{ X } 16^3 + 14 \text{ X } 16^2 + 10 \text{ X } 16^1 + 7 \text{ X } 16^0$ in base 10
- $3771_8 = 3 \text{ X } 8^3 + 7 \text{ X } 8^2 + 7 \text{ X } 8^1 + 1 \text{ X } 8^0$ in base 10

Converting Between Bases 2, 8, 16

- Since 8 and 16 are powers of 2, converting between bases 8 and 2 or 16 and 2 requires no calculation, just grouping and regrouping of base 2 bits

Example

101101001111_2 to base 8
Each group of 3 binary digits forms 1 octal digit
Must group starting on the **right**
Pad with 0s if necessary on left
$(101)(101)(001)(111) = 5517_8$

- What about conversion from base 2 to base 16?
  - Use groups of 4 bits
- What about conversion between base 8 and base 16?
  - First convert from original base to base 2, then convert to final base.

Converting a Long Binary String to Decimal

When a binary number has long unbroken strings of 0s and 1s, there is a trick that might simplify the calculation.

- Simple version of the trick: long string of 1s = next power of 2 – 1.
- Example:
  - $11111 = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 16 + 8 + 4 + 2 + 1 = 31$
  - Note that the next power of 2 is 2^5 so value = $2^5 – 1$

More general version of the shortcut

- What if the binary digit contains a mix of strings of 0s and strings of 1s?
- The more general version does computations based on the number of transitions from 0 to 1 or 1 to 0.

- Take the string to be converted and pad with 0s left and right
  - 0 11111 0
- Read digits from **right to left**
  - 0 to 1 transition: **subtract** next power of 2
  - 1 to 0 transition: **add** next power of 2
- Value = (0 to 1 transition) $– 2^0$ + (1 to 0 transition) $2^5 = – 1 + 32 = 31$

Exercise

- 1110000111100011
- Brute force solution: $2^{15} + 2^{14} + 2^{13} + 2^8 + 2^7 + 2^6 + 2^5 + 2^1 + 2^0$
- Shortcut: 0 1110000111100011 0

Coming Up

- Booth's Algorithm for Binary Multiplication
- Example of an algorithm commonly implemented in hardware
- Algorithm is easier to understand if you know this shortcut first

Converting From Base 10 to Another Base (2, 8, 16)

Algorithms require more calculation, although they are still simple.
Commonly makes use of integer divide and integer remainder (modulo) ops.
Note the following tricks with div and mod.
In base 10, div and mod can be used to strip away and/or isolate specific digits.
Example:

15346 / 10 = 1534 (deletes rightmost digit)
15346 % 10 = 6 (isolates rightmost digit)
15346 / 100 = 153 (delete rightmost 2 digits)
15346 % 100 = 46 (isolates rightmost 2 digits)

As a warm up, let's convert 56197 from base 10 to base 10 (no change) to demo the steps with div and mod:

56197 % 10 = 7	56197 / 10 = 5619	
5619 % 10 = 9	5619 / 10 = 561	
561 % 10 = 1	561 / 10 = 56	
56 % 10 = 6	56 / 10 = 5	
5 % 10 = 5	5 / 10 = 0	← done

- At each step we compute one digit of the result starting with the rightmost digit, working our way left.
- To advance the calculation to the next step, we chop off one digit on the right and continue with the remaining digits.
- When the remaining value reaches 0, the conversion is complete

Now let's convert it to base 2:

56197 % 2 = 1	56197 / 2 = 28098
28098 % 2 = 0	28098 / 2 = 14049
14049 % 2 = 1	14049 / 2 = 7024
7024 % 2 = 0	7024 / 2 = 3512
3512 % 2 = 0	3512 / 2 = 1756
1756 % 2 = 0	1756 / 2 = 878
878 % 2 = 0	878 / 2 = 439
439 % 2 = 1	439 / 2 = 219
219 % 2 = 1	219 / 2 = 109
109 % 2 = 1	109 / 2 = 54
54 % 2 = 0	54 / 2 = 27
27 % 2 = 1	27 / 2 = 13
13 % 2 = 1	13 / 2 = 6
6 % 2 = 0	6 / 2 = 3
3 % 2 = 1	3 / 2 = 1
1 % 2 = 1	1 / 2 = 0 ← done

Reassemble the digits (topmost is rightmost0 to get 1101101110000101

Check:

$2^{15} + 2^{14} + 2^{12} + 2^{11} + 2^9 + 2^8 + 2^7 + 2^2 + 2^0$
$= 32768 + 16384 + 4096 + 2048 + 512 + 256 + 128 + 2 + 1 = 56197$

Exercise: convert 56197 to base 16

There are other conversion algorithms. One alternative is to produce the resulting bits left to right. This algorithm requires you to first calculate the largest power of the base that is not larger than the number being converted.