Decimal Representation

- Can interpret decimal number 4705 as:
  
  \[ 4 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 5 \times 10^0 \]

- The *base* or *radix* is 10
  - Digits 0 – 9

- Place values:
  
  \[
  \begin{array}{cccc}
  & 1000 & 100 & 10 & 1 \\
  \cdots & \cdots & \cdots & \cdots & \cdots \\
  & 10^3 & 10^2 & 10^1 & 10^0 \\
  \end{array}
  \]

- Write number as 4705_{10}
  - Note use of subscript to denote base
In a similar way, can interpret binary number 1011 as:

\[ 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \]

The *base* or *radix* is 2

Digits 0 and 1

Place values:

\[ \cdots \quad 8 \quad 4 \quad 2 \quad 1 \]

\[ \cdots \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \]

Write number as 1011\textsubscript{2}

\((= 11\textsubscript{10})\)
Hexadecimal Representation

- Can interpret hexadecimal number 3AF1 as:
  \[3 \times 16^3 + 10 \times 16^2 + 15 \times 16^1 + 1 \times 16^0\]
- The base or radix is 16
  Digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Place values:
  \[
  \begin{array}{cccccc}
  \ldots & 4096 & 256 & 16 & 1 \\
  \ldots & 16^3 & 16^2 & 16^1 & 16^0 \\
  \end{array}
  \]
- Write number as \(3AF1_{16}\)
  \((= 15089_{10})\)
Binary to Hexadecimal

- **Idea**: Collect bits into groups of four starting from right to left
- “pad” out left-hand side with 0’s if necessary
- Convert each group of four bits into its equivalent hexadecimal representation (given in table above)
Binary to Hexadecimal

- Example: Convert 1011111000101001\textsubscript{2} to Hex:

<table>
<thead>
<tr>
<th>1011</th>
<th>1110</th>
<th>0010</th>
<th>1001\textsubscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>E</td>
<td>2</td>
<td>9\textsubscript{16}</td>
</tr>
</tbody>
</table>

- Example: Convert 10111101011100\textsubscript{2} to Hex:

<table>
<thead>
<tr>
<th>0010</th>
<th>1111</th>
<th>0101</th>
<th>1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>F</td>
<td>5</td>
<td>C\textsubscript{16}</td>
</tr>
</tbody>
</table>
Hexadecimal to Binary

- Reverse the previous process
- Convert each hex digit into equivalent 4-bit binary representation
- Example: Convert $\text{AD}5_{16}$ to Binary:

<table>
<thead>
<tr>
<th>A</th>
<th>D</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>1101</td>
<td>0101$_2$</td>
</tr>
</tbody>
</table>
Values that we normally treat as atomic can be viewed as bits, e.g.

- **char** = 1 byte = 8 bits  ('a' is 01100001)
- **short** = 2 bytes = 16 bits  (42 is 0000000000101010)
- **int** = 4 bytes = 32 bits  (42 is 0000000000000000...0000101010)
- **double** = 8 bytes = 64 bits

The above are common sizes and don’t apply on all hardware e.g. `sizeof(int)` might be 2, 4 or 8.

C provides a set of operators that act bit-by-bit on pairs of bytes. E.g. \((10101010 \& 11110000) == 10100000\) (bitwise AND)

C bitwise operators: \& | ~ << >>
Binary Constants

Literal numbers in decimal, hexadecimal, octal, binary.
In hexadecimal, each digit represents 4 bits

<table>
<thead>
<tr>
<th>0100</th>
<th>1000</th>
<th>1111</th>
<th>1010</th>
<th>1011</th>
<th>1100</th>
<th>1001</th>
<th>0111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x</td>
<td>4</td>
<td>8</td>
<td>F</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>9</td>
</tr>
</tbody>
</table>

In octal, each digit represents 3 bits

<table>
<thead>
<tr>
<th>01</th>
<th>001</th>
<th>000</th>
<th>111</th>
<th>110</th>
<th>101</th>
<th>011</th>
<th>110</th>
<th>010</th>
<th>010</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

In binary, each digit represents 1 bit

```
0b01001000111110101011110010010111
```