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:
  \[ \cdots 1000 \ 100 \ 10 \ 1 \]
  \[ \cdots 10^3 \ 10^2 \ 10^1 \ 10^0 \]
- Write number as \(4705_{10}\)
  - Note use of subscript to denote base

Binary Representation

- 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:
  \[ \cdots 8 \ 4 \ 2 \ 1 \]
  \[ \cdots 2^3 \ 2^2 \ 2^1 \ 2^0 \]
- Write number as \(1011_2\)
  \((= 11_{10})\)

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:
  \[ \cdots 4096 \ 256 \ 16 \ 1 \]
  \[ \cdots 16^3 \ 16^2 \ 16^1 \ 16^0 \]
- 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 $1011111000101001_2$ to Hex:
  
  \[
  \begin{array}{cccc}
  1011 & 1110 & 0010 & 1001 \\
  B & E & 2 & 9_{16}
  \end{array}
  \]

- Example: Convert $10111101011100_2$ to Hex:
  
  \[
  \begin{array}{cccc}
  0010 & 1111 & 0101 & 1100 \\
  2 & F & 5 & C_{16}
  \end{array}
  \]

Hexadecimal to Binary

- Reverse the previous process
- Convert each hex digit into equivalent 4-bit binary representation
- Example: Convert $AD5_{16}$ to Binary:
  
  \[
  \begin{array}{cccc}
  A & D & 5 \\
  1010 & 1101 & 0101
  \end{array}
  \]

Bits in Bytes in Words

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...000101010)
- 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

\[
\begin{array}{cccccccccccc}
0100 & 1000 & 1111 & 1010 & 1011 & 1100 & 1001 & 0111 \\
0x & 4 & 8 & F & A & B & C & 9 & 7
\end{array}
\]

In octal, each digit represents 3 bits

\[
\begin{array}{cccccccccc}
01 & 001 & 000 & 111 & 110 & 101 & 011 & 110 & 010 & 010 & 111 \\
0 & 1 & 1 & 0 & 7 & 6 & 5 & 3 & 6 & 2 & 2 & 7
\end{array}
\]

In binary, each digit represents 1 bit

\[
\begin{array}{ccccccc}
0b01000100011110110111100100010111
\end{array}
\]