Number Systems

1 Decimal

The ”decimal point” in general is called the **radix point**, this indicates the position of the ”units”, immediately to the left of the radix point

2 Positional number systems

The **base**(or **radix**) of the number system is the number of symbols (including 0).
The subscript after the number indicates the base

The contribution of symbol x, which is the i^{th} symbol in the order, is the

$$(i-1) \times base^{position}$$

Where position is the number of places to the **left** of the units

Position	2	1	0	.	-1	-2
$Base^{Position}$	2^2	2^1	2^0	.	2^{-1}	2^{-2}
Decimal Value	4	2	1	.	.5	.25
Example	1	1	0	.	1	1

For this example it is equivalent to the decimal 6.75

3 Binary

Each digit in a binary number system is known as a bit

- **B**inary dig**IT**

A bit can have only one of two possible values

- 0 or 1 (false/true, off/on, LOW/HIGH)

Groups of bits are known as:

- **Nibble** - 4 bits
- **Byte** - 8 bits
- **Half Word** - 16 bits
- **Word** - 32 bits
- **Double Word** - 64 bits

Note that the value of a ”word” is CPU dependent, as for a 64 bit CPU, a word is 64 bits, rather than the 32 stated above

4 Hexadecimal

This has 16 distinct symbols: **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**

Why do we need Hexadecimal?

- Reading and writing binary values is difficult for humans

Advantages to using Hexadecimal

- **More Compact** than other number systems
- **Easy to convert** between binary and decimal

Programmers must be aware of what they are writing

- BEEF and BEEF_{16} have very different meanings
- In Java use a prefix to denote a hexadecimal value: $0\text{xBEEF}=\text{BEEF}_{16}$

4.1 Hexadecimal conversion

Position	2	1	0	.	-1	-2
$Base^{Position}$	16^2	16^1	16^0	.	16^{-1}	16^{-2}
Decimal Value	256	16	1	.	.0625	.00390625
Example	C	2	D	.	1	0

For this example it is equal to 3117.0625 in decimal

5 Conversion

5.1 Binary to Hex

1. Starting from the **radix point**, separate the binary number into groups of **four** binary digits (nibbles)
2. Then **translate each group** (nibble) into its hexadecimal equivalent, group by group, maintaining left to right order

Remember to add zeroes on the start and end to ensure that you get full nibbles if needed

0011	0101	1101	1000	.	0010
3	5	D	8	.	2

5.2 Hex to binary

1. Starting from the **radix point**, separate the hexadecimal number into digits
2. Then translate each digit into a **4-digit binary nibble**, maintaining right to left order