CSC180: Lecture 36

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Low-level Programming

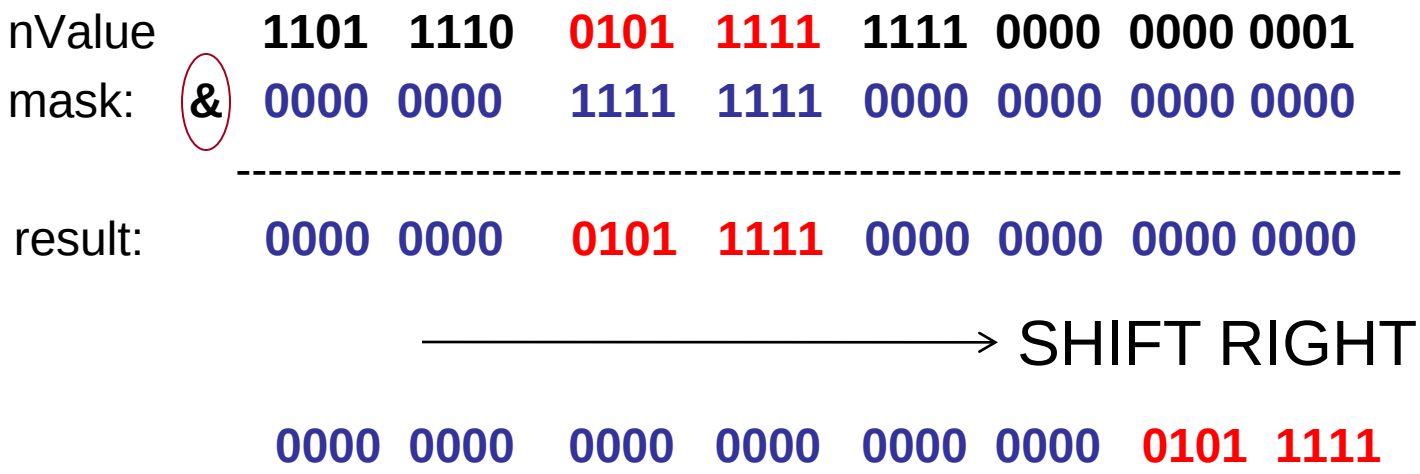
Data Compression – pack

- Example:

```
int pack4char (char char1, char char2,
                        char char3, char char4)
{
        int a=char1;
        a =  (a<<8)|char2;
        a  = (a<<8)|char3;
        a  = (a<<8)|char4;
        return a;
   }
```

Data unpacking: using a mask

- Constant or variable to extract desired bits
  - Recall: ANDing any bit with 0 yields 0, and ANDing any bit with 1 yields the original bit
  - E.g.



Multiplication using left shifting

- you can multiply by *any* power of two by shifting left the same number of places.

- E.g. if you wanted to multiply a number by 16, which is 2^4, you could simply shift it left four places.
  - x = y * 8; x = y << 3;
  - x = y * 64; x = y << 6;
  - x = y * 32768; x = y << 15;

Multiplication using left shifting

- How come?

0010 1101 = (**1** * 2^5) + (**1** * 2^3) + (**1** * 2^2) + (**1** * 2^0)

0010 1101 * 2 = 2(**1** * 2^5) + 2(**1** * 2^3) + 2(**1** * 2^2) + 2(**1** * 2^0)

0010 1101 * 2 = (**1** * 2^6) + (**1** * 2^4) + (**1** * 2^3) + (**1** * 2^1)

= **0101 1010**

0010 1101

↙ ↙↙↙ ↙↙ ↙

0101 1010

Division using right shifting

- you can divide by *any* power of two by shifting right the same number of places.

- E.g. if you wanted to divide a number by 4, which is 2^2, you could simply right shift two places.
  - x = y / 4; x = y >> 2;
  - x = y / 32; x = y >> 5;

Number Systems and C Programming

- You can’t specify a binary number in C ☹
- You need to use base10, base8 or base16, encoding.

Number Systems

System	Base	Symbols
Decimal	10	0, 1, … 9
Binary	2	0, 1
Octal	8	0, 1, … 7
Hexa-decimal	16	0, 1, … 9, A, B, … F

Quantities/Counting (1 of 3)

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Quantities/Counting (2 of 3)

Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

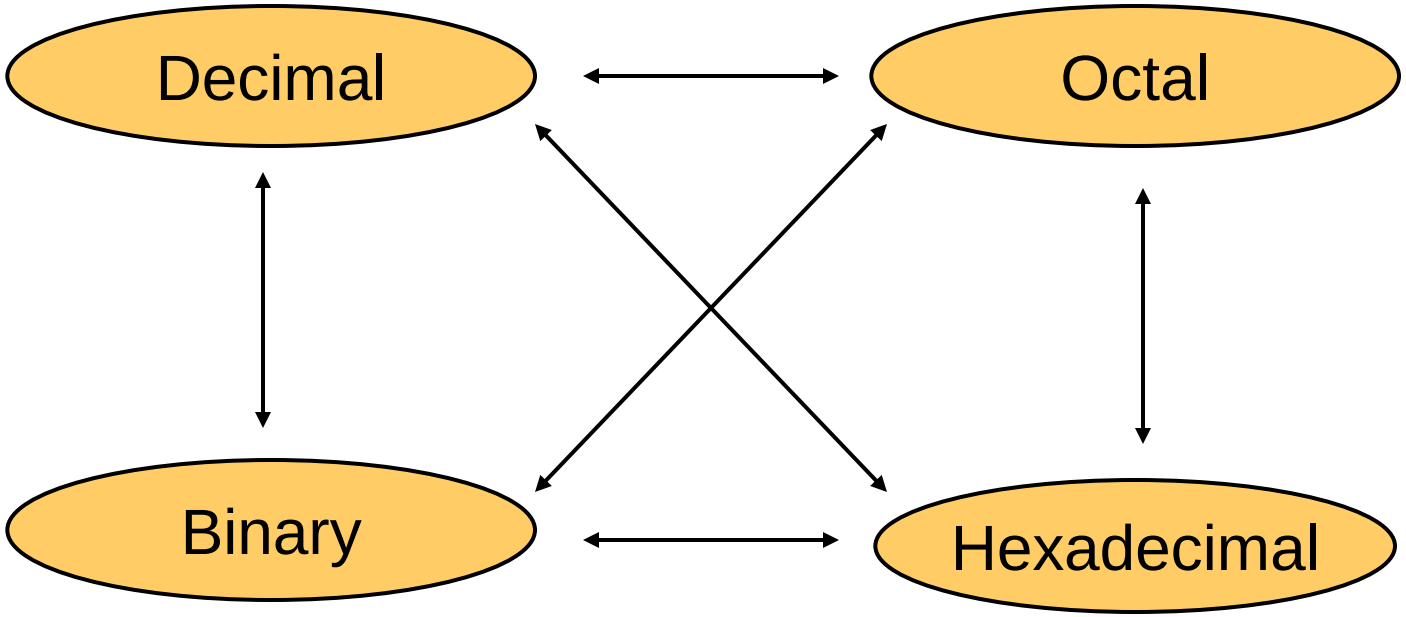
Quantities/Counting (3 of 3)

Decimal	Binary	Octal	Hexa-decimal
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

Etc.

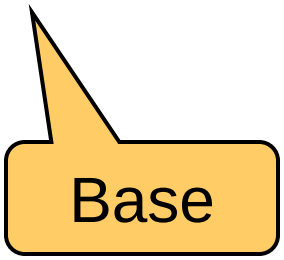
Conversion Among Bases

- The possibilities:

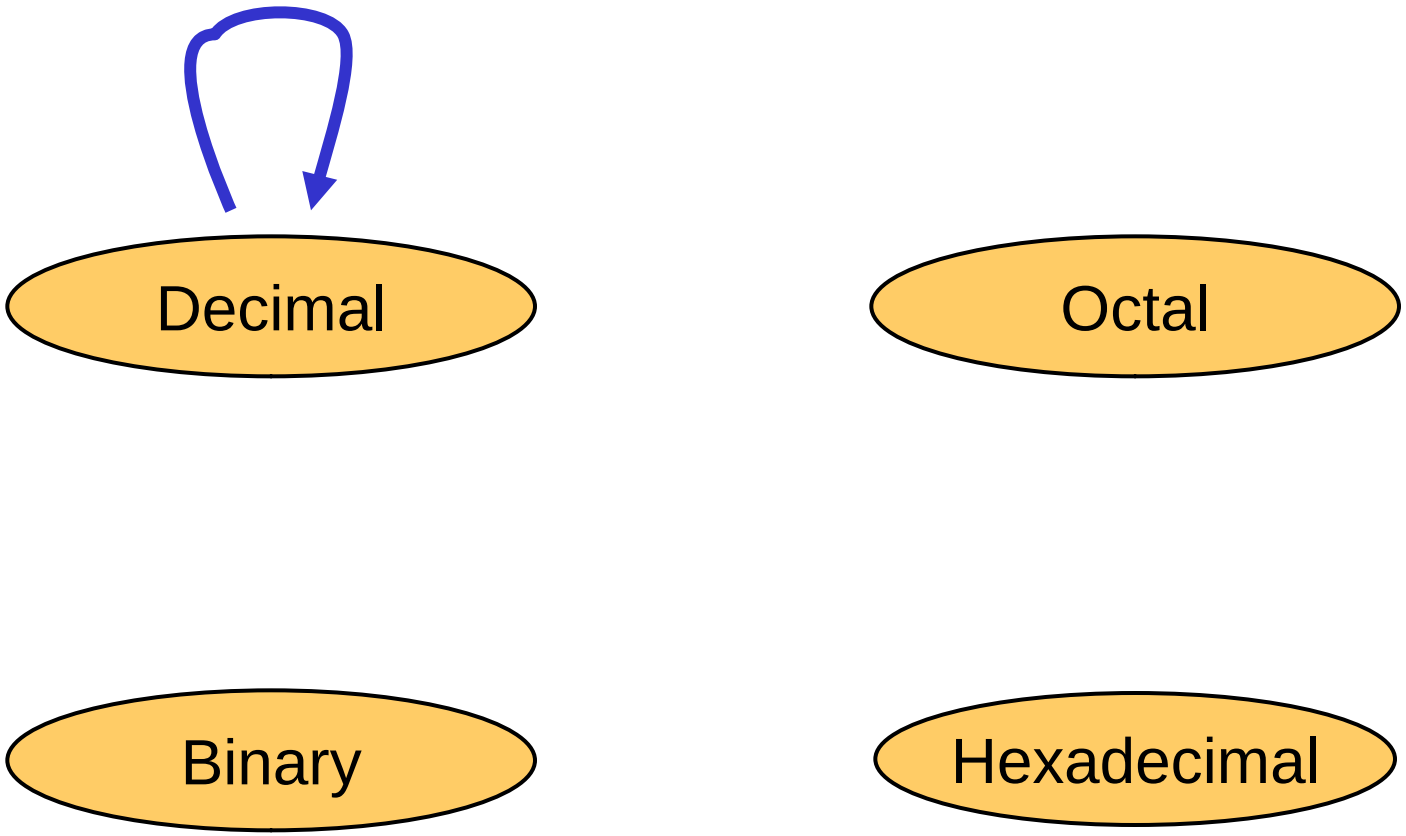


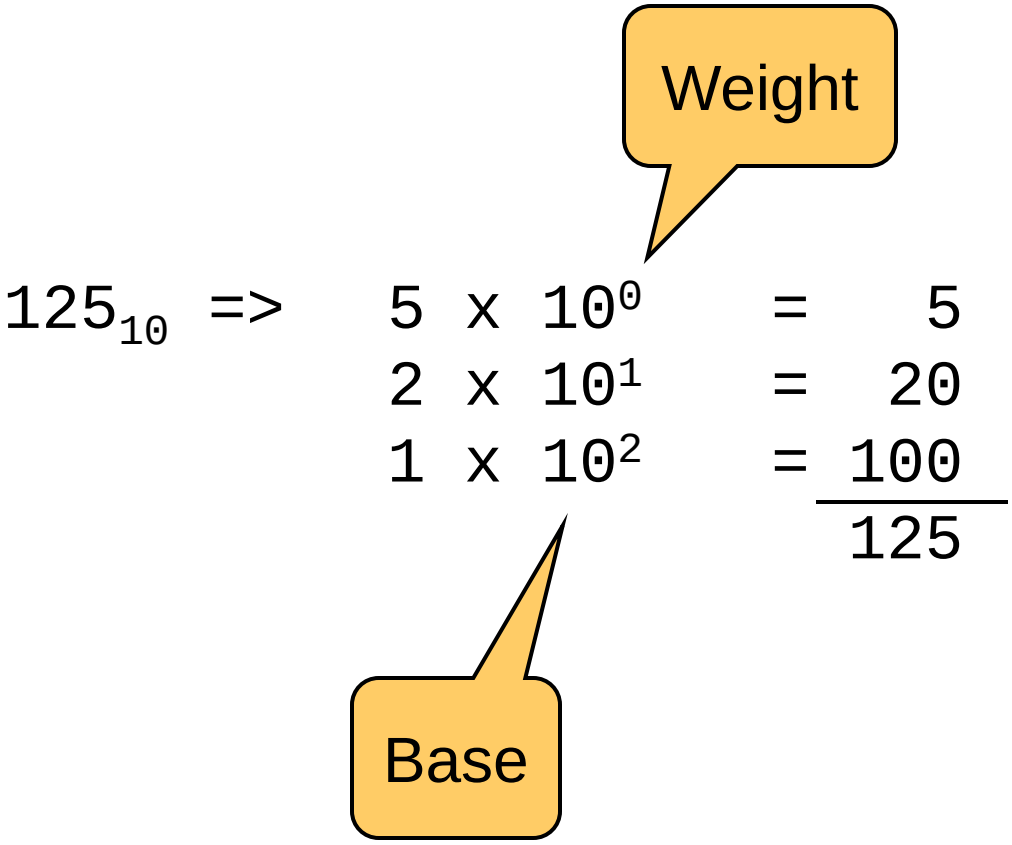
Quick Example

$$25_{10} = 11001_{2} = 31_{8} = 19_{16}$$

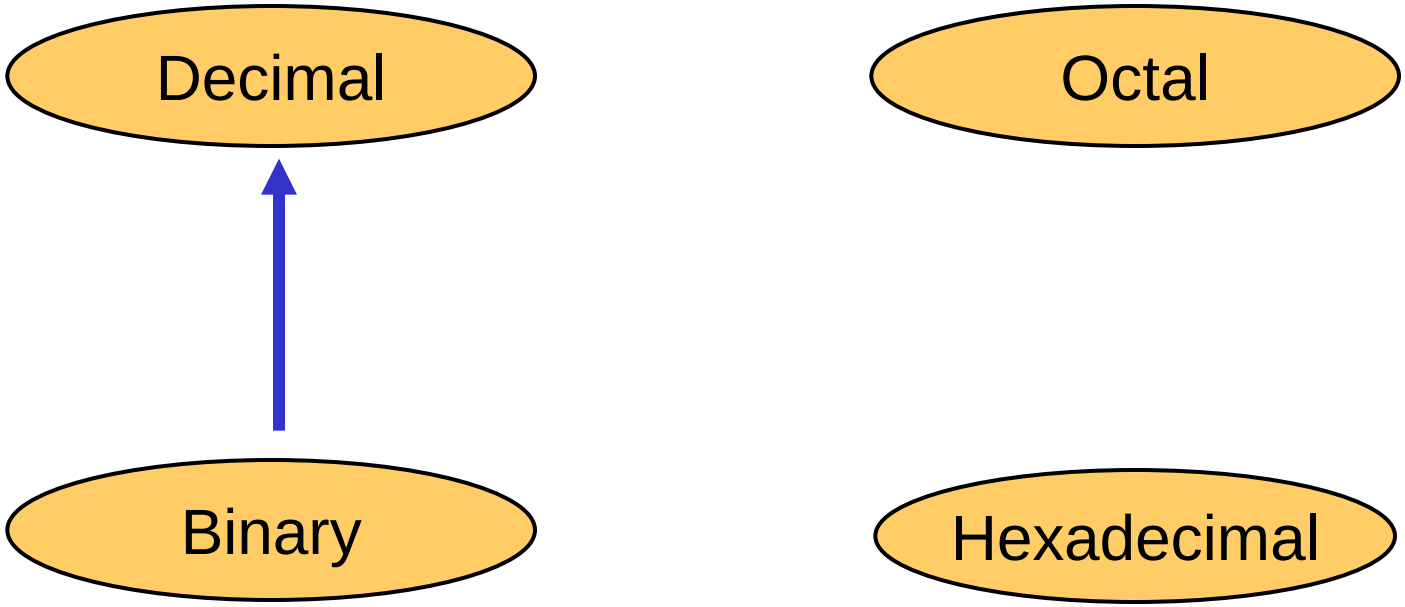


Decimal to Decimal (just for fun)




Weight
125_{10} =>
5 x 10^0 = 5
2 x 10^1 = 20
1 x 10^2 = 100
125
Base

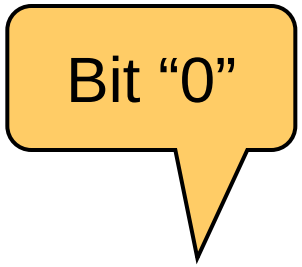
Binary to Decimal



Binary to Decimal

- Technique
  - Multiply each bit by 2^n, where n is the “weight” of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example



101011_2 =>

$1 \times 2^0 = 1$

$1 \times 2^1 = 2$

$0 \times 2^2 = 0$

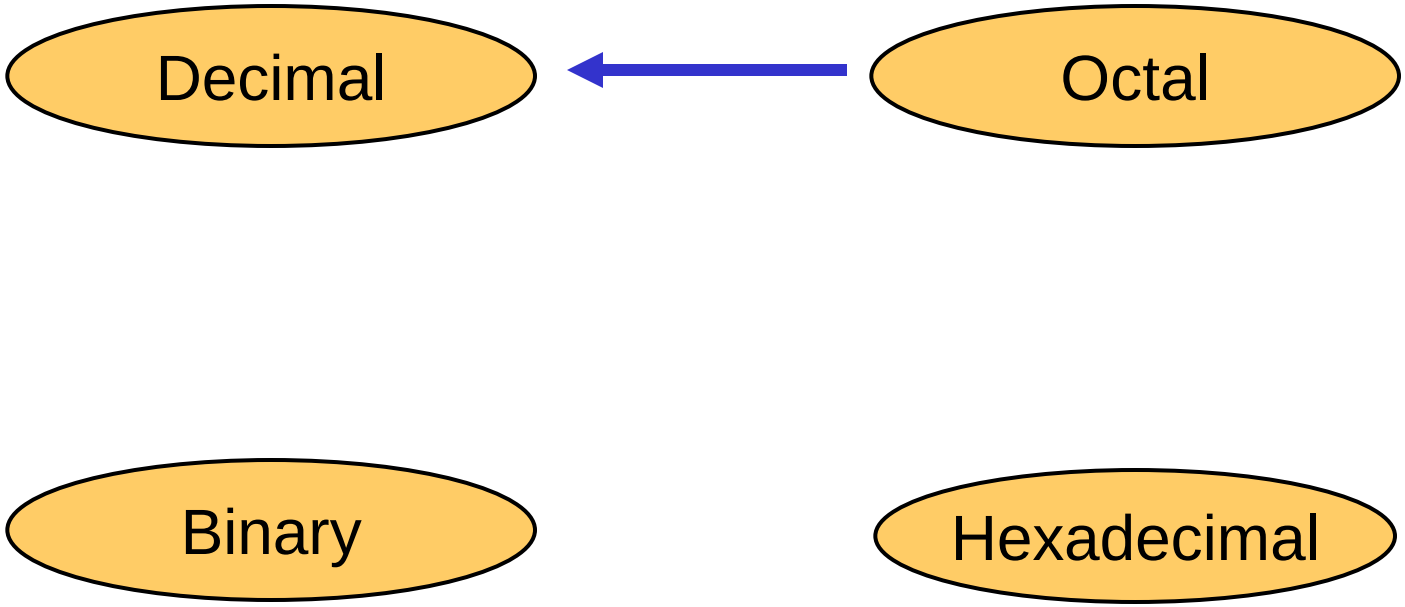
$1 \times 2^3 = 8$

$0 \times 2^4 = 0$

$1 \times 2^5 = 32$

43_{10}

Octal to Decimal



Octal to Decimal

- Technique
  - Multiply each bit by 8^n, where *n* is the “weight” of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

724_8 =>

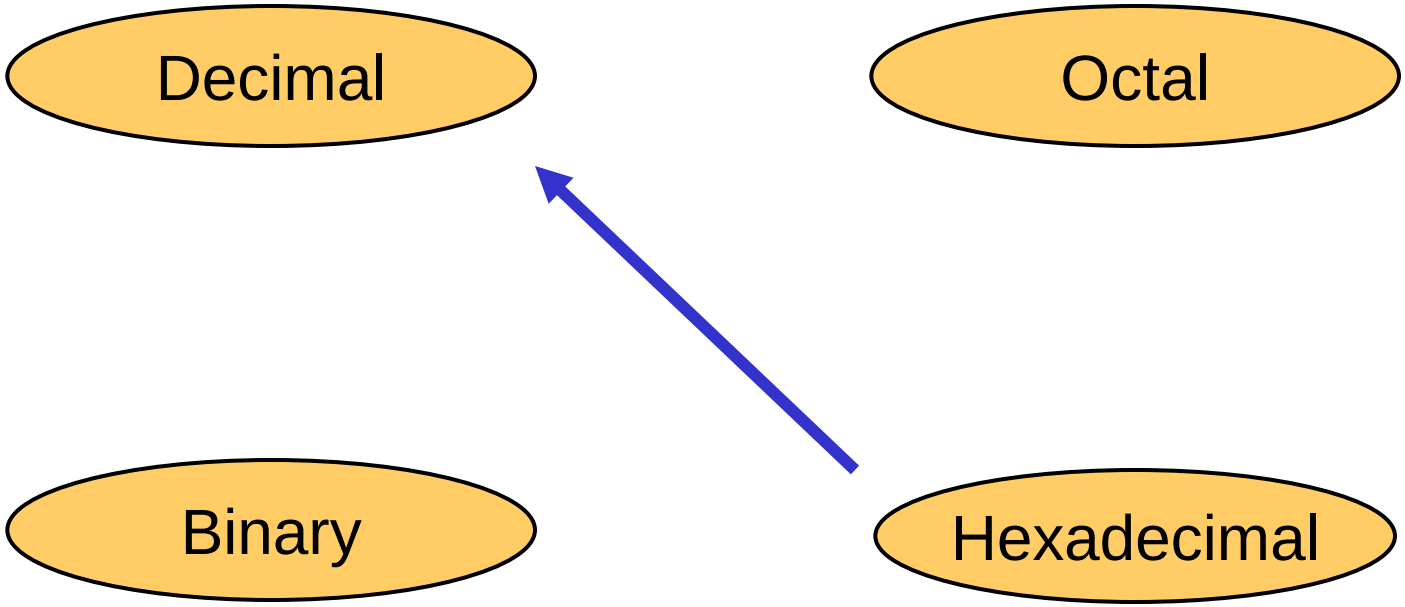
$4 \times 8^0 = 4$

$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

468_{10}

Hexadecimal to Decimal



Hexadecimal to Decimal

- Technique
  - Multiply each bit by 16^n, where *n* is the “weight” of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

ABC_{16} =>

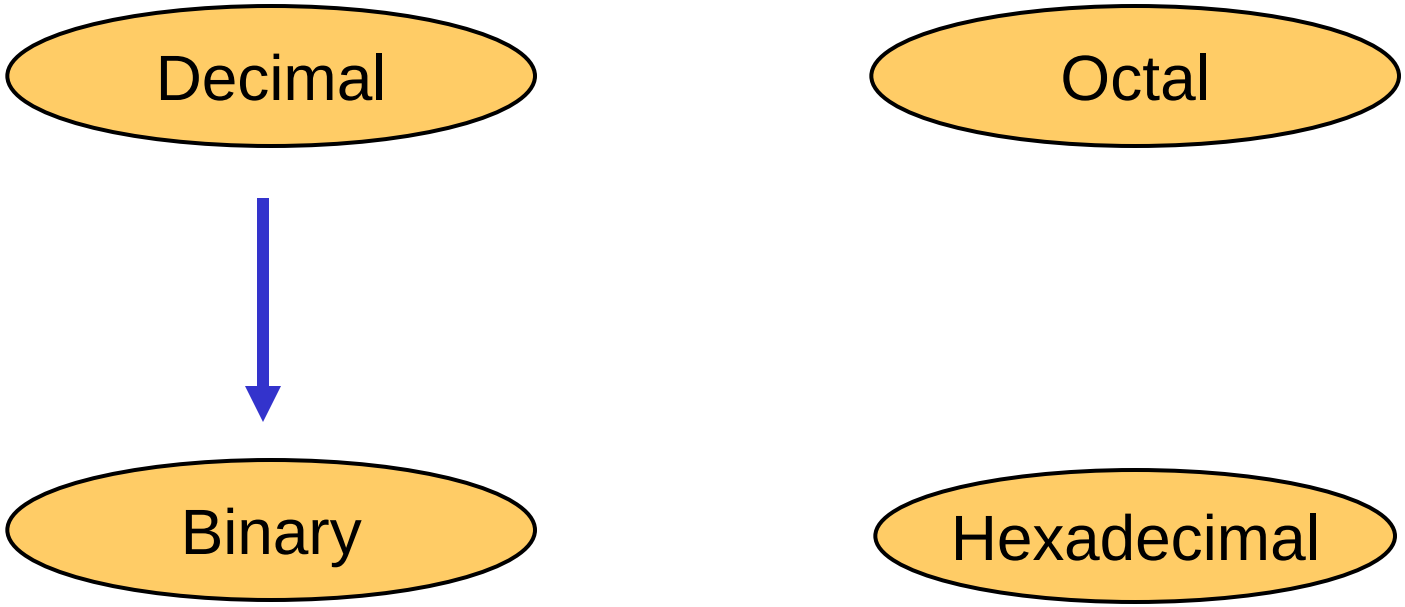
$C \times 16^0 = 12 \times 1 = 12$

$B \times 16^1 = 11 \times 16 = 176$

$A \times 16^2 = 10 \times 256 = 2560$

2748_{10}

Decimal to Binary



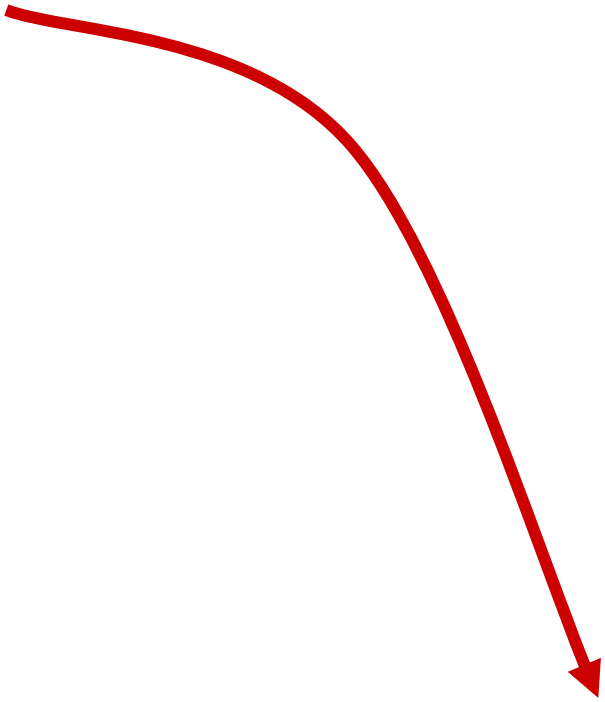
Decimal to Binary

- Technique
  - Divide by two, keep track of the remainder
  - First remainder is bit 0 (LSB, least-significant bit)
  - Second remainder is bit 1
  - Etc.

Example

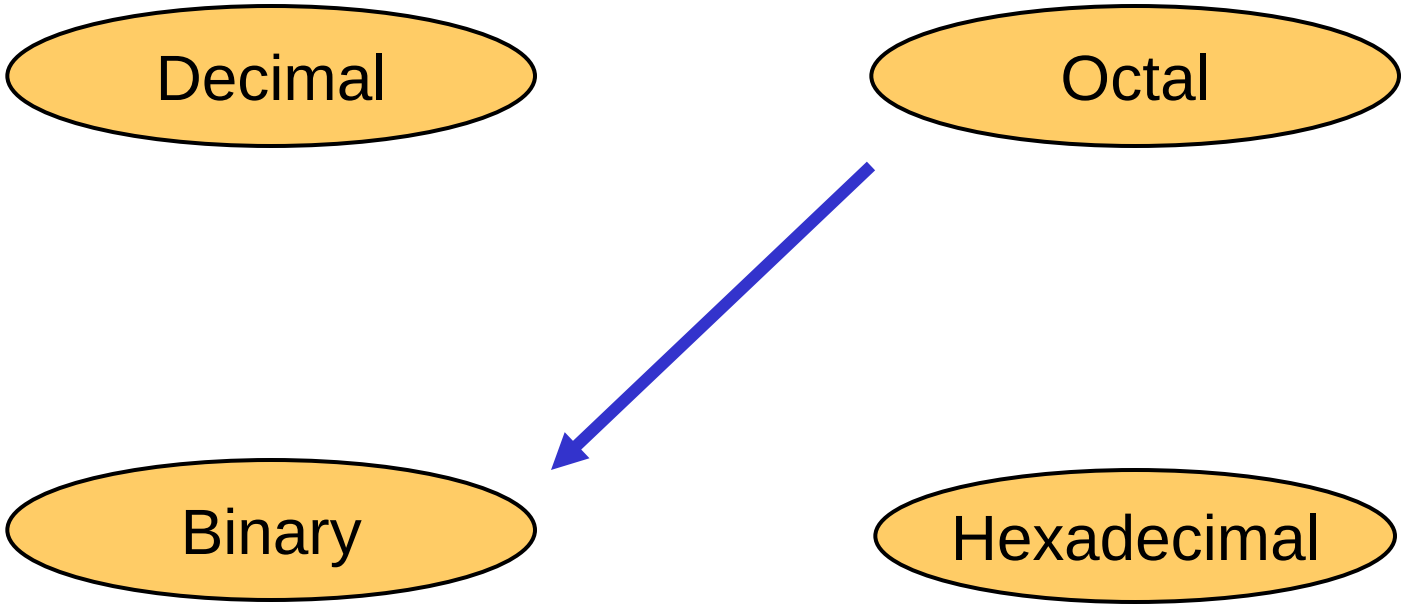
$125_{10} = ?_2$

2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1



$125_{10} = 1111101_2$

Octal to Binary

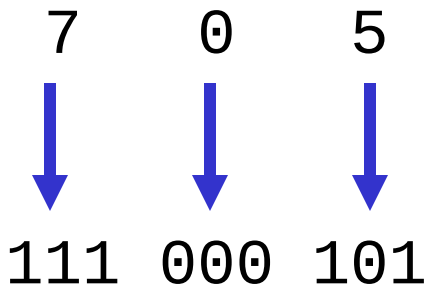


Octal to Binary

- Technique
  - Convert each octal digit to a 3-bit equivalent binary representation

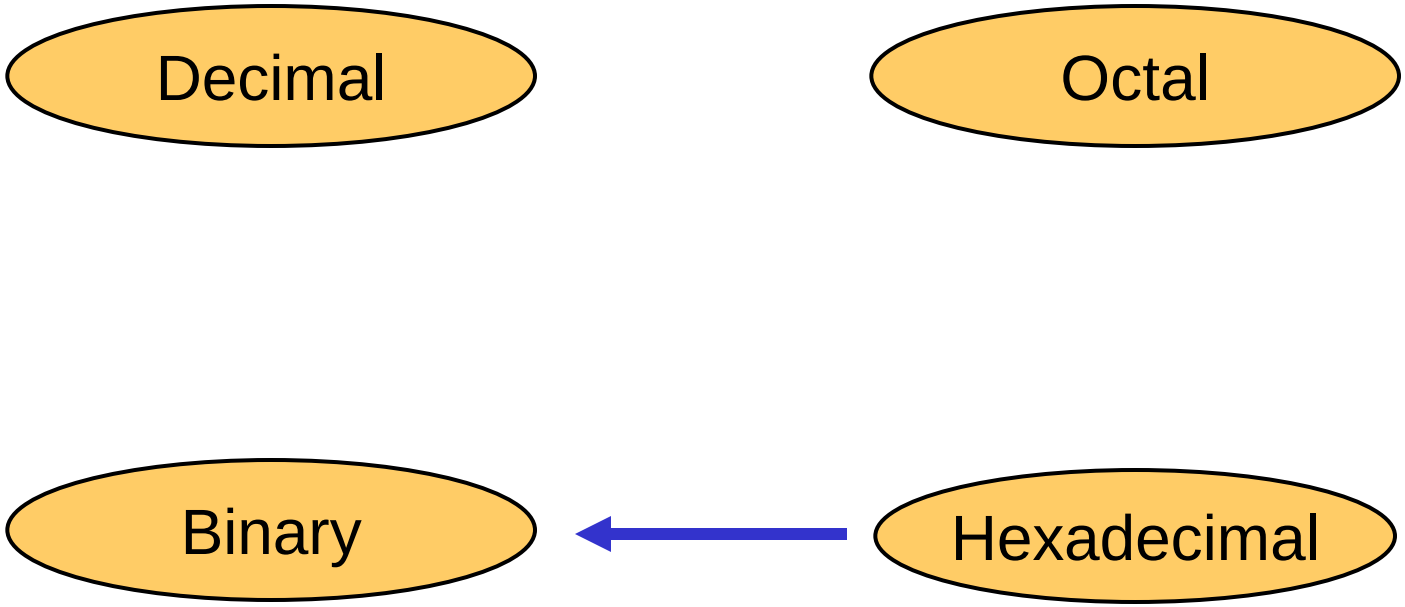
Example

$705_8 = ?_2$



$705_8 = 111000101_2$

Hexadecimal to Binary

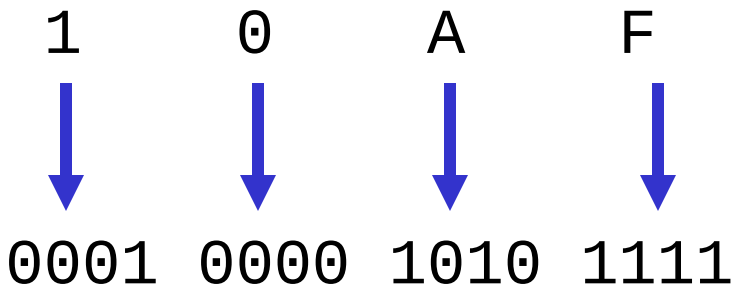


Hexadecimal to Binary

- Technique
  - Convert each hexadecimal digit to a 4-bit equivalent binary representation

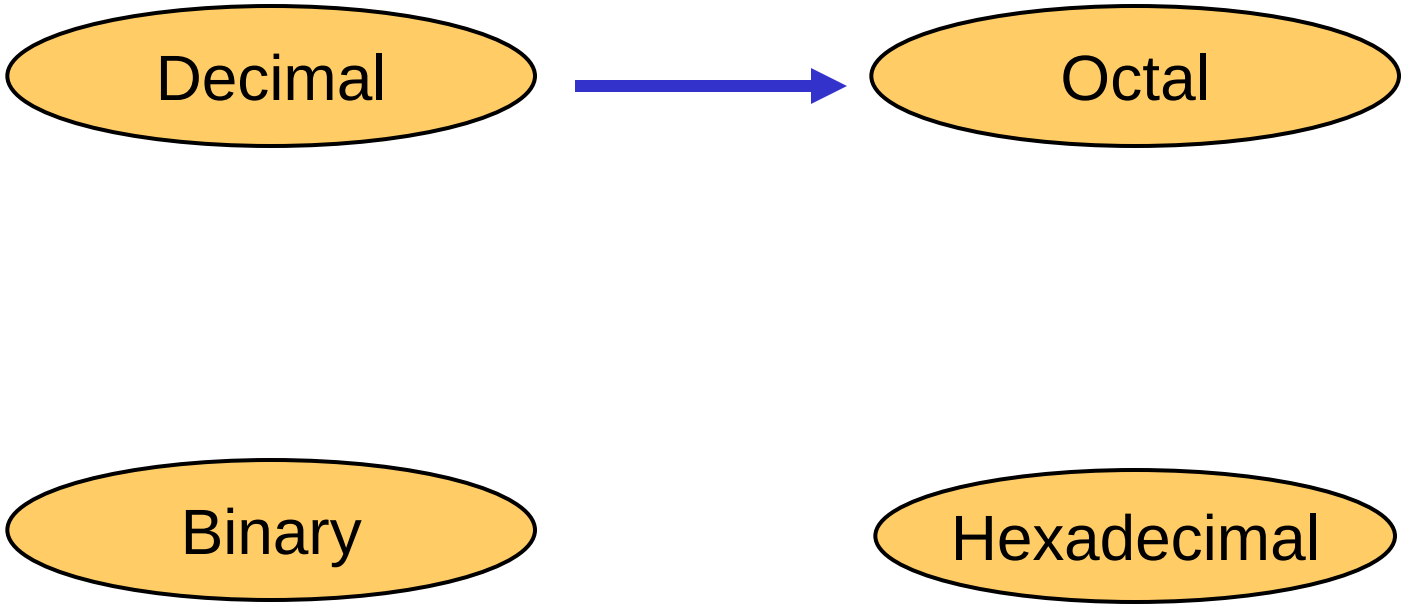
Example

$10AF_{16} = ?_2$



$10AF_{16} = 0001000010101111_2$

Decimal to Octal



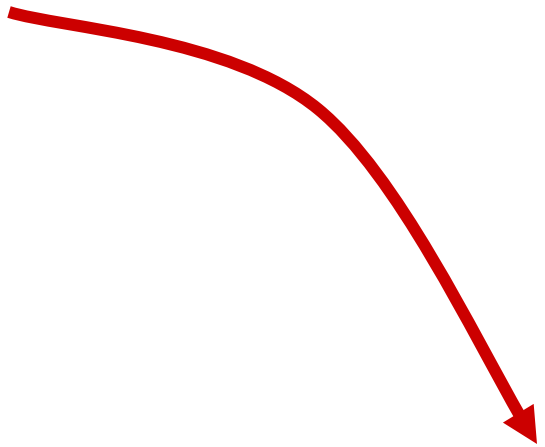
Decimal to Octal

- Technique
  - Divide by 8
  - Keep track of the remainder

Example

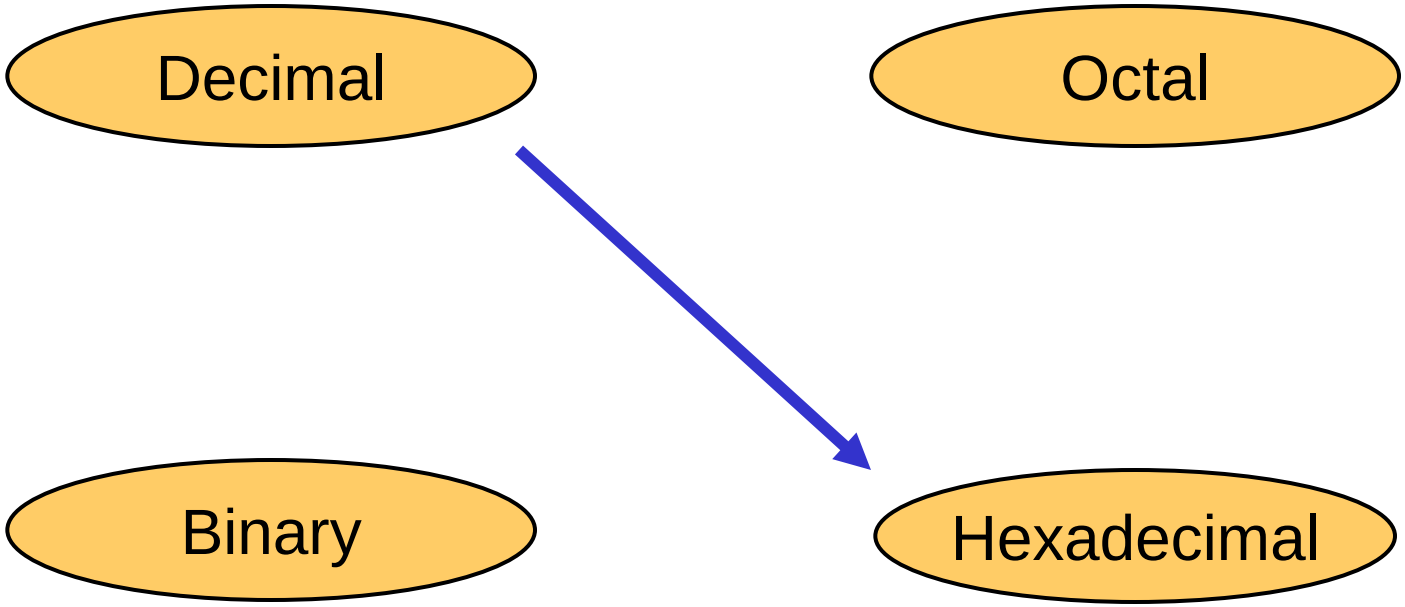
$1234_{10} = ?_{8}$

8	1234	
8	154	2
8	19	2
8	2	3
	0	2



$1234_{10} = 2322_{8}$

Decimal to Hexadecimal



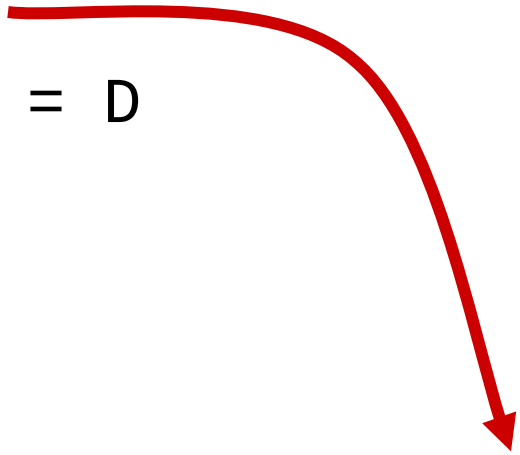
Decimal to Hexadecimal

- Technique
  - Divide by 16
  - Keep track of the remainder

Example

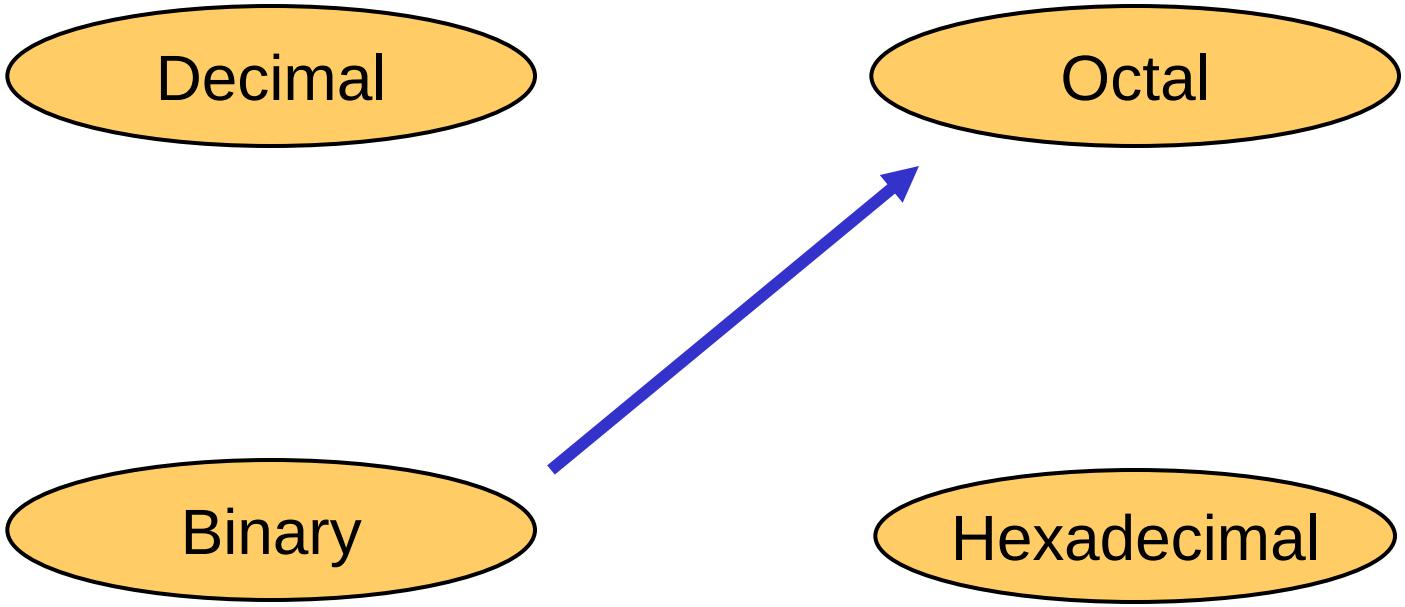
$1234_{10} = ?_{16}$

16	1234	
16	77	2
16	4	13 = D
	0	4



$1234_{10} = 4D2_{16}$

Binary to Octal

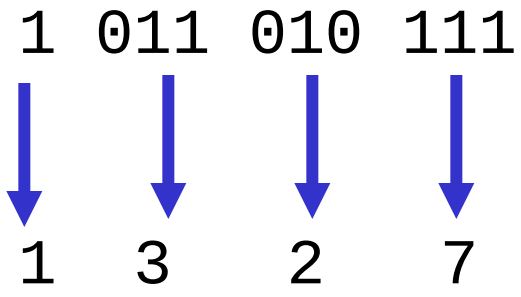


Binary to Octal

- Technique
  - Group bits in threes, starting on right
  - Convert to octal digits

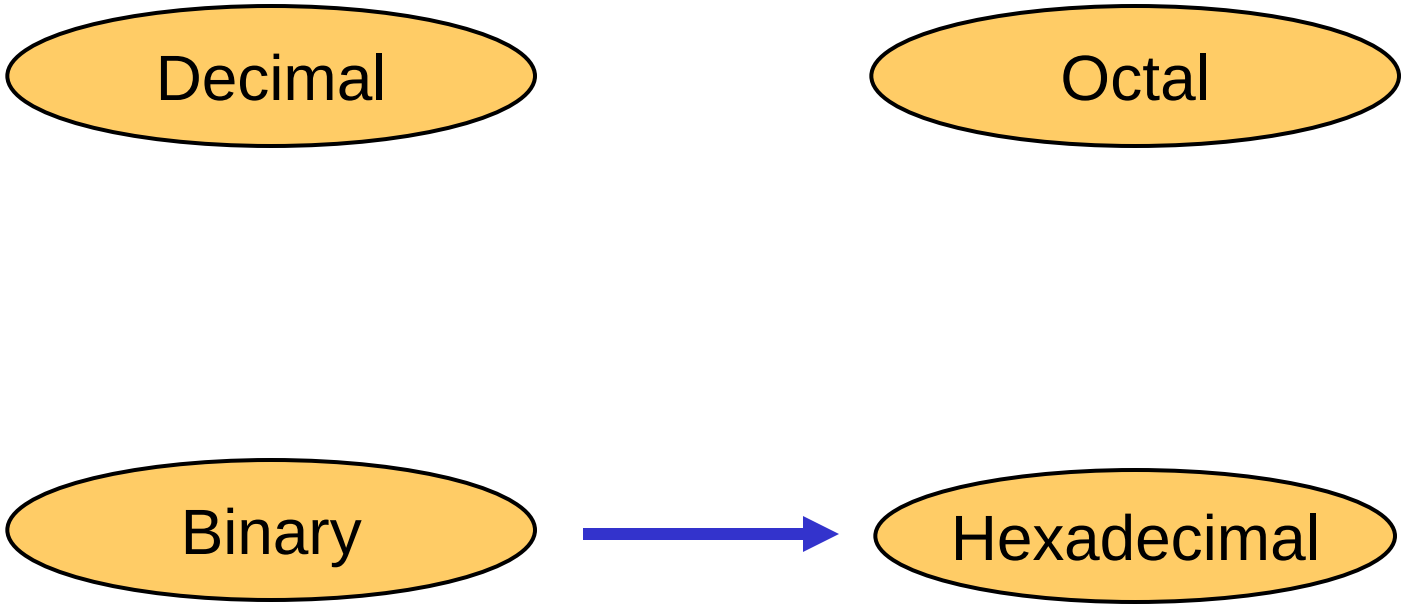
Example

$1011010111_2 = ?_8$



$1011010111_2 = 1327_8$

Binary to Hexadecimal



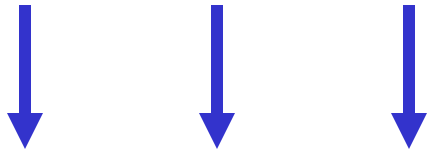
Binary to Hexadecimal

- Technique
  - Group bits in fours, starting on right
  - Convert to hexadecimal digits

Example

$1010111011_2 = ?_{16}$

10 1011 1011



2 B B

$1010111011_2 = 2BB_{16}$