CSE1003-Digital Logic Design

Dr.Penchalaiah Palla

Dept. of Micro and Nanoelectronics

School of Electronics Engineering,VIT, Vellore

CSE1003	DIGITAL LOGIC AND DESIGN	L	T	P	J	C
		3	0	2	0	4
Pre-requisite	NIL	Syllabus version				
		v1.0				

Course Objectives:

1. Introduce the concept of digital and binary systems.
2. Analyze and Design combinational and sequential logic circuits.
3. Reinforce theory and techniques taught in the classroom through experiments in the laboratory.

Expected Course Outcome:

1. Comprehend the different types of number system.
2. Evaluate and simplify logic functions using Boolean Algebra and K-map.
3. Design minimal combinational logic circuits.
4. Analyze the operation of medium complexity standard combinational circuits like the encoder, decoder, multiplexer, demultiplexer.
5. Analyze and Design the Basic Sequential Logic Circuits
6. Outline the construction of Basic Arithmetic and Logic Circuits
7. Acquire design thinking capability, ability to design a component with realistic constraints, to solve real world engineering problems and analyze the results.

Module:1	**INTRODUCTION**	**3 hours**
Number System - Base Conversion - Binary Codes - Complements(Binary and Decimal)		
Module:2	**BOOLEAN ALGEBRA**	**8 hours**
Boolean algebra - Properties of Boolean algebra - Boolean functions - Canonical and Standard forms - Logic gates - Universal gates – Karnaugh map - Don"t care conditions - Tabulation Method		
Module:3	**COMBINATIONAL CIRCUIT - I**	**4 hours**
Adder - Subtractor - Code Converter - Analyzing a Combinational Circuit		
Module:4	**COMBINATIONAL CIRCUIT –II**	**6 hours**
Binary Parallel Adder- Look ahead carry - Magnitude Comparator - Decoders – Encoders - Multiplexers –Demultiplexers.		

Module:5	**SEQUENTIAL CIRCUITS – I**	**6 hours**
Flip Flops - Sequential Circuit: Design and Analysis - Finite State Machine: Moore and Mealy model - Sequence Detector.		
Module:6	**SEQUENTIAL CIRCUITS – II**	**7 hours**
Registers - Shift Registers - Counters – Ripple and Synchronous Counters – Modulo counters – Ring and Johnson counters		
Module:7	**ARITHMETIC LOGIC UNIT**	**9 hours**
Bus Organization - ALU - Design of ALU - Status Register – Design of Shifter - Processor Unit – Design of specific Arithmetic Circuits Accumulator - Design of Accumulator.		
Module:8	**Contemporary Issues: RECENT TRENDS**	**2 hours**
	Total Lecture hours:	**45 hours**

Text Book(s)	
1.	M. Morris Mano and Michael D.Ciletti– Digital Design: With an introduction to Verilog HDL, Pearson Education – 5th Edition- 2014. ISBN:9789332535763.
Reference Books	
1.	Peterson, L.L. and Davie, B.S., 2007. Computer networks: a systems approach. Elsevier.
2.	Thomas L Floyd. 2015. Digital Fundamentals. Pearson Education. ISBN: 9780132737968
3.	Malvino, A.P. and Leach, D.P. and Goutam Saha. 2014. Digital Principles and Applications (SIE). Tata McGraw Hill. ISBN: 9789339203405.
4.	Morris Mano, M. and Michael D.Ciletti. 2014. Digital Design: With an introduction to Verilog HDL. Pearson Education. ISBN:9789332535763
Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar	

List of Challenging Experiments (Indicative)		
1.	Realization of Logic gates using discrete components, verication of truth table for logic gates, realization of basic gates using NAND and NOR gates	4.5 hours
	Implementation of Logic Circuits by verification of Boolean laws and verification of De Morgans law	3 hours
	Adder and Subtractor circuit realization by implementation of Half-Adder and Full-Adder, and by implementation of Half-Subtractor and Full-Subtractor	4.5 hours
	Combinational circuit design i. Design of Decoder and Encoder ii. Design of Multiplexer and De multiplexer iii. Design of Magnitude Comparator iv. Design of Code Converter	4.5 hours
	Sequential circuit design i. Design of Mealy and Moore circuit ii. Implementation of Shift registers iii. Design of 4-bit Counter iv. Design of Ring Counter	4.5 hours
	Implementation of different circuits to solve real world problems: A digitally controlled locker works based on a control switch and two keys which are entered by the user. Each key has a 2-bit binary representation. If the control switch is pressed, the locking system will pass the difference of two keys into the controller unit. Otherwise, the locking system will pass the sum of the two numbers to the controller unit. Design a circuit to determine the input to the controller unit.	4.5 hours
	Implementation of different circuits to solve real world problems: A bank queuing system has a capacity of 5 customers which serves on first come first served basis. A display unit is used to display the number of customers waiting in the queue. Whenever a customer leaves the queue, the count is reduced by one and the count is increased by one if a customer joins a queue. Two sensors (control signals) are used to sense customers leaving and joining the queue respectively. Design a circuit that displays the number of customers waiting in the queue in binary format using LEDs. Binary 1 is represented by LED glow and 0 otherwise.	4.5 hours
	Total Laboratory Hours	**30 hours**

Mode of assessment: Project/Activity			
Recommended by Board of Studies	28-02-2017		
Approved by Academic Council	No. 46	Date	24-08-2017

1. Number Systems

Module:1	INTRODUCTION	3 hours
Number System - Base Conversion - Binary Codes - Complements(Binary and Decimal)		

Common Number Systems

System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, … 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, … 7	No	No
Hexa-decimal	16	0, 1, … 9, A, B, … F	No	No

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

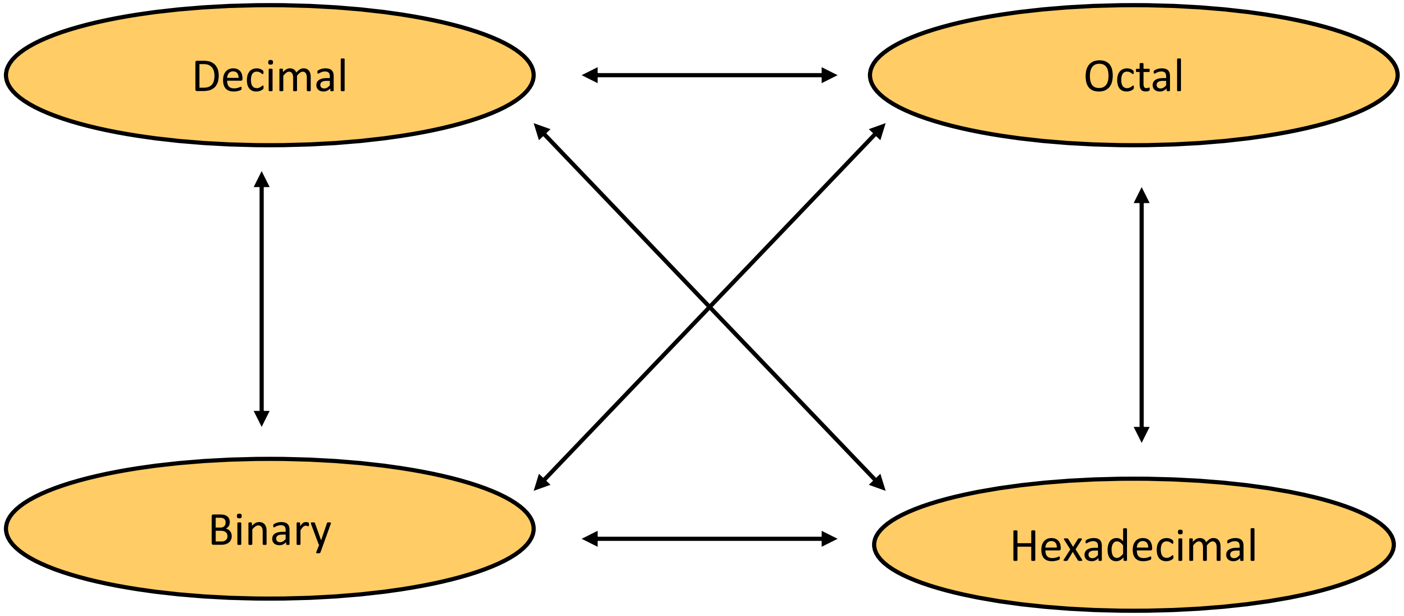
Quantities/Counting (3 of 3)

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:

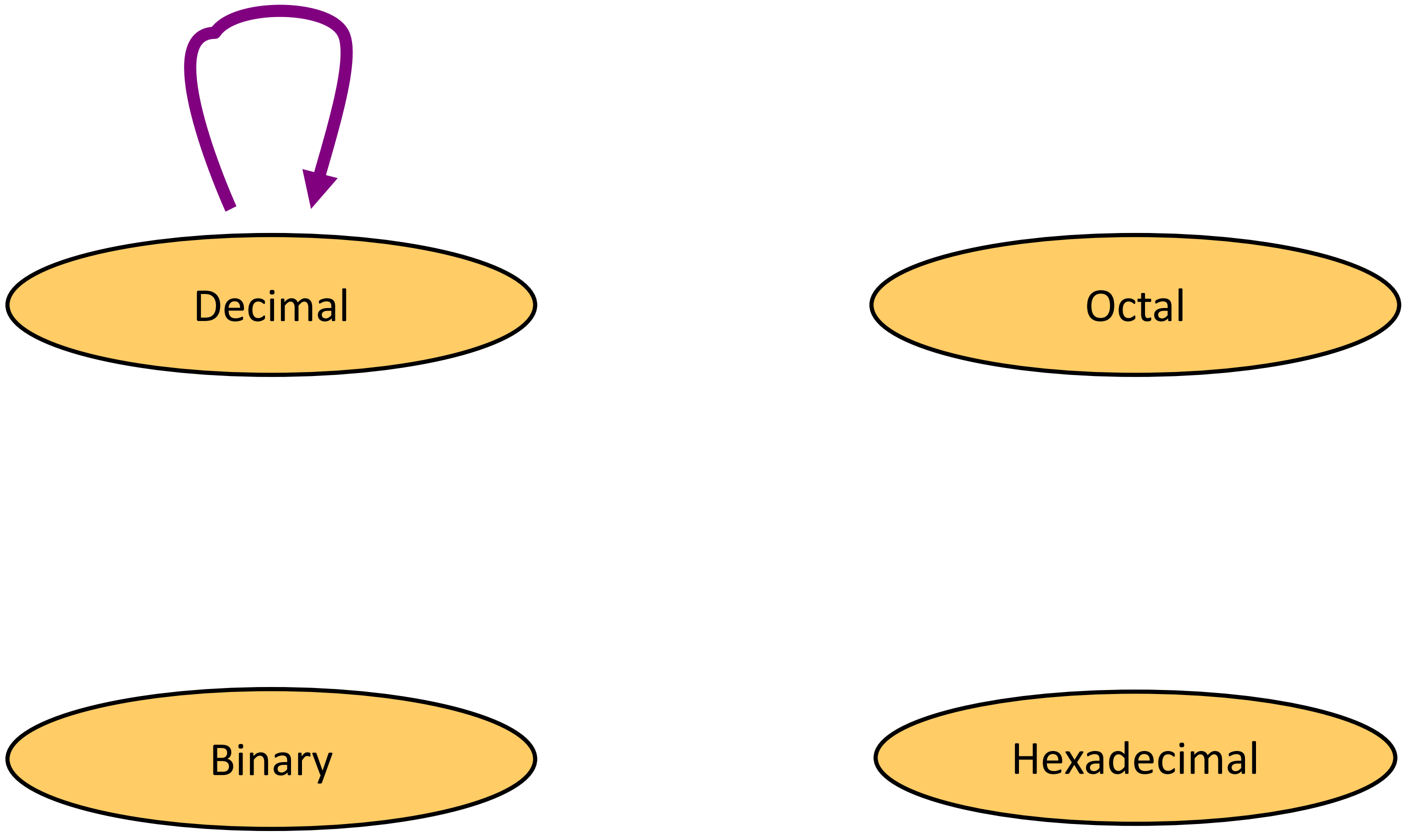


Quick Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

Base/radix

Decimal to Decimal (just for fun)

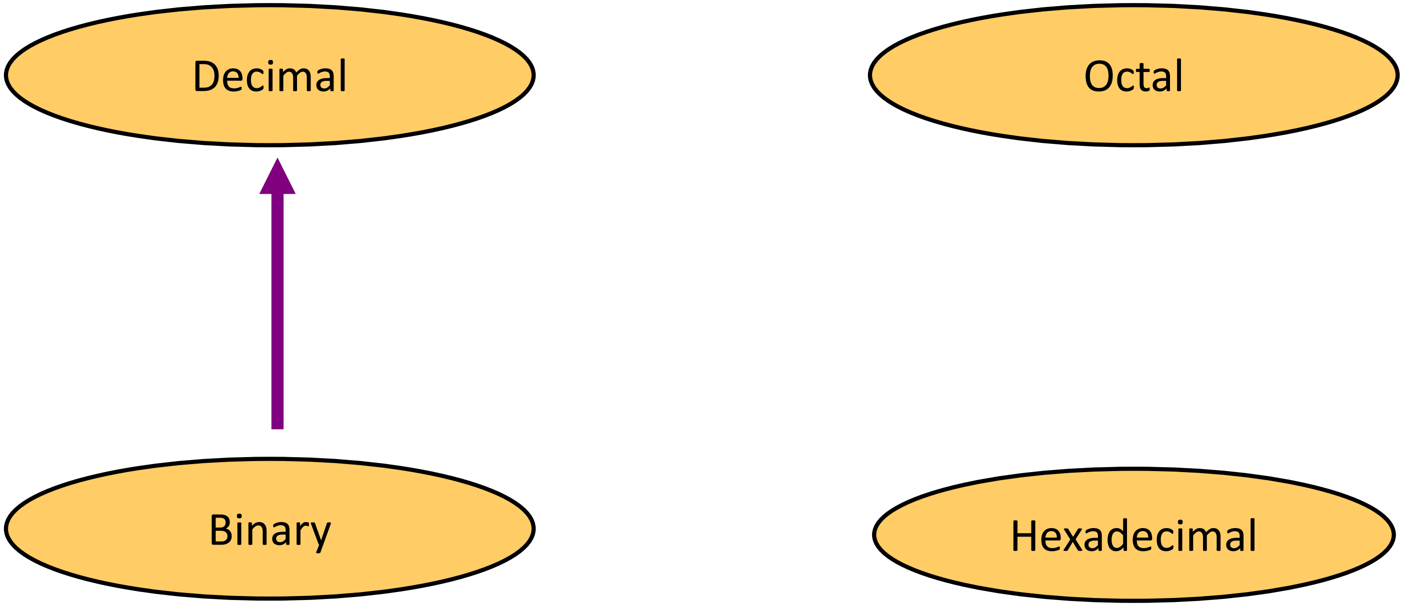


Weight

$$
\begin{array}{llcr}
125_{10} \Rightarrow & 5 \times 10^{0} & = & 5 \\
 & 2 \times 10^{1} & = & 20 \\
 & 1 \times 10^{2} & = & 100 \\
\hline
 & & & 125
\end{array}
$$

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

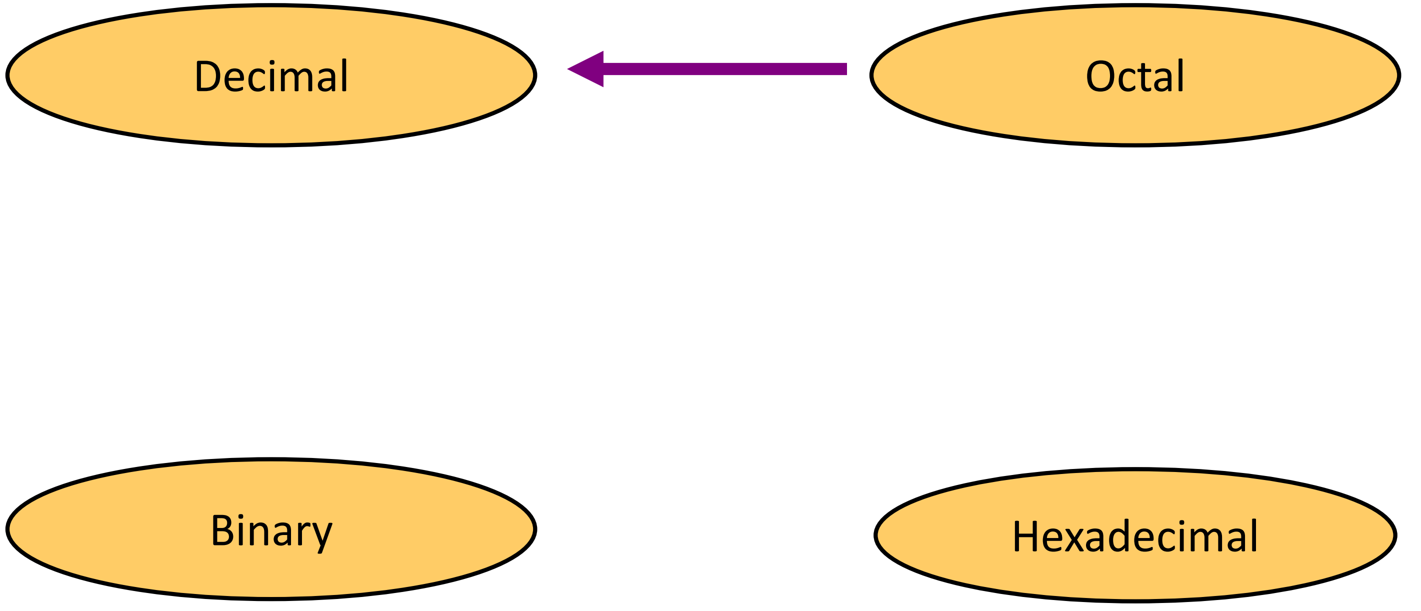
Example

Bit “0”

$$101011_2 \Rightarrow$$

$$
\begin{aligned}
1 \times 2^0 &= 1 \\
1 \times 2^1 &= 2 \\
0 \times 2^2 &= 0 \\
1 \times 2^3 &= 8 \\
0 \times 2^4 &= 0 \\
1 \times 2^5 &= 32 \\
\hline
 & 43_{10}
\end{aligned}
$$

Octal to Decimal



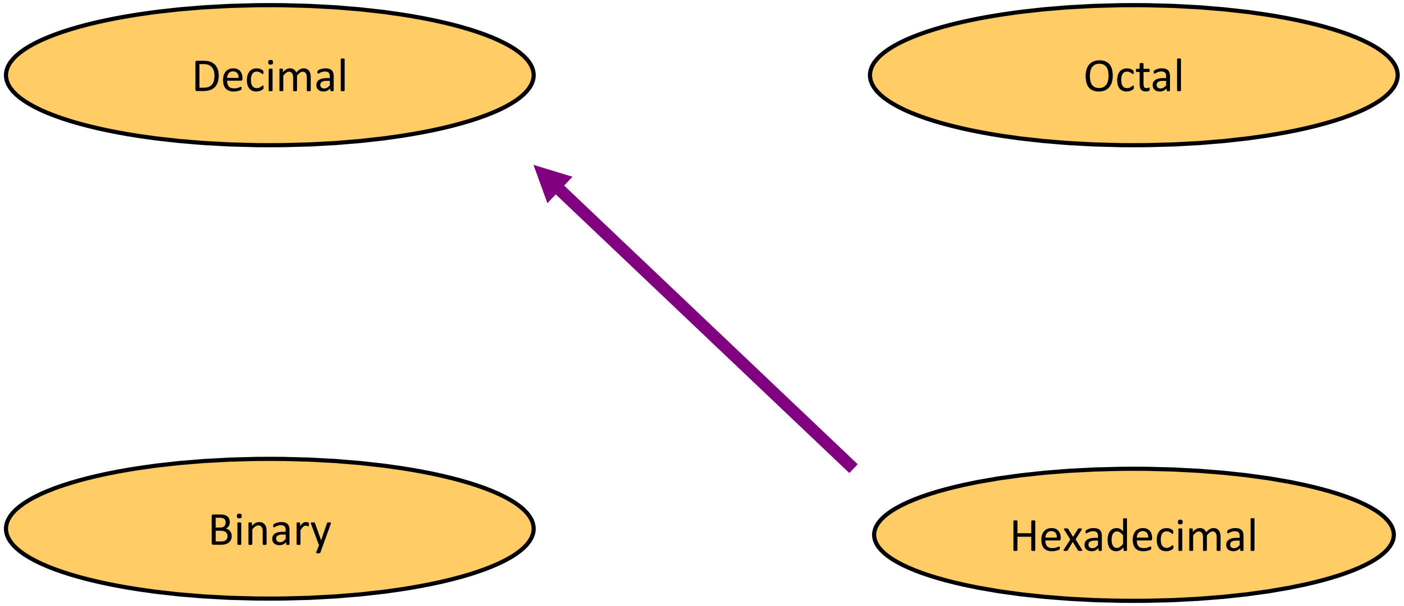
Octal to Decimal

- 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

$$724_8 \Rightarrow \begin{array}{rcrcr} 4 & \times & 8^0 & = & 4 \\ 2 & \times & 8^1 & = & 16 \\ 7 & \times & 8^2 & = & 448 \\ \hline & & & & 468_{10} \end{array}$$

Hexadecimal to Decimal



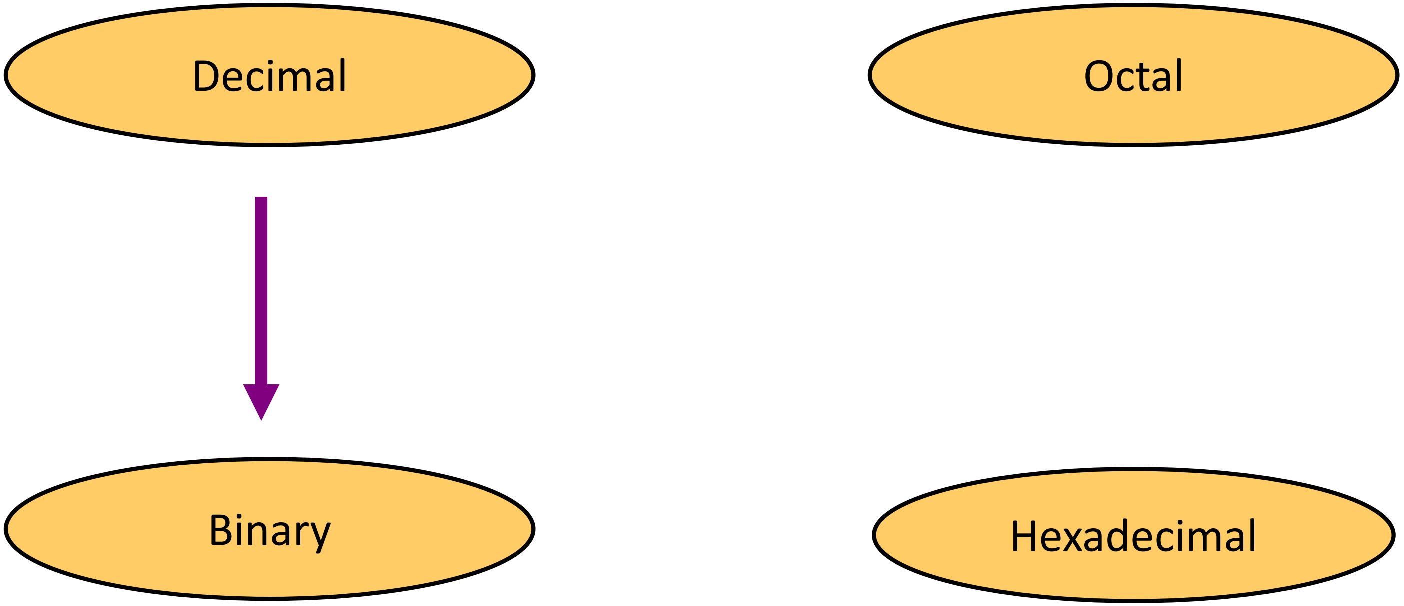
Hexadecimal to Decimal

- 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

$$
\begin{array}{llrcrcr}
ABC_{16} \Rightarrow & C \times 16^0 = & 12 & \times & 1 & = & 12 \\
& B \times 16^1 = & 11 & \times & 16 & = & 176 \\
& A \times 16^2 = & 10 & \times & 256 & = & 2560 \\
\hline
& & & & & & 2748_{10}
\end{array}
$$

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.

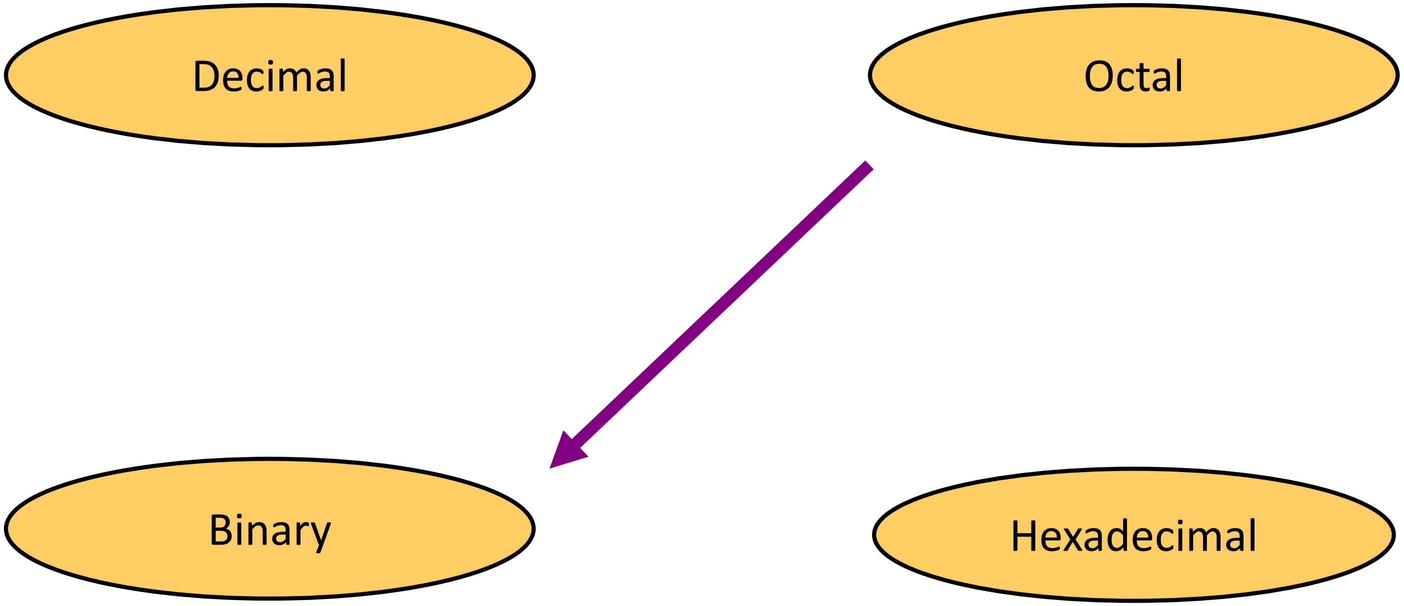
Example

$125_{10} = ?_2$

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

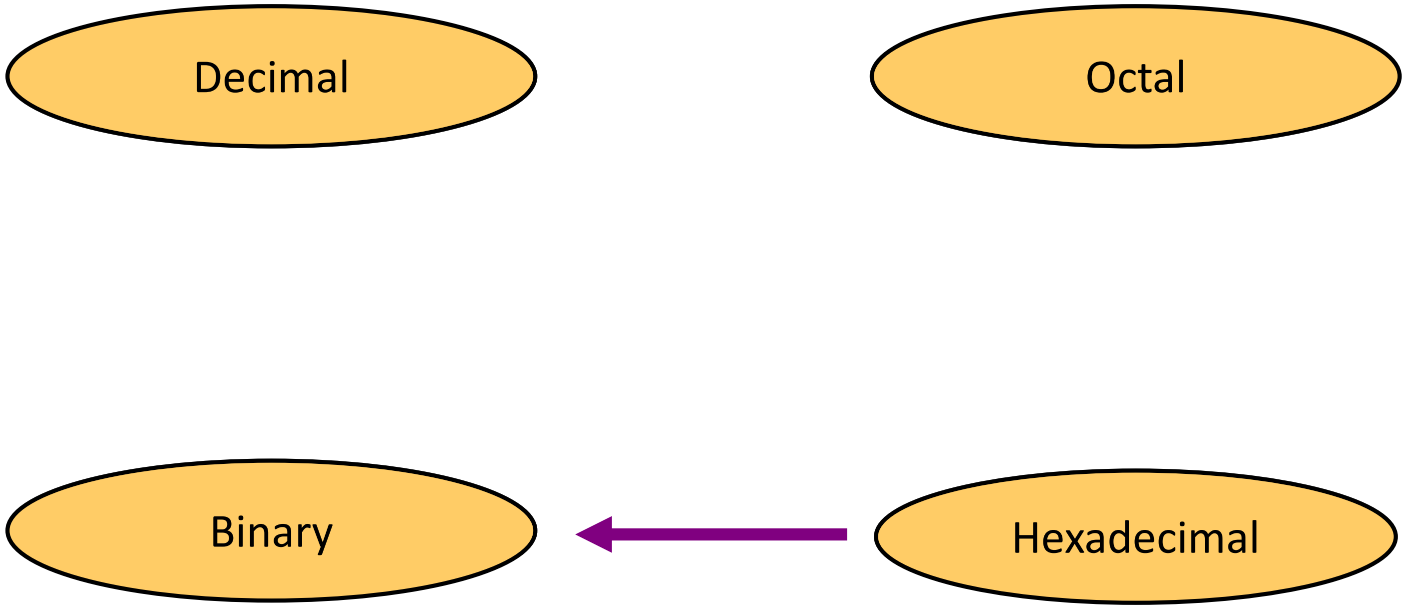
Example

$705_8 = ?_2$

$$\begin{array}{ccc} 7 & 0 & 5 \\ \downarrow & \downarrow & \downarrow \\ 111 & 000 & 101 \end{array}$$

$705_8 = 111000101_2$

Hexadecimal to Binary



Hexadecimal to Binary

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

Example

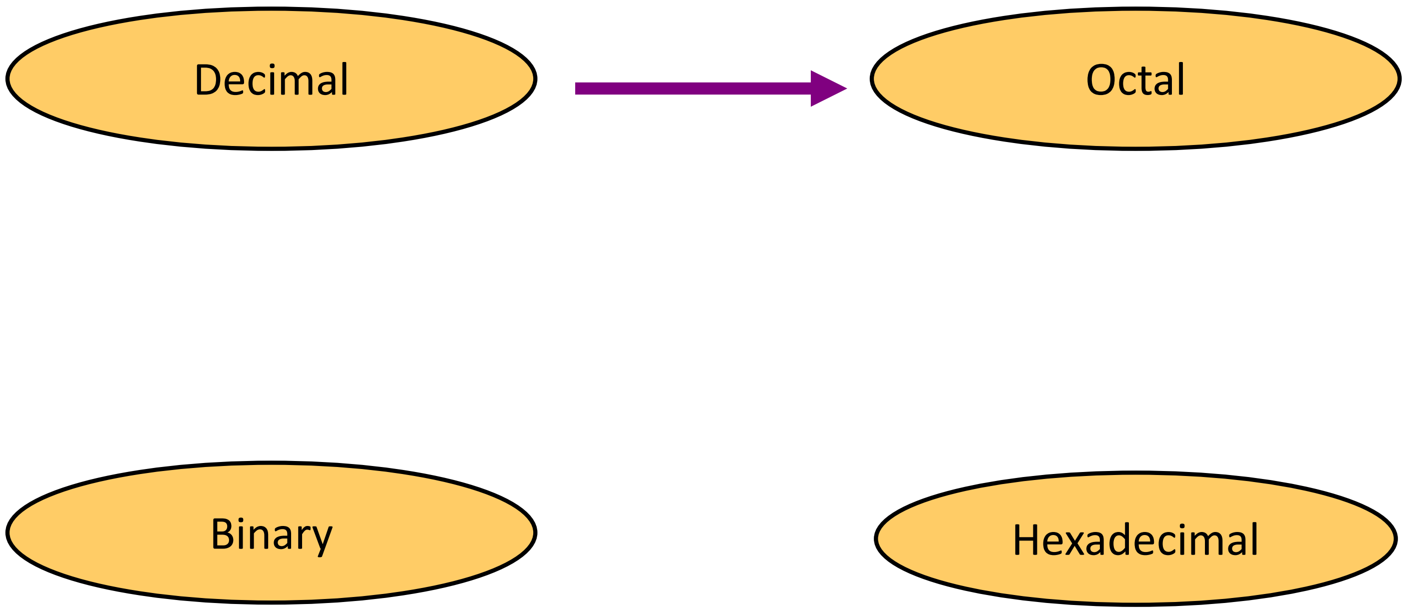
$10AF_{16} = ?_2$

1 0 A F

0001 0000 1010 1111

$10AF_{16} = 0001000010101111_2$

Decimal to Octal



Decimal to Octal

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

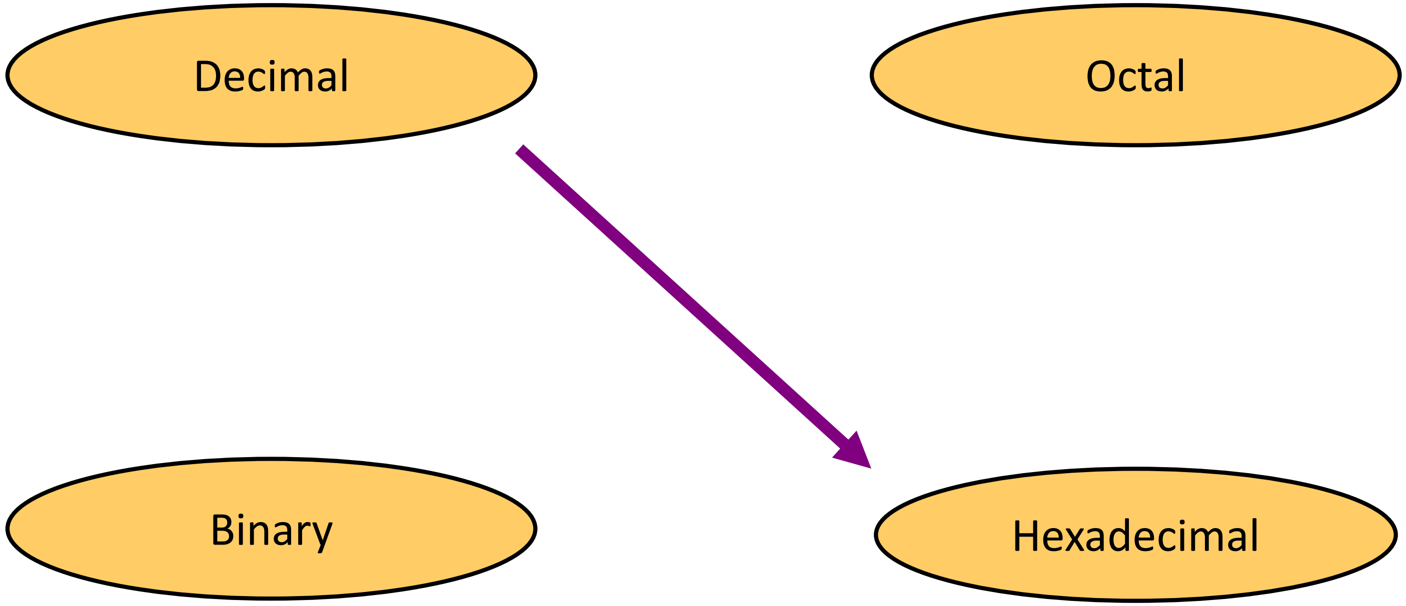
Example

$1234_{10} = ?_8$

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

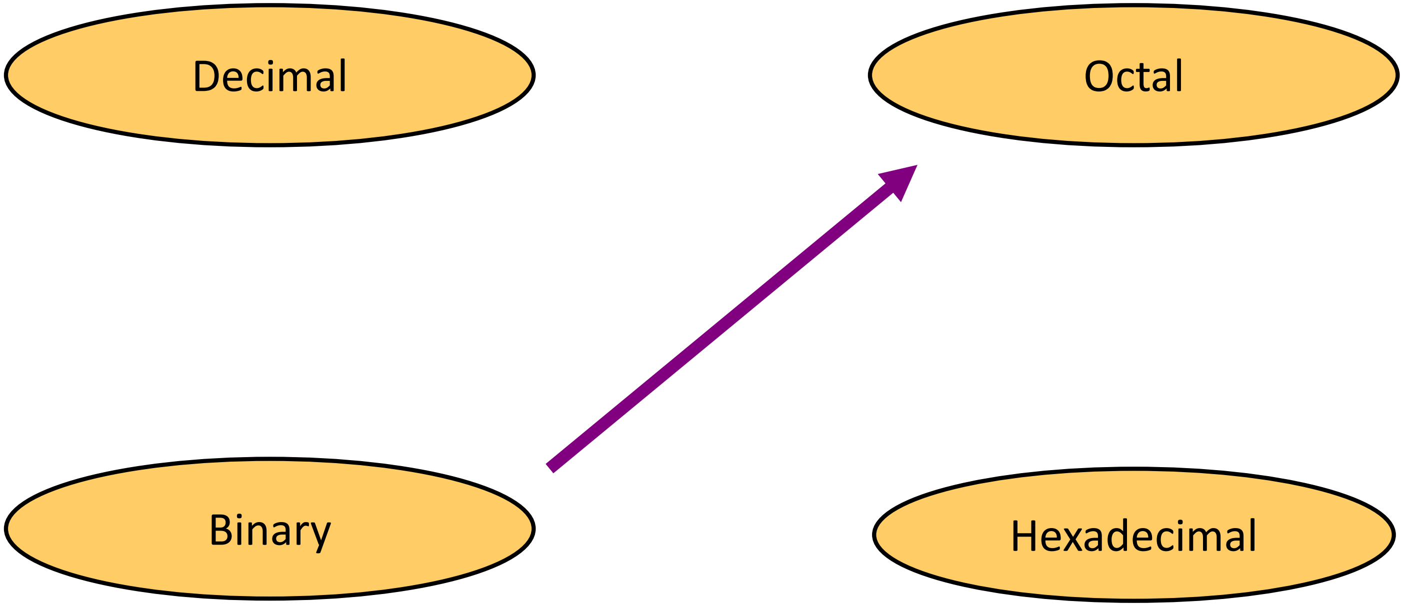
Example

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

$$
\begin{array}{r|l l}
16 & 1234 & \\
\hline
16 & 77 & 2 \\
\hline
16 & 4 & 13 = D \\
\hline
 & 0 & 4
\end{array}
$$

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

Binary to Octal



Binary to Octal

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

Example

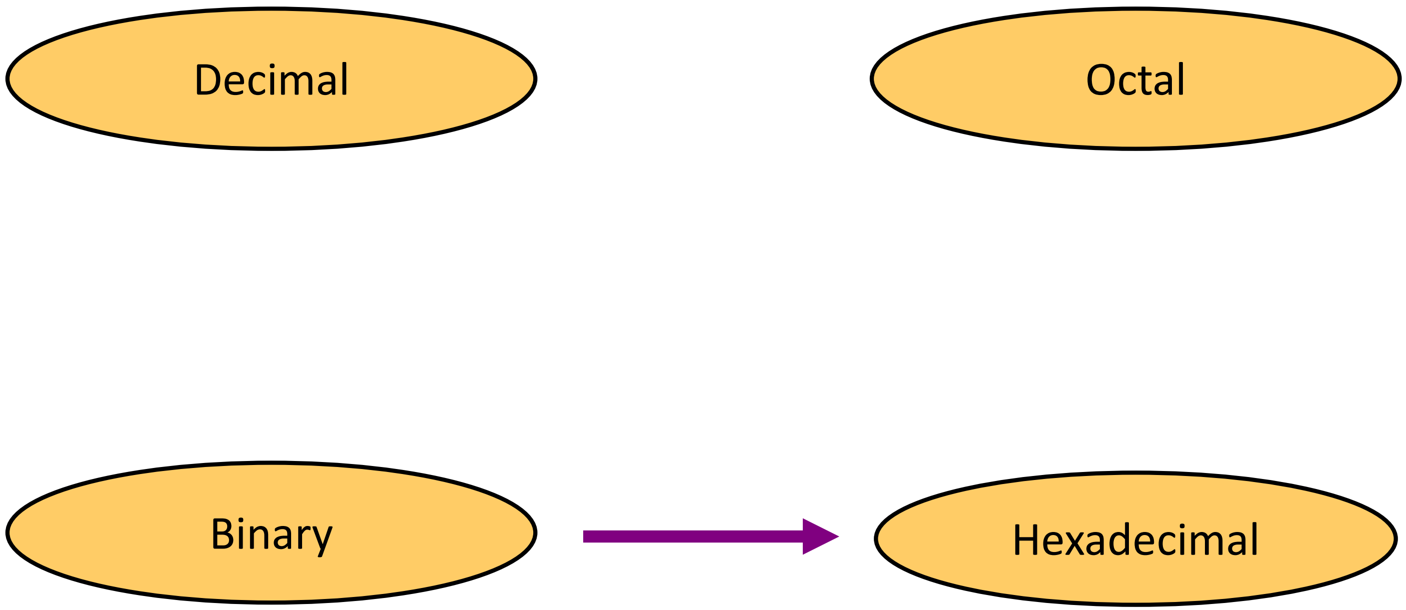
$1011010111_2 = ?_8$

1 011 010 111

1 3 2 7

$1011010111_2 = 1327_8$

Binary to Hexadecimal



Binary to Hexadecimal

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

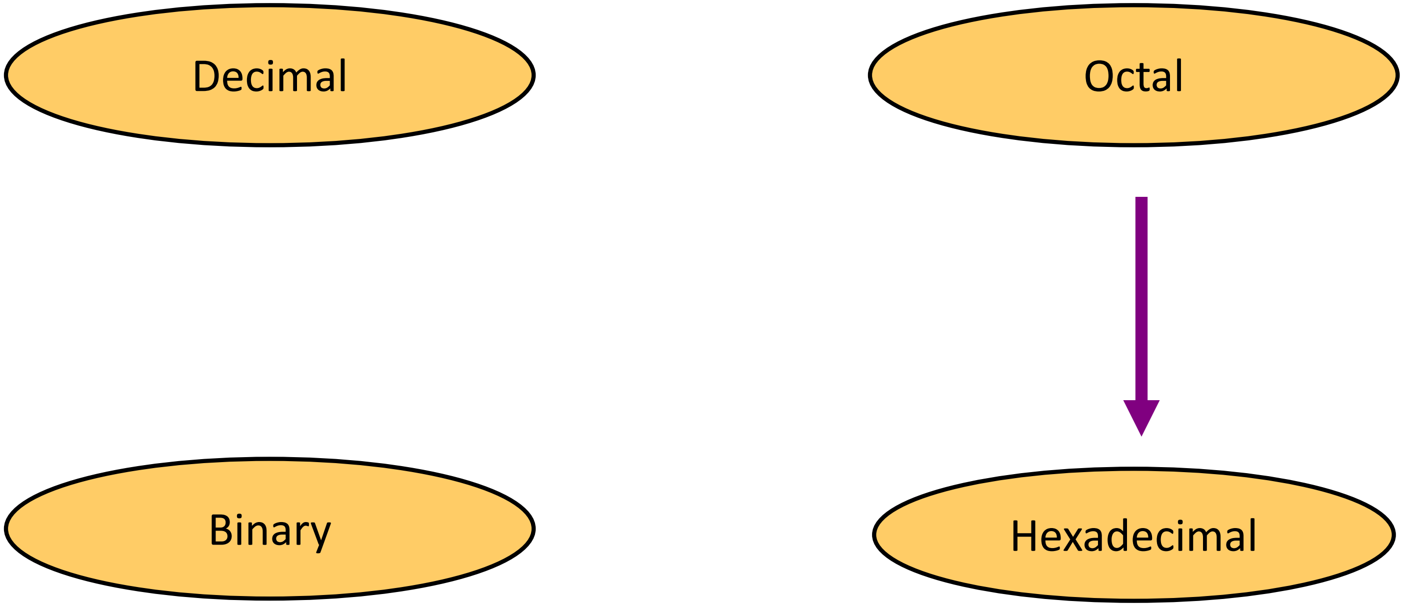
Example

$1010111011_2 = ?_{16}$

$$\begin{array}{ccc} 10 & 1011 & 1011 \\ \downarrow & \downarrow & \downarrow \\ 2 & B & B \end{array}$$

$1010111011_2 = 2BB_{16}$

Octal to Hexadecimal



Octal to Hexadecimal

- Technique
  - Use binary as an intermediary

Example

$1076_8 = ?_{16}$

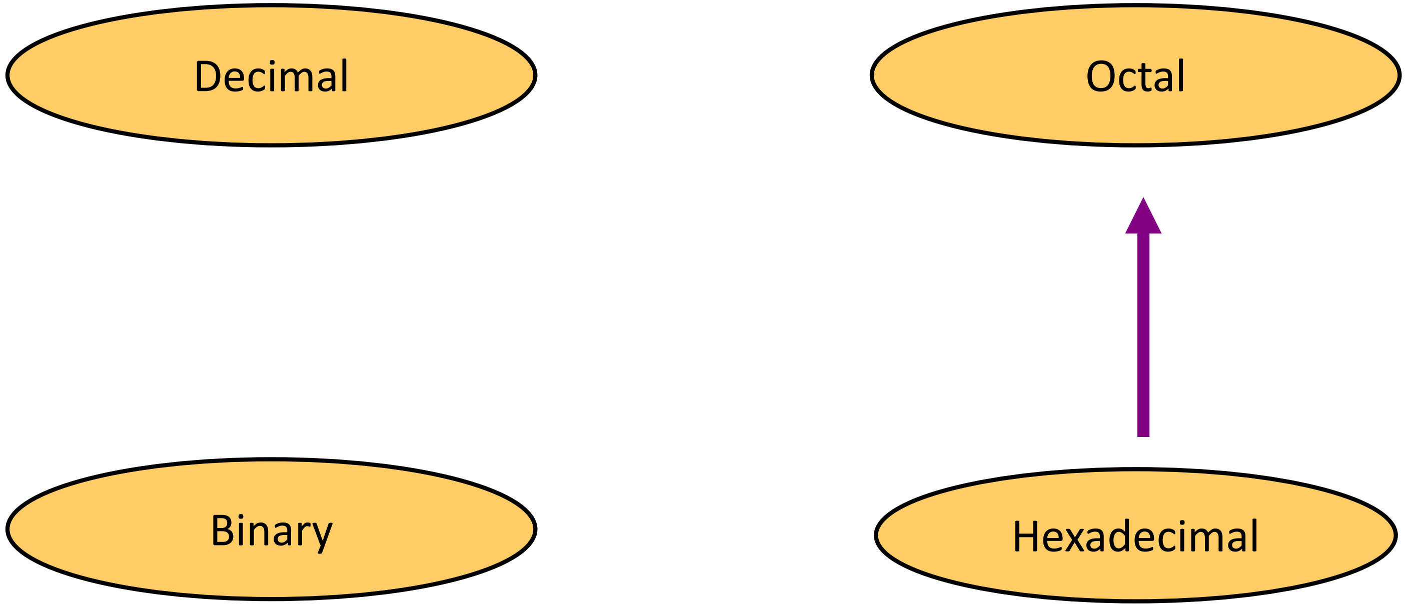
1 0 7 6

001 000 111 110

2 3 E

$1076_8 = 23E_{16}$

Hexadecimal to Octal



Hexadecimal to Octal

- Technique
  - Use binary as an intermediary

Example

$1F0C_{16} = ?_8$

1	F	0	C
0001	1111	0000	1100

0	001	111	1 00	00 1	100
	1	7	4	1	4

$1F0C_{16} = 17414_8$

Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF

Don’t use a calculator!

Skip answer

Answer

Exercise – Convert

Answer

Decimal	Binary	Octal	Hexa-decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF

Common Powers (1 of 2)

- Base 10

Power	Preface	Symbol	Value
10^{-12}	pico	p	.000000000001
10^{-9}	nano	n	.000000001
10^{-6}	micro	µ	.000001
10^{-3}	milli	m	.001
10^{3}	kilo	k	1000
10^{6}	mega	M	1000000
10^{9}	giga	G	1000000000
10^{12}	tera	T	1000000000000

Common Powers (2 of 2)

- Base 2

Power	Preface	Symbol	Value
2^{10}	kilo	k	1024
2^{20}	mega	M	1048576
2^{30}	Giga	G	1073741824

- What is the value of "k", "M", and "G"?
- In computing, particularly w.r.t. <u>memory</u>, the base-2 interpretation generally applies

Fractions

- Decimal to decimal (just for fun)

$$
\begin{array}{llr}
3.14 \Rightarrow & 4 \times 10^{-2} = & 0.04 \\
 & 1 \times 10^{-1} = & 0.1 \\
 & 3 \times 10^{0} = & 3 \\
\hline
 & & 3.14
\end{array}
$$

Fractions

- Binary to decimal

$$
10.1011 \Rightarrow \quad
\begin{aligned}
1 \times 2^{-4} &= 0.0625 \\
1 \times 2^{-3} &= 0.125 \\
0 \times 2^{-2} &= 0.0 \\
1 \times 2^{-1} &= 0.5 \\
0 \times 2^{0} &= 0.0 \\
1 \times 2^{1} &= \underline{2.0} \\
&\ 2.6875
\end{aligned}
$$

Fractions

- Decimal to binary

3.14579

```
 .14579
x     2
-------
0.29158
x     2
-------
0.58316
x     2
-------
1.16632
x     2
-------
0.33264
x     2
-------
0.66528
x     2
-------
1.33056

etc.
```

11.001001...

Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
29.8			
	101.1101		
		3.07	
			C.82

Don't use a calculator!

Skip answer

Answer

Exercise – Convert

Answer

Decimal	Binary	Octal	Hexa-decimal
29.8	11101.110011…	35.63…	1D.CC…
5.8125	101.1101	5.64	5.D
3.109375	11.000111	3.07	3.1C
12.5078125	1100.10000010	14.404	C.82

Complements

- They are used to simplify the subtraction operation
- Two types (for each *base-r* system)
  - Diminishing radix complement (r-1 complement)
  - Radix complement (r complement)

For *n*-digit number *N*

$(r^n - 1) - N$ → **r-1 complement**

$r^n - N$ → **r complement**

9’s and 10’s Complements

- 9’s complement of 674653
  - 999999-674653 = 325346
- 9’s complement of 023421
  - 999999-023421 = 976578
- 10’s complement of 674653
  - 325346+1 = 325347
- 10’s complement of 023421
  - 976578+1=976579

1's and 2's Complements

- 1's complement of 10111001
  - 11111111 – 10111001 = 01000110
  - Simply replace 1's and 0's
- 1's complement of 10100010
  - 01011101
- 2's complement of 10111001
  - 01000110 + 1 = 01000111
  - Add 1 to 1's complement
- 2's complement of 10100010
  - 01011101 + 1 = 01011110

Subtraction with Complements of Unsigned

- M – N
  - Add M(minuend) to r’s complement of N (subtrahend)
    - *Sum* = $M+(r^n - N) = M - N + r^n$
  - If M > N, *Sum* will have an end carry r^n , can be discarded
  - If M<N, *Sum* will not have an end carry and
    - *Sum* = $r^n - (N - M)$ (which is r’s complement of N – M)
    - So M – N = – (r’s complement of Sum)

Subtraction with Complements of Unsigned

- 65438 – 5623 (using 10’s complement)

	65438
10’s complement of 05623	**+94377**
	159815
Discard end carry 10^5	**-100000**
Answer	**59815**

Subtraction with Complements of Unsigned

- 5623 – 65438 (using 10’s complement)

	05623
10’s complement of 65438	**+34562**
	40185

There is no end carry =>

-(10’s complement of 40185)

-59815

Subtraction with Complements of Unsigned

- 10110010 – 10011111 (using 2’s complement)

	10110010
2’s complement of 10011111	**+01100001**
	100010011
Discard end carry 2^8	**-100000000**
Answer	**000010011**

Subtraction with Complements of Unsigned

- 10011111 -10110010 (using 2's complement)

	10011111
2's complement of 10110010	**<u>+01001110</u>**
	11101101

There is no end carry =>
-(2's complement of 11101101)
Answer = -00010011

1010.11 – 1001.01

Solution:

2’s complement of 1001.01 is 0110.11. Hence

Minued -		1 0 1 0 . 1 1
2’s complement of subtrahend -		0 1 1 0 . 1 1
Carry over	1	0 0 0 1 . 1 0

After dropping the carry over we get the result of subtraction as 1.10.

10100.01 – 11011.10

Solution:

2’s complement of 11011.10 is 00100.10. Hence

Minued -	1 0 1 0 0 . 0 1
2’s complement of subtrahend -	0 1 1 0 0 . 1 0
Result of addition -	1 1 0 0 0 . 1 1

As there is no carry over the result of subtraction is negative and is obtained by writing the 2’s complement of 11000.11.

Hence the required result is – 00111.01.

Subtraction with Complements of Unsigned

- 10110010 – 10011111 (using 1's complement)

Subtraction with Complements of Unsigned

- 10011111 -10110010 (using 1’s complement)

Signed Binary Numbers

- Unsigned representation can be used for positive integers
- How about negative integers?
  - Everything must be represented in binary numbers
  - Computers cannot use – or + signs

Negative Binary Numbers

- Three different systems have been used
  1. Signed magnitude (used in ordinary arithmetic)
  2. Signed compliment (used in computer)
     - (i) One’s complement
     - (ii) Two’s complement (most commonly used)

NOTE: For negative numbers the sign bit is always 1, and for positive numbers it is 0 in these three systems

Signed Magnitude

- The leftmost bit is the sign bit (0 is + and 1 is -) and the remaining bits hold the absolute magnitude of the number
- Examples
  - -47 = 1 0 1 0 1 1 1 1
  - 47 = 0 0 1 0 1 1 1 1

For 8 bits, we can represent the signed integers –128 to +127

How about for N bits?

One’s complement

- Replace each 1 by 0 and each 0 by 1
- Example (-6)
  - First represent 6 in binary format (00000110)
  - Then replace (11111001)

Two’s complement

- Find one’s complement
- Add 1
- Example (-6)
  - First represent 6 in binary format (00000110)
  - One’s complement (11111001)
  - Two’s complement (11111010)

Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Arithmetic Addition

- Usually represented by 2's complement

+ 5	00000101
+11	00001011
+16	00010000

Discard

- 5	11111011
+11	00001011
+6	100000110

+ 5	00000101
-11	11110101
-6	11111010

- 5	11111011
-11	11110101
-16	111110000

Discard

Arithmetic subtraction

(+ or - A) – (+B) = (+ or – A) + (-B)

(+ or - A) – (-B) = (+ or – A) + (+B)

Example: (-6) – (-13) ?

Binary Code

In the coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code. The digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**. The binary code is represented by the number as well as alphanumeric letter.

Advantages of Binary Code

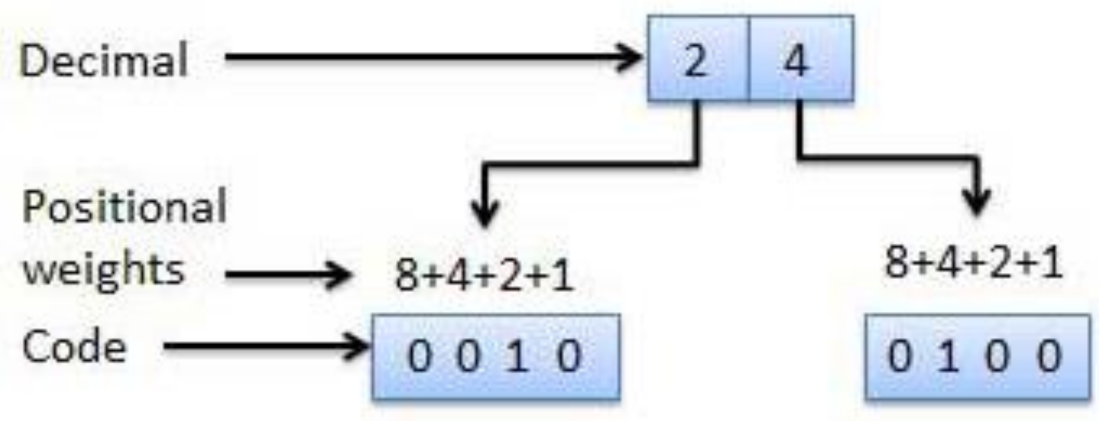
Following is the list of advantages that binary code offers.

- Binary codes are suitable for the computer applications.
- Binary codes are suitable for the digital communications.
- Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- Since only 0 & 1 are being used, implementation becomes easy.

Classification of codes

Weighted codes:Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.

Example: Binary, BCD, 8421, 2421



Non-weighted codes: In this type of binary codes, the positional weights are not assigned.

Example: ex-3, gray (unit distance code)

Reflective code (self complementing): 2421, ex-3 , 8 4 -2 -1

Alpha numeric codes: ASCII

Error detection and correction codes: Hamming, parity

Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combinations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1