Lesson 14…..Binary, Hex, and Octal

We will examine four different number systems here,…decimal, binary, hexadecimal (hex), and octal. In your study of these number systems it is very important to note the **similarities** of each. Study these similarities carefully. This is ultimately how you will understand the new number systems.

Decimal, base 10

There are only **10** digits in this system:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Note that even though this is base **10**, there is no single digit for **10**. Instead we use two of the permissible digits, 1 and 0 to make **10**.

<u>Positional value</u>: Consider the decimal number 5,402.

1000	100	10	1
10^3	10^2	10^1	10^0
5	4	0	2

2 * 1 = 2
0 * 10 = 0
4 * 100 = 400
5 * 1000 = <u>5000</u>
5402

Binary, base 2

There are only **2** digits in this system:
0, 1

Note that even though this is base **2**, there is no single digit for **2**. Instead we use two of the permissible digits, 1 and 0 to make **10_{bin} (2_{dec})**.

<u>Positional value</u>: Consider the conversion of binary number 1101_{bin} to decimal form.

8	4	2	1
2^3	2^2	2^1	2^0
1	1	0	1

1 * 1 = 1
0 * 2 = 0
1 * 4 = 4
1 * 8 = <u>8</u>
13_{dec}

<u>Bits and Bytes</u>: Each of the positions of 1101_{bin} is called a bit… it's a four-bit number. When we have **eight bits** (example, 10110101_{bin}) we call it a **byte**. If we say that a computer has 256mb of memory, where *mb* stands for megabytes, this means it

has 256 million bytes. See Appendix Y for more on kilobytes, megabytes, and gigabytes, etc.

Hexadecimal (hex), base 16

There are only **16** digits in this system:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A,  B,  C,  D,  E,  F
                              |    |    |    |    |   |
                             10  11  12  13  14  15
```

Note that even though this is base **16**, there is no single digit for **16**. Instead we use two of the permissible digits, 1 and 0 to make $\mathbf{10_{hex}}$ $\mathbf{(16_{dec})}$.

Positional value: Consider the conversion of hex number $5C02_{hex}$ to decimal form.

```
4096   256   16    1
16^3   16^2  16^1  16^0
5      C     0     2
|      |     |     |
|      |     |     |_____  2 *    1   =      2
|      |     |___________  0 *   16   =      0
|      |_________________ 12 *  256   =   3072
|________________________  5 * 4096   =  20480
                                         23554dec
```

Octal, base 8

There are only **8** digits in this system:

0, 1, 2, 3, 4, 5, 6, 7

Note that even though this is base **8**, there is no single digit for **8**. Instead we use two of the permissible digits, 1 and 0 to make $\mathbf{10_{oct}}$ $\mathbf{(8_{dec})}$.

Positional value: Consider the conversion of octal number 5402_{oct} to decimal form.

```
512   64    8     1
8^3   8^2   8^1   8^0
5     4     0     2
|     |     |     |
|     |     |     |_____  2 *   1   =      2
|     |     |___________  0 *   8   =      0
|     |_________________  4 *  64   =    256
|_______________________  5 * 512   =   2560
                                        2818dec
```

Following are examples that show how we can use these different number systems with Java.

Store a hex number:

```
int x = 0x4CB3;  //the leading 0x indicates hex format
System.out.println(x);  //19635 ...Notice it automatically prints in decimal form
```

Store an octal number:

int x = 0734; //**the leading 0 indicates octal format**
System.out.println(x); //476 …Notice it automatically prints in decimal form

Convert an integer variable to a hex *String*:

int x = 3901;
System.out.println(Integer.toHexString(x)); //$\mathbf{f3d}_{\mathbf{hex}}$
//…or Integer.toString(x, 16);

Convert an integer variable to a binary *String*:

int x = 3901;
System.out.println(Integer.toBinaryString(x)); // $\mathbf{111100111101}_{\mathbf{bin}}$
//…or Integer.toString(x, 2);

Convert an integer variable to an octal *String*:

int x = 3901;
System.out.println(Integer.toOctalString(x)); //$\mathbf{7475}_{\mathbf{oct}}$
//…or Integer.toString(x, 8);

Notice in the last three examples above the following method was an alternate way to convert bases:

String s = Integer.toString(i, b);

The first parameter *i* is of type *int* and *b* is the base to which we wish to convert. *b* (also an *int* type) can be any base ranging from 2 to 36. Just as for hexadecimal numbers where we use the letters A – F, the bases higher than 16 (hex) use the remaining letters of the alphabet. For example, Integer.toString(8162289, 32) returns "7p2vh".

Base conversion using *parseInt*:

It is also possible to go from some strange base (in *String* form) back to a decimal *int* type. For example, *Integer.parseInt("3w4br", 35)* converts $3w4br_{35}$ into 5879187_{dec}.

A technique for converting 147 from decimal to binary:

2	147		; 2 divides into 147 73 times with a remainder of
2	73	1	;1. 2 divides into 73, 36 times with a remainder of
2	36	1	;1. 2 divides into 36, 18 times with a remainder of
2	18	0	;0. 2 divides into 18, 9 times with a remainder of
2	9	0	;0. etc.
2	4	1	
2	2	0	
2	1	0	
	0	1	Now list the 1's and 0's from <u>bottom to top</u>. $\mathbf{10010011}_{\mathbf{bin}}$ **=** $\mathbf{147}_{\mathbf{dec}}$

A technique for converting 3741 from decimal to hex:

16	3741		;divide 3741 by 16. It goes 233 times with a
16	233	13	;remainder of 13.
16	14	9	
	0	14	

Now list the numbers from bottom to top. Notice, when listing the 14 we give its hex equivalent, E, and for 13 we will give D.

$E9D_{hex} = 3741_{dec}$

An octal multiplication example (47_{oct} * 23_{oct}):

```
  2
 47
 23
  5
```

$3 * 7 = 21_{dec}$

8 divides into 21 2 times with a remainder of 5. Notice the 5 and the carry of 2

```
 1 2
  47
  23
 165
```

$(3*4) + 2 = 14_{dec}$

8 divides into 14 1 time with a remainder of 6

```
  1
  47
  23
 165
  6
```

$2 * 7 = 14_{dec}$

8 divides into 14 1 time with a remainder of 6

```
 11
   47
   23
  165
 116
```

$2*4 + 1 = 9_{dec}$

8 divides into 9 1 time with a remainder of 1

Now we are ready to add:

```
  1
  165
 116 .
 1345oct
```

Notice in adding 6 + 6 we get 12. 8 divides 12 1 time, remainder 4.

Binary addition:

The rules to remember are:

0 + 0 = 0   0 + 1 = 1   1 + 1 = 0 with a carry of 1

Add the two binary numbers 110011 and 100111.

1			1	1	1	
	1	1	0	0	1	1
	1	0	0	1	1	1
1	0	1	1	0	1	0

The problem we have done here:

$110011_{bin} + 100111_{bin} = 1011010_{bin}$

is equivalent to:

$51_{dec} + 39_{dec} = 90_{dec}$

A trick for converting binary into hex:

Begin with the binary number 10110111010. Starting on the right side, partition this into groups of four bits and get 101 1011 1010 To each four bit group, assign a hex digit.
5 B A

Thus we have $10110111010_{bin} = 5BA_{hex}$. Similarly, partition a binary number into groups of 3 to convert to Octal.

See Appendix D for the decimal, hex, octal, and binary equivalents of 0 – 127.

For an enrichment activity concerning a **Binary** File Editor, see Appendix U. There, you will have an opportunity to specify software, search on the Internet, and publish the information you discover…. Appendix G explains how negative numbers are handled in the binary system.

Project… Basically Speaking

Create a project called *TableOfBases* with class *Tester*. The *main* method should have a *for* loop that cycles through the integer values 65 <= j <= 90 (These are the ASCII codes for characters A – Z). Use the methods learned in this lesson to produce a line of this table on each pass through the loop. Display the equivalent of the decimal number in the various bases just learned (binary, octal, and hex) as well as the character itself:

Decimal	Binary	Octal	Hex	Character
65	1000001	101	41	A
66	1000010	102	42	B
67	1000011	103	43	C
68	1000100	104	44	D
69	1000101	105	45	E
70	1000110	106	46	F
71	1000111	107	47	G
72	1001000	110	48	H
73	1001001	111	49	I
74	1001010	112	4a	J
75	1001011	113	4b	K
76	1001100	114	4c	L
77	1001101	115	4d	M
78	1001110	116	4e	N
79	1001111	117	4f	O
80	1010000	120	50	P
81	1010001	121	51	Q
82	1010010	122	52	R
83	1010011	123	53	S
84	1010100	124	54	T
85	1010101	125	55	U
86	1010110	126	56	V
87	1010111	127	57	W
88	1011000	130	58	X
89	1011001	131	59	Y
90	1011010	132	5a	Z

Exercise on Lesson 14

1. Convert $3C4F_{hex}$ to decimal.

2. Convert 100011_{bin} to decimal.

3. Convert 637_{oct} to decimal.

4. Is the following code legal? If not, why?
```
int v = 04923;
```

5. Is the following code legal? If not, why?
```
int w = 0xAAFF;
```

6. Convert $9A4E_{hex}$ to decimal.

7. Convert 1011011_{bin} to decimal.

8. Convert 6437_{oct} to decimal.

9. Write code that will store $5C3B_{hex}$ in the integer variable *a*.

10. Write code that will store 3365_{oct} in the integer variable *k*.

11. Convert 478_{dec} to binary.

12. Convert 5678_{dec} to hex.

13. Convert 5678_{dec} to octal.

14. Multiply $2C6_{hex}$ times $3F_{hex}$ and give the answer in hex.

15. Add 3456_{oct} and 745_{oct} and give the answer in octal.

16. What is the decimal equivalent of A_{hex}?

17. What is the decimal equivalent of 8_{hex}?

18. What is the base of the hex system?

19. How do you write 16_{dec} in hex?

20. What is the base of the binary system?

21. Add these two binary numbers:

    1111000 and 1001110.

22. Add these two binary numbers:

    1000001 and 1100001

23. Explain the following "joke".

    "There are only 10 types of people in the world…those who understand binary and those who don't."

24. Suppose you have *String s* that represents a number that you know is expressed in a base given by *int b*. Write code that will convert this into an equivalent decimal based integer and store the result in *int i*.

25. Show code that will convert $9322gf_{33}$ into *String s* that is the equivalent number in base 28.

26. Add $3FA6_{hex}$ to $E83A_{hex}$ and give the answer in hex.

27. Multiply 7267_{oct} times 4645_{oct} and give the answer in octal.

28. Add 2376_{oct} to 567_{oct} and give the answer in octal.

29. Multiply $3E_{hex}$ times $5B_{hex}$ and give the answer in hex.