Chapter 4: Computer Number System

Before discussing numbering systems with in a computer context there are a few terms we need to define.

1. Bit:
    - Is the smallest unit of data representation.
    - Is a space that can hold a single state.
2. Byte:
    - Is the smallest unit of data measurement.
    - Is a single group of 8 bits. [1 Byte = 8 Bits.]
3. Word:
    - Is a group of 8 Bytes. [1 word = 8 Bytes.]

Decimal Number System

Z = {0,1,2,3,4,5,6,7,8,9}; B = 10

• Each position to the left of a digit increases by a power of 10.

• Each position to the right of a digit decreases by a power of 10.

Example:

$(47692)_{10} = 2 * 10^0 + 9 * 10^1 + 6 * 10^2 + 7 * 10^3 + 4 * 10^4$

$= 2 + 90 + 600 + 7000 + 40000$

Binary Number System

• Has two states 1(ON state) and 0(OFF state).

Z = {0,1}; B = 2

<u>Unsigned number representation</u>

With **n** bits we can represent $\mathbf{2^n}$ unsigned integer numbers.

Range: **from 0 to $\mathbf{2^n-1}$**

So with 8 bits we can represent 2^8 (256) numbers from 0 to 2^8-1(255)

• Each position to the left of a digit increases by a power of 2.

• Each position to the right of a digit decreases by a power of 2.

Example:

$(11001)_2 = 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3 + 1*2^4$

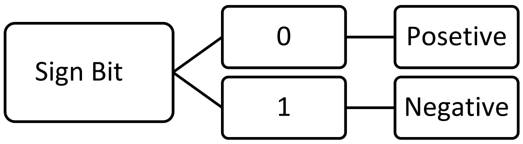
$= 1 + 0 + 0 + 8 + 16 = \underline{\underline{(25)_{10}}}$

Signed number representation

Here we would need to consider sign, meaning we represent both positive and negative numbers. To do this we will first look at a method called sign and magnitude.

Sign and magnitude

With this method we reserve the most significant bit of the left most bit for sign. This bit is called sign bit.



Similar to the unsigned numbers with n bits we represent 2n numbers. The difference here is that both positive and negative are to be represented.

Range:

- Since we took 1 bit for parity we have n-1 bits left for the number.
- We want to include negative as well as positive numbers and zero so our range starts from the negative goes through zero and reaches the positive.

With these two ideas in mind our range becomes **from $-(2^{n-1}-1)$ to $2^{n-1}-1$**

So with 8 bits we can represent 28 numbers (256)

$-(2^{8-1}-1)$ to $(2^{8-1}-1) = -(2^7-1)$ to $(2^7-1) = -127$ to 127.

Example: $(11001)_2$

The first bit (MSB) indicates the number is negative since it is 1.

$= 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3$

$= -(1 + 0 + 0 + 8) = (-9)_{10}$

Octal Number System

Z = {0,1,2,3,4,5,6,7}; B = 8

• Each position to the left of a digit increases by a power of 8.

• Each position to the right of a digit decreases by a power of 8.

Example:

$(12403)_8 = 3 * 8^0 + 0 * 8^1 + 4 * 8^2 + 2 * 8^3 + 1 * 8^4$

$= 3 * 1 + 0 * 8 + 4 * 64 + 2 * 512 + 1 * 4096$

$= (5379)_{10}$

Hexadecimal Number System

Z = (0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F}; B = 16

• Each position to the left of a digit increases by a power of 16.

• Each position to the right of a digit decreases by a power of 16.

Example:

FB40A16 = $10 * 16^0 + 0 * 16^1 + 4 * 16^2 + 11 * 16^3 + 15 * 16^4$

= 10 * 1 + 0 * 16 + 4 * 256 + 11 * 4096 + 15 * 65536

= $\underline{(1{,}029{,}130)_{10}}$

Comparison of Number Systems

Decimal	Dual	Octal	Hexadecimal	Decimal	Dual	Octal	Hexadecimal
0	00 000	0 0	0 0	16	10 000	2 0	1 0
1	00 001	0 1	0 1	17	10 001	2 1	1 1
2	00 010	0 2	0 2	18	10 010	2 2	1 2
3	00 011	0 3	0 3	19	10 011	2 3	1 3
4	00 100	0 4	0 4	20	10 100	2 4	1 4
5	00 101	0 5	0 5	21	10 101	2 5	1 5
6	00 110	0 6	0 6	22	10 110	2 6	1 6
7	00 111	0 7	0 7	23	10 111	2 7	1 7
8	01 000	1 0	0 8	24	11 000	3 0	1 8
9	01 001	1 1	0 9	25	11 001	3 1	1 9
10	01 010	1 2	0 A	26	11 010	3 2	1 A
11	01 011	1 3	0 B	27	11 011	3 3	1 B
12	01 100	1 4	0 C	28	11 100	3 4	1 C
13	01 101	1 5	0 D	29	11 101	3 5	1 D
14	01 110	1 6	0 E	30	11 110	3 6	1 E
15	01 111	1 7	0 F	31	11 111	3 7	1 F

Conversions Between Number Systems

Conversion of Natural Numbers: Base 10 ⇒ Base B Conversion

We use the method of iterated division to convert a number N_{10} of base 10 into a number N_B of base B.

Step 1. Divide the number N_{10} by: whole number BN & remainder R.

- ⇨ Take N as the next number to be divided by B.
- ⇨ Keep R as the next left-most digit of the new number.

Step 2. If N is zero then STOP, else set N10 = N and go to step 1.

Example: Decimal to Binary $(1020)_{10} = (\ )_2$

Number to be converted: $N_{10} = 1020$; target base B=2

1020 / 2 = 510 Remainder: 0

510 / 2 = 255 Remainder: 0

255 / 2 = 127 Remainder: 1

127/ 2 = 63 Remainder: 1

63 / 2 = 31 Remainder: 1

31 / 2 = 15 Remainder: 1

15 / 2 = 7 Remainder: 1

7 / 2 = 3 Remainder: 1

3 / 2 = 1 Remainder: 1

1 / 2 = **0** Remainder: 1

$(1020)_{10} = (1111111100\ )_2$

Powers of 2

N	2^N	N	2^N
0	1	17	131,072
1	2	18	262,144
2	4	19	524,288
3	8	20	1,048,576
4	16	21	2,097,152
5	32	22	4,194,304
6	64	23	8,388,608
7	128	24	16,777,216
8	256	25	33,554,432
9	512	26	67,108,864
10	1,024	27	134,217,728
11	2,048	28	268,435,456
12	4,096	29	536,870,912
13	8,192	30	1,073,741,824
14	16,384	31	2,147,483,648
15	32,768	32	4,294,967,296
16	65,536	33	8,589,934,592

Note:

By looking at the power of 2, we can figure out a better way of converting decimals to binary.

Example: **Decimal to Octal** $(1020)_{10} = ()_8$

Number to be converted: ; target base

1020 / 8 = 127 Remainder: **4**

127 / 8 = 15 Remainder: **7**

15 / 8 = 1 Remainder: **7**

1 / 8 = 0 Remainder: **1**

$(1020)_{10} = (1774)_8$

Example: **Decimal to Hexadecimal** $(1020)_{10} = ()_{16}$

1020 / 16 = 63 Remainder: **C**

63 / 16 = 3 Remainder: **F**

3 / 16 = 0 Remainder: **3**

$(1020)_{10} = (3FC)_8$

Conversion of Rational Numbers

<u>Converting binary fractions to rational numbers (decimal fractions)</u>

To convert binary fractions to their decimal equivalents, all we would need to do is multiply each bit with its power of 2 position.

Example: $(0.11001)_2 = ()_{10}$

$0.11001 = 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 0 * 2^{-4} + 1 * 2^{-5}$

$= 1 * 0.5 + 1 * 0.25 + 0 * 0.125 + 0 * 0.0625 + 1 * 0.03125$

$= 0.5 + 0.25 + 0.03125$

= <u>0.7812510</u>

Note: To convert octal and hexadecimal fractions to their decimal equivalents we follow the same steps we used above except we change the multiplier(Base) to 8 and 16 respectively.

<u>Converting rational numbers to Baes B fractions</u>

To convert rational numbers to their Base B fraction equivalent we will follow the following steps.

Step 1: Multiply the number after decimal point by B

Step 2: Take number before the decimal point from the result.

Step 3: repeat steps 1 and 2 until there is no decimal point to multiply

Example 1: $(0.15625)_{10} = ()_2$

Step i	N	Operation	$Next$	$z_{(-i)}$
1	0.15625	0.15625 * 2 = 0.3125	0.3125	0
2	0.3125	0.3125 * 2 = 0.625	0.625	0
3	0.625	0.625 * 2 = 1.25	0.25	1
4	0.25	0.25 * 2 = 0.5	0.5	0
5	0.5	0.5 · 2 = 1	None	1

$(0.15625)_{10} = (00101)_2$

Example 2: $(0.15625)_{10} = ()_8$

Step i	N	Operation	$Next$	$z_{(-i)}$
1	0.15625	0.15625 * 8 = 1.25	0. 25	1
2	0.25	0.25 * 8 = 2	None	2

$(0.15625)_{10} = (12)_8$

Example 3: $(0.15625)_{10} = ()_{16}$

Step i	N	Operation	$Next$	$z_{(-i)}$
1	0.15625	0.15625 * 16 = 2.5	0. 5	2
2	0.5	0.5 * 16 = 8	None	8

$(0.15625)_{10} = (28)_8$

Conversion of binary digits to octal and hexadecimal digits

To convert binary digits to their octal equivalent it's important to answer one question.

How many binary digits are equivalent to one octal digit? (1 octal bit = ? binary bits)

Octal => base 8 => 8=2^3 hence 1 octal digit is equivalent with 3 binary digits. (1 octal bit = 3 binary bits)

So all we need to do is group each 3 binary digits and convert to its decimal equivalent.

Converting binary digits to Hexadecimal is also done in the same manner. So we will answer the same kind of question we asked earlier, how many binary digits are equivalent to one hexadecimal digit?

Hexadecimal => base 16 => 16=2^4 hence 1 hexadecimal digit is equivalent with 4 binary digits.

(1 hex bit = 4 binary bits)

Example: $(101111010)_2$ = $\boxed{101}$ $\boxed{111}$ $\boxed{010}$ => 101 = 5, 111 = 7, 010 = 2

= $\underline{(5\ 7\ 2)_8}$

Example: (101111010)2 = $\boxed{0001}$ $\boxed{0111}$ $\boxed{1010}$

= $\underline{(1\ 7\ A)_{16}}$