Computer Number Systems

All computers – from large mainframes to hand-held micros – ultimately can do one thing: detect whether an electrical signal is "on" or "off". Computer programs in all high-level languages are converted by various pieces of systems software into sequences of bits (*B*inary dig*IT*s) which correspond to sequences of on/off (equivalently TRUE/FALSE or 1/0) signals. These bits must represent the operation and the address for each instruction. Binary digits (or bits) are also used to represent all forms of data including integers, floating point or decimal values, character strings, sound, and visual images. Proficiency in the binary number system is essential to understanding how a computer works.

Since binary numbers representing moderate values quickly become rather lengthy, bases eight (octal) and sixteen (hexadecimal) are frequently used as short-hand. Octal numbers group binary numbers in bunches of 3 digits and convert the triplet to a single digit between 0 and 7, inclusive. For example, $1001010110_2 = 001\ 001\ 010\ 110_2 = 1126_8$. This is because $8 = 2^3$ and the value of three bits $111 = 1 + 2 + 4 = 7$ using powers of 2. Hexadecimal numbers group binary numbers by fours, and convert the quadruplet to a single digit in the range 0, 1, 2 …, 9, A, B, C, D, E, F. The digits A through F have decimal values of 10 through 15 respectively. This is because $16 = 2^4$ and the value of four bits $1111 = 1 + 2 + 4 + 8 = 15$ using powers of 2. For example, $10110110100101_2 = 0010\ 1101\ 1010\ 0101_2 = 2DA5_{16}$. All of the basic rules of number theory apply to every base, but these three bases 2, 8, and 16 are uniquely suited for computer science.

Therefore, converting from any base to base 10 involves multiplying each digit by an increasing power of that base. For example, $457_8 = 7 \times 8^0 + 5 \times 8^1 + 4 \times 8^2 = 7 + 40 + 256 = 303_{10}$. Converting from base 10 to any other base involves finding how many times each decreasing power of that base can be divided evenly into the number and repeating the process with the remainder. For example, $500_{10} = 256 \times 1 + 16 \times 15 + 1 \times 4 = 1F4_{16}$. Another way to accomplish this us to repeatedly divide the number by the base as follows:

$500 \div 16 = 31r4$

$31 \div 16 = 1r15$ Therefore, reading the remainders from bottom to top give you $1F4_{16}$.

$1 \div 16 = 0r1$

Adding in bases other than 10 means that you must carry the value of that base and subtracting in bases other than 10 means that you must borrow the value of that base if necessary. For example, $\begin{array}{r} FEED_{16} \\ +9A3_{16} \\ \hline 10890_{16} \end{array}$ since D=13 and 13+3 = 16 so leave the 0 and carry the 16 as a 1. Then E=14 and A = 10 so 1+14+10 = 25 so leave the 9 and carry the 16 as a 1. E=14 so 1 + 14 + 9 = 24 so leave the 8 and carry the 16 as 1. Finally, F=15 so 1 + 15 = 16 so make that 10.

Subtracting in base 8 is as follows: $\begin{array}{r} 4572_8 \\ -756_8 \\ \hline 3614_8 \end{array}$. Borrow 1 = 8 from the 7 since 2 + 8 - 6 = 4. Therefore, 6 – 5 = 1. Then, borrow 1 = 8 from the 4 since 5 + 8 – 7 = 6. Then the last digit on the left is a 3.

References

Many pre-Algebra textbooks cover bases other than 10. From the computer science point of view, most books covering Assembly Language also cover binary, octal and hex number systems. The texts cited for the Boolean Algebra category cover computer number systems.

Sample Problems

Solve for *X*. $X_{16} = 3676_8$	One method of solution is to convert 3676_8 into base 10, and then convert that number into base 16 to yield the value of *X*. An easier solution, less prone to arithmetic mistakes, is to convert from octal (base 8) to hexadecimal (base 16) through the binary (base 2) representation of the number: $3676_8 = 011\ \ 110\ \ 111\ \ 110_8$ $= 0111\ \ 1011\ \ 1110_8$ $= 7BE_{16}$
Solve for *X*. $X_{16} = FEED_{16} - 6ACE_{16}$	The rightmost digit becomes *F*, because *1D-E=F*. Next, *D-C*=1 (the *E* becomes a *D* because we had to borrow from it to do the units' subtraction), *E-A*=4, and then *F*-6=9. Combining these results of each column, we get a final answer of $941F_{16}$.
In the ACSL computer, each "word" of memory contains 20 bits representing 3 pieces of information. The most significant 6 bits represent Field A; the next 11 bits, Field B; and the last 3 bits represent Field C. For example, the 20 bits comprising the "word" 18149_{16} has fields with values of 6_{16}, 29_{16} and 1_{16}. What is Field B in $E1B7D_{16}$? (Express your answer as a base 16 number.)	E 1 B 7 D = 1110 0001 1011 0111 1101 = 1110 00 01 1011 0111 1 101 Field B = 01 1011 0111 1 = 011 0110 1111 = 3 6 F_{16}
Which of the following has the least 1's in its binary equivalent? A. $FAD_{16} - ABE_{16}$ B. $5647_8 + 1543_8$ C. $101110_2 * 1010_2$ D. $2400_{10} / 5_{10}$	A. $FAD_{16} - ABE_{16} = 4EF_{16} = 10011101111_2$ (8 1's) B. $5647_8 + 1543_8 = 7412_8 = 111100001010_2$ (6 1's) C. $101110_2 * 1010_2 = 111001100_2$ (5 1's) D. $2400_{10} / 5_{10} = 480_{10} = 111100000_2$ (4 1's) Therefore, the answer is D.