CPTR 215 Assembly Language Programming I
Outline for Positional Number Systems

MOTIVATION: Positional number systems are important because you have to understand them in order to understand numbers in different bases. Base 2 (binary) is important because the fundamental principle by which computer circuits work involves two states (ON and OFF). Base 16 (hexadecimal) is a compromise. Hex numbers are easier for humans to deal with than binary (but not as easy as decimal would be). But hex numbers are very easy to convert to binary.

I. The key feature of any positional number system is that the value of a symbol depends on its position in the number. Thus, the number "686" means 6*100 + 8*10 + 6. The "6" in the hundreds position has a value of six hundred, but the other "6" has value of only six.

II. In base x, there are always x symbols. So in base 10 there are ten symbols: 0,1,2,3,4,5,6,7,8,9. In base 2 there are two symbols: 0 and 1. In hexadecimal there are 16 symbols: 0,1,2,3, 4,5,6,7,8,9,A,B,C,D,E,F. Of course, it doesn't matter which symbols are used so long as everyone agrees on the value of each symbol.

III. A subscript on a number is used to indicate its base. Thus, $100_{10} = 64_{16}$. In addition, in computer programming an "h" at the end usually indicates hexadecimal, "b" indicates binary, and no trailing symbol indicates decimal. 100h = 64.

IV. All conventional computer circuits work in base 2. So it is important to be comfortable working in base 2. First counting from zero to ten:

zero	0000	six	0110
one	0001	seven	0111
two	0010	eight	1000
three	0011	nine	1001
four	0100	ten	1010
five	0101		

Note that every new column starts with 0 and only increments to 1 when all the columns to its right have reached 1. Of course, leading zeros (zeros on the left) add nothing to the number, so 0011 is equal to 11. (This should be obvious since it also applies in base 10: 00456 = 456.)

V. The largest binary number that is n digits long is $2^n - 1$. If you include zero, there are always exactly 2^n binary numbers that are n digits long. A binary digit is called a bit. Eight bits is called a byte. Sixteen bits is a word.

VI. A K is exactly 1024. 64K is 65536.

VII. It is handy when working with binary numbers to define a function called the base 2 logarithm.

> The base 2 logarithm of x (lg x) is equal to the number that when 2 is raised to that power, yields x. For example, lg(65536) = 16 since $2^{16} = 65536$.

If your calculator does not have a lg function on it, it is easy to compute it using this formula:

$$lg(x) \equiv log(x) / log(2)$$

Here is a sample problem. How many bits does it take to store the number 2,358,619? Answer: log(2,358,619) / log(2) = 21.1695. Rounded up, this gives 22 bits.

> lg(x) = n is defined to be that x such that $2^n = x$.

VIII. Addition works the same in binary as it does in decimal. We start with an addition table.

+	0	1
0	0	1
1	1	10

Now a sample addition:

```
  1001110
+ 1100101
---------
 10110011
```

IX. Multiplication is similar.

*	0	1
0	0	0
0	0	1

```
   1011
 * 1101
------
   1011
  0000
 1011
1011
--------
10001111
```

X. Hexadecimal notation is convenient to manipulate and is easy to convert to binary notation because of a (not accidental) coincidence: Four binary digits correspond exactly to one hexadecimal digit.

Binary	Hex	Binary	Hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Memorize this table

Thus, conversion between binary and hex is trivial, just think of the binary number in groups of four bits (starting from the right). 11001010111010b = 11 0010 1011 1010b = 32BAh. (By the way, viewing binary numbers as groups of four digits is perfectly analogous to viewing decimal numbers as groups of three digits, separated by commas.)

This can be used to convert binary numbers to hex and vice versa.

XI. Converting between hexadecimal and decimal is a little harder. Here is one way to do it.

What is 25A6h in decimal? Use a calculator to do the following from left to right:

2 * 16 + 5 * 16 + 10 * 16 + 6 =

Or with an HP:

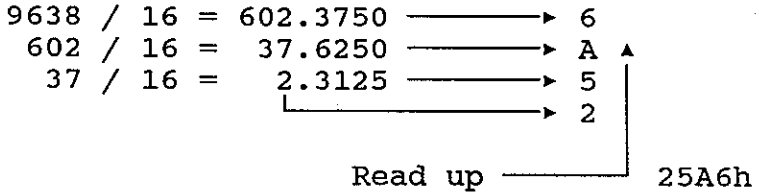
2 ENTER 16 * 5 + 16 * 10 + 16 * 6 +

(The answer is 9638.)

To convert the other way is a little harder. The following table of fractions is helpful:

0/16.....0.0000	8/16.....0.5000
1/16.....0.0625	9/16.....0.5625
2/16.....0.1250	10/16....0.6250
3/16.....0.1875	11/16....0.6875
4/16.....0.2500	12/16....0.7500
5/16.....0.3125	13/16....0.8125
6/16.....0.3750	14/16....0.8750
7/16.....0.4375	15/16....0.9375

What is 9638d in Hex? Use a calculator to do the following from left to right:



XII. Negative numbers are represented in four common forms.

A. Sign-Magnitude. The most-significant bit of the number is 1 if the number is negative. This is most like our customary decimal system, where the dash "-" represents negative numbers. In computer systems it is most frequently used to represent the mantissas of floating point numbers.

B. Ones' Complement. All the bits of a number are inverted to produce the negative number. This system is not used much any more. The problem is that there are two representations for zero.

C. Two's Complement. All the bits are inverted and then 1 is added. This is widely used for storing integers. Its primary advantage is that addition and subtraction of numbers with mixed signs can be done without need for adjustments.

D. Offset Form. All numbers have some offset added to them, typically 2^n or $2_n \pm 1$, where n is number of bits in the number. For example, for 8-bit representations, you might see Offset-128, or Offset-127 form. In offset-128, zero is 10000000. In offset-127, zero is 01111111.

Here is a chart showing the representations for all four forms. These are all 8-bit numbers.

Decimal	S-M	O-C	T-C	Offset 128
-128	N/A	N/A	10000000	00000000
-127	11111111	10000000	10000001	00000001
-73	11001001	10110110	10110111	00110111
-1	10000001	11111110	11111111	01111111
0	00000000	00000000	00000000	10000000
73	01001001	01001001	01001001	11001001
127	01111111	01111111	01111111	11111111

XIII. Note that Offset-128 is the same as two's complement with the most-significant bit inverted.