**Memory Addresses and Hexadecimal Numbers**

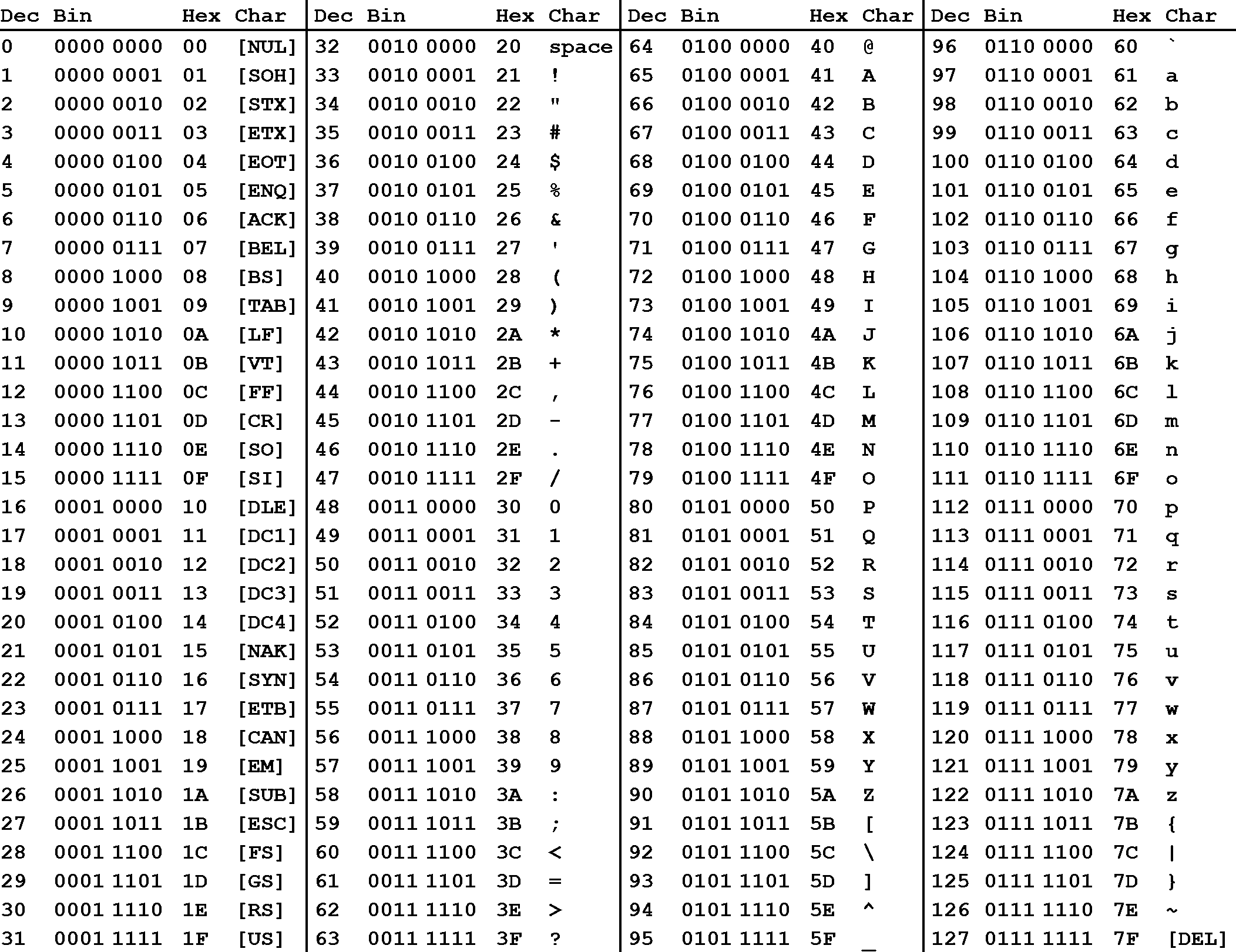
Understanding the number system used by computers to store and process data is essential for effective memory management, which is why we will start with an introduction into the binary and hexadecimal number systems and the structure of memory addresses.

Early attempts to invent an electronic computing device met with disappointing results as long as engineers and computer scientists tried to use the decimal system. One of the biggest problems was the low distinctiveness of the individual symbols in the presence of [**noise**](https://en.wikipedia.org/wiki/Noise_%28electronics%29). A 'symbol' in our alphabet might be a letter in the range A-Z while in our decimal system it might be a number in the range 0-9. The more symbols there are, the harder it can be to differentiate between them, especially when there is electrical interference. After many years of research, an early pioneer in computing, John Atanasoff, proposed to use a coding system that expressed numbers as sequences of only two digits: one by the presence of a charge and one by the absence of a charge. This numbering system is called Base 2 or binary and it is represented by the digits 0 and 1 (called 'bit') instead of 0-9 as with the decimal system. Differentiating between only two symbols, especially at high frequencies, was much easier and more robust than with 10 digits. In a way, the ones and zeroes of the binary system can be compared to Morse Code, which is also a very robust way to transmit information in the presence of much interference. This was one of the primary reasons why the binary system quickly became the standard for computing.

Inside each computer, all numbers, characters, commands and every imaginable type of information are represented in binary form. Over the years, many coding schemes and techniques were invented to manipulate these 0s and 1s effectively. One of the most widely used schemes is called ASCII (American Standard Code for Information Interchange), which lists the binary code for a set of 127 characters. The idea was to represent each letter with a sequence of binary numbers so that storing texts on in computer memory and on hard (or floppy) disks would be possible.

The film enthusiasts among you might know the scene in the hit movie "The Martian" with Mat Daemon, in which an ASCII table plays an important role in the rescue from Mars.

The following figure shows an ASCII table, where each character (rightmost column) is associated with an 8-digit binary number:



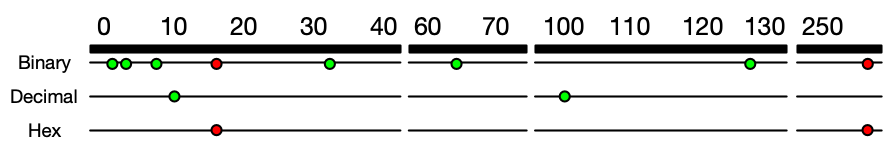
The letter U for example can be represented by the following sequence of bits: 0101 0101

In addition to the decimal number (column "Dec") and the binary number, the ASCII table provides a third number for each character (column "Hex"). According to the table above, the letter z is referenced by the decimal number 122, by the binary number 0111 1010 and by 7A. You have probably seen this type of notation before, which is called "hexadecimal". Hexadecimal (hex) numbers are used often in computer systems, e.g for displaying memory readouts - which is why we will look into this topic a little bit deeper. Instead of having a base of 2 (such as binary numbers) or a base of 10 (such as our conventional decimal numbers), hex numbers have a base of 16. The conversion between the different numbering systems is a straightforward operation and can be easily performed with any scientific calculator. More details on how to do this can e.g. be found [**here**](https://en.wikipedia.org/wiki/Computer_number_format).

There are several reasons why it is preferable to use hex numbers instead of binary numbers (which computers store at the lowest level), three of which are given below:

* **Readability**: It is significantly easier for a human to understand hex numbers as they resemble the decimal numbers we are used to. It is simply not intuitive to look at binary numbers and decide how big they are and how they relate to another binary number.
* **Information density**: A hex number with two digits can express any number from 0 to 255 (because 16^2 is 256). To do the same in the binary system, we would require 8 digits. This difference is even more pronounced as numbers get larger and thus harder to deal with.
* **Conversion into bytes**: Bytes are units of information consisting of 8 bits. Almost all computers are byte-addressed, meaning all memory is referenced by byte, instead of by bit. Therefore, using a counting system that can easily convert into bytes is an important requirement. We will shortly see why grouping bits into a byte plays a central role in understanding how computer memory works.

The reason why early computer scientists have decided to not use decimal numbers can also be seen in the figure below. In these days (before pocket calculators were widely available), programers had to interpret computer output in their head on a regular basis. For them, it was much easier and quicker to look at and interpret 7Einstead of 0111 1110. Ideally, they would have used the decimal system, but the conversion between base 2 and base 10 is much harder than between base 2 and base 16. Note in the figure that the decimal system's digit transitions never match those of the binary system. With the hexadecimal system, which is based on a multiple of 2, digit transitions match up each time, thus making it much easier to convert quickly between these numbering systems.



**[Each dot represents an increase in the number of digits required to express a number in different number systems. For base 2, this happens at 2, 4, 8, 32, 64, 128 and 256. The red dots indicate positions where several numbering systems align. Note that there are breaks in the number line to conserve space.](https://classroom.udacity.com/nanodegrees/nd213/parts/789a1625-9b09-4615-9210-ddbc12e9247b/modules/b2145e6c-f349-4071-b1a5-682cda25eba8/lessons/ec63b3b7-590d-43ef-9492-66f6f23d9988/concepts/e6830afc-c398-4af8-9221-f2675293f46f)**