Answers for Week 1 Questions

A1: Binary numbers are modulo 2. That means if we want to convert them to decimal numbers we are going to use powers of two. The decimal number is equal to the sum of binary digits d_n times their power of 2^n. Therefore; it is $1.2^0 + 1.2^1 + 1.2^2 + 1.2^3$ which is equal to 15.

A2: When we want to convert a decimal to binary, we apply the algorithm below:

- Divide the number by 2.
- Get the integer quotient for the next iteration.
- Get the remainder for the binary digit.
- Repeat the steps until the quotient is equal to 0.

29/2 - remainder: 1

14/2 - remainder: 0

7/2 - remainder: 1

3/2 - remainder: 1

1

When you write them from bottom to top, 11101.

A3: You can find the answer in the lecture notes. However, there is an easy way to remember. For AND gates, unless both inputs are 1, the answer is 0. For OR gate, if any of the inputs is 1, the output is 1. For XOR, if both inputs are different from each other (0,1 or 1,0) the output is 1.

A4: In order to reach a specific address, you don't need to follow all previous addresses. In other word, it doesn't have to be sequential. You can reach a specific address directly. Data can be read or modified in any order. For example another data storage media such as CD, data is accessed in a fixed sequence.

A5: That means the represented string is Hexadecimal. We put 0x in front of something to indicate that it is hexadecimal.

A6: CPU registers store memory addresses. Therefore the amount you can store depends on the CPU. If your CPU is 32-bit, that means you can only store 2^{32} memory addresses. That means a 32-bit CPU can only reach to that many

addresses, which in turn means that the capacity of the RAM which can be used is limited to 4GB. In 64-bit, it is much more than that.

A7: Cache is a smaller and faster memory used by the CPU. It is located closer to CPU compared to main memory. Main memory RAM is DRAM where Cache is SRAM, which is faster. It is used to reduce the time CPU spends to access data. It stores data which are most frequently used by CPU.

A8: There are two ways: *lossless* and *lossy*. For media, lossy data compression techniques are generally used.

A9: 0, 1, 1, 1

A10: Parity bit can only check for those bits which come before it. This has several problems. For example, if we use odd parity and there are even number of errors, we cannot detect it. By using checkbytes, we scatter parity bits in a certain way. Instead of giving each consecutive 8-bits a parity bit, we use a parity bit in a different order. A bit in the checkbyte is a parity bit. That parity bit will use not the consecutive 8-bits, but the n'th digits of the incoming bits.

Assume that we are sending the message *HELLO* in *ASCII*. It is 01001000 01100101 01101100 01101100 01101111. If we use odd parity, we will add these parity bits to the end of every letter: 1, 1, 1, 1, 1. If we use a checkbyte and choose the order to be the same with the incoming message: Now we are going to find parity bits for the following: 00000, 11111, 01111, 00000, 10111, 01111, 00001, 01001. Be careful that these are the combination of 1st digits, 2nd digits and so on. Therefore our checkbyte now becomes, 1, 0, 1, 1, 1, 1, 0, 1.