Hexadecimal Classwork

1. What are the first 4 place values in a base-16 number system? Write them as the slots would appear in a base-16 cashbox (with ***the smallest number on the right***):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

1. What are the first 13 place values in a base-2 number system? Write them as the slots would appear in a binary cashbox (with ***the smallest number on the right***):

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Compare these two sequences (Q1 and Q2). What do you notice?
2. Circle the vertical lines to the *right* of each equivalent place value in base-2 (Q2). This splits the place values up into groups of:
3. Focus on the first group only (to the right of your right-most circled line).
   1. How many different numbers can you represent with these digits in? Remember that zero counts as a number.
   2. What is the biggest number that you can represent with these digits in binary?
4. Think about a single digit in hexadecimal:
   1. How many different numbers can you represent with 1 digit in *hexadecimal*? Remember that zero counts as a number.
   2. What is the biggest number you can represent with 1 digit in *hexadecimal*?
   3. What is the *hexadecimal* digit that we use to represent the number in the previous question?
5. Take the *binary* number b10110101 and translate it to base-10 and then into *hexadecimal*? Show place values over your digits:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Binary | | | | | | | |  | Decimal | | |  | Hex | |
|  |  |  |  |  |  |  |  | → |  |  |  | → |  |  |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |  |  |  |  |  |  |  |

1. Look at *just* the right 4 bits of the binary from Q7.
   1. Rewrite them here.
   2. By itself, what is this number (Q8a) in base-10:
   3. What is the hex digit we use for that number (Q8b)?
2. Look at *just* the left 4 bits of the binary from Q7.
   1. Rewrite them here.
   2. By itself, what is this number (Q9a) in base-10.
   3. What is the hex digit we use for that number (Q9b)?
3. How do your answers to Q8c and Q9c relate to the *hex* of the original number (from Q7)? Why is this interesting?
4. Let’s see if that relationship holds with some other examples. Try b101011. Redo Q7 – Q10 with this number. You’ll notice that this number doesn’t break up into two 4-digit chunks like the last one, but you can always add leading zeros to “pad” the number without changing its value. This means that b101011 is the same as b00101011 which *can* be broken into two 4-digit chunks: 0010 and 1011. Show your work here…

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Binary | | | | | | | |  | Decimal | | |  | Hex | |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | → | 100 | 10 | 1 | → | 16 | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

First 4 bits:

Last 4 bits:

1. So, the 4-digit chunks match up. What’s the big deal? (rhetorical question) Let’s try a bigger number: b010001101100. This is much harder to translate to base-10. Do it anyway (you can use a calculator but show your work).
2. See how painful that was. Now translate that base-10 number into hexadecimal (you can use a calculator but show your work).
3. That was also not especially easy / fun. Now let’s try the “trick” that we discovered and see how much easier it is. Break the original number up into three, 4-digit binary chunks and translate them *individually* into hexadecimal digits. You could calculate it or just look them up in your table above:

1st chunk of 4: in hex:

2nd chunk of 4: in hex:

3rd chunk of 4: in hex:

*It should be the same answer you got for Q13. But it’s a lot easier this way…*

1. The same works for translating the other way (from hexadecimal to binary). What’s E509 in binary? The E is in the 4096’s place. That’s gross, but fortunately, we don’t have to deal with it: Translate each hex digit *individually* (or just look it up in your table) into a 4-bit chunk:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | | | | 5 | | | | 0 | | | | 9 | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Note that in hex, 0 has to translate to 4 binary zeros even though zero (by itself) can be represented with just 1 zero. This is because the corresponding hexadecimal zero has to “hold” the equivalent of 4 binary places. Use Google to check your answer by searching something like “0**x**E509 in binary”.

1. Have each person in the group demonstrate translating at least one 4-digit hex number into binary in the tables below. Use Google to check yourself by searching something like “0**x**4a in binary”.

|  |  |
| --- | --- |
| Hex Number | Binary Number |
|  |  |
|  |  |
|  |  |
|  |  |

1. Have each person in the group demonstrate translating at least one 16-digit binary number into hex in the tables below. Use Google to check yourself by searching something like “0**b**10110101 in hex”.

|  |  |
| --- | --- |
| Binary Number | Hex Number |
|  |  |
|  |  |
|  |  |
|  |  |

1. As a group, log onto a single computer, **turn off the sound**, open up a browser, and go to flippybitandtheattackofthehexadecimalsfrombase16.com . Have each person play a round.
   1. Explain how the game demonstrates the techniques that you learned within the worksheet.