**Score: \_\_\_\_\_**

**DA1 – Unsigned Binary Numbers**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

Today’s class introduced unsigned binary representation, drew parallels between binary representation and the decimal representation we are familiar with and demonstrated how to convert between the two. In today’s activities you will gain practice with unsigned binary representation, explore its limits and see how those limits manifest in programs in some real computing systems.

**Basic Terminology:**

Just to ensure that you have a good handle on the terminology that was introduced, this section asks some basic direct questions that require you to know the terminology.

🔑 1. Use the following binary string below to complete the table below.

0011 0100 1010 0000 0110 0100 0011 1101

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | How many bits are in this string? |  |  |
|  | The bit in the most significant position in this string? |  |  |
|  | How many bytes are in this string? |  |  |
|  | The bit in the least significant position in this string? |  |  |
|  | How many nibbles are in this string? |  |  |
|  | How many words are in this string? |  |  |
|  | How many half-words are in this string? |  |  |
|  |  |  |  |

**Binary/Decimal Conversions:**

There are many calculators and web sites that will perform binary/decimal conversions without you having to understand them at all. However, our goal here is not just to do the conversion but to understand how decimal (base 10) numbers are represented in binary (base 2). **To reinforce that purpose, and to convince me that you understand the material, please show all of your work using a format similar to what was used in the class slides or in one of the videos linked below. Correct answers that do not show adequate work will not receive full credit.** However, you should feel free to use or calculators or web sites to check your work and generate additional practice problems.

The following are not required viewing, but if you would like to see some additional worked examples of unsigned binary conversions, or if you get stuck the following may help:

* Abigale Bornstein demonstrates conversions in her videos:
	+ *Binary: converting decimal to binary (part 1 of 2)*
		- <https://www.youtube.com/watch?v=qWxiXU02ZQM> (5:00)
	+ *Binary: converting binary to decimal (part 2 of 2).*
		- <https://www.youtube.com/watch?v=UUqtjb8WEUs> (3:23)
* Keith Maycock converts both to and from binary in his video *Binary to Decimal Conversion and back again!* Note that he uses a different algorithm for converting from decimal to binary than I did. I find it less intuitive, but some find it easier to apply. Feel free to use his approach if it works for you.
	+ <https://www.youtube.com/watch?v=ZR0pC6_iW5Q> (3:14)

🔑 2. Find the base 10 value for the following 8-bit unsigned binary value: 0110 10112. Show sufficient work to demonstrate how you arrived at your answer.

🔑 3. Find the 8-bit unsigned binary representation for the following decimal number: 15010. Show sufficient work to demonstrate how you arrived at your answer.

🔑 4. Find the 8-bit unsigned binary representation for the following decimal number: 11710. Show sufficient work to demonstrate how you arrived at your answer.

**Binary Addition:**

As with binary/decimal conversions there are tools that will perform binary addition for you. But again, here we are more concerned with understanding the process than with obtaining the answer. Thus, again **I am asking that you show sufficent work including indicating all carries that occur. Even correct answers that do not show adequate work may be required to revise and resubmit.**

It is not required viewing, but if you would like to see additional examples of how to do binary additions you might check out John the YouDoober’s video *How To Do Binary Addition (The Easy Way)*.

* <https://www.youtube.com/watch?v=ypqYoFbPfTk> (3:47)

🔑 5. Perform the following additions of 8-bit unsigned binary numbers. Be sure to show all carries that happen.

 0101 1001 0111 1010

 + 1001 1011 + 0011 1001

🏆 6. Perform the following subtraction of 8-bit unsigned binary numbers. Be sure to show any borrows that you perform.

 1101 0011

 - 0110 1010

**Exploring Unsigned Binary:**

When a computer is constructed the hardware is designed to perform its operations on numbers that are represented using a specific number of bits (e.g. 8, 16, 32 or 64). Once the number of bits is fixed, the range of values that can be represented is also fixed. This is no different than with base 10 numbers. For example, if you are restricted to using 3 digits, then you can only represent unsigned numbers from 0 to 999.

7. In this question you will determine the range of values that can be represented in unsigned binary using a fixed number of bits.

a. What is the smallest base 10 number that can be represented in unsigned binary?

b. What is the 8-bit unsigned binary representation for the number you gave in a?

c. Fill in the remaining rows of the following table to get a feel for the largest base 10 value that can be represented in unsigned binary using a given number of bits.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **Numberof Bits (n)** | **Largest UnsignedBinary Value** | **Base 10Value** | **2n** |  |
|  | **1** | **12** | **110** | **21 = 2** |  |
|  | **2** | **112** | **310** | **22 = 4** |  |
|  | **4** |  |  |  |  |
|  | **6** |  |  |  |  |
|  | **8** |  |  |  |  |
|  |  |  |  |  |  |

d. Look for a pattern in the table above and use what you find to write a formula that gives the largest decimal value that can be represented in unsigned binary representation using n bits. Be sure to check your formula against the values in your table above.

UBMax(n) = ???

e. Using your formula from d, fill in the following table. To help you get a real feel for the maximum base 10 values that can be stored in a number of bits, do not use exponents. Instead, write out the full base 10 values.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Numberof Bits (n)** | **Largest Base 10 Value Representable in** **n-bit Unsigned Binary** |  |
|  | **8** |  |  |
|  | **16** |  |  |
|  | **32** |  |  |
|  | **64** |  |  |
|  |  |  |  |

🏆 f. The C programming language allow you to declare variables using the unsigned keyword. When this keyword is used in a declaration, as shown in the table below, it instructs the computer to use unsigned binary representation to store the value of the variable. Use your favorite search engine and your answer to part d to fill in the table below indicating the number of bits and bytes used to represent each of the unsigned data types in the C programming language and their maximum value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **C Data Type** | **Number of Bytes** | **Number of Bits** | **Largest possible Value** **in Base 10** |  |
|  | unsigned char |  |  |  |  |
|  | unsigned short |  |  |  |  |
|  | unsigned int |  |  |  |  |
|  | unsigned long |  |  |  |  |
|  |  |  |  |  |  |

It may seem weird to think about the base 10 value of a char (e.g. for unsigned char). This data type could equally well have been called an unsigned byte. But the creators of C used char because at that time each character (e.g. letter) was represented by 8 bits. We’ll learn more about that in the next class! It is also worth noting that Java does not allow you to use unsigned data types at all. All of its integer type variables use a signed representation that we’ll learn about in a few days.

**Unsigned Arithmetic Overflow:**

You now know that computers operate on a fixed number of bits at a time and that programming languages like C use a specific number of bits for each data type. You now also know that the range of decimal values that can be represented using unsigned binary is determined by the number of bits used. So, it should not come as too much of a surprise that when performing computations like addition, subtraction and multiplication it is possible to have results that are out of the representable range for a data type. Formally, we call these occurrences *arithmetic overflow*.

For example, consider the following 8-bit unsigned binary addition:

 11 11

 0101 00012

 + 1100 00112

 0001 0100

Notice that when the most significant bits are added together there is a carry generated (shown in red). If this operation were being performed on a machine built to use 8-bit unsigned binary numbers, the result would also have to be 8 bits. Thus, there would be no place for this extra carry bit to be stored. It would simply be discarded. When this happens to unsigned binary values, the result of the addition will be incorrect.

🔑 8. Let’s assume we have data type named unsigned nibble that uses 4-bit unsinged binary representation. Note: I made up the unsigned nibble data type for this problem.

a. Perform each of the following additions on unsigned nibbles (i.e. 4-bit unsigned binary values). Be sure to show all of the carries that occur.

 i. ii. iii. iv.

 0101 0110 1001 1110

 + 11002 + 11002 + 01002 + 10112

b. Several of the additions above resulted in overflow. Which ones? How can you tell?

c. Consider now, part a.1 above in which overflow occurred.

i. Give the base 10 values of the unsigned binary numbers that were being added and their correct sum.

ii. What is the maximum base 10 value that can be represented using 4-bit unsigned binary representation? Hint: See your answer earlier in this assignment!

iii. Compare your answers to part i and ii, and explain in a few sentences why arithmetic overflow occurred in this addition.

9. Consider the short C language program at

[https://repl.it/@braughtg/NonInfiniteLoopByte#main.c](https://repl.it/%40braughtg/NonInfiniteLoopByte#main.c)

Click “Files” and “main.c” to see the code. This program will look similar to the one we saw back in the first class, except it it is in C and not in Java. At first glance this program looks like an infinite loop. But you know better, and we are now in a position to understand why it is not!

a. The data type of the variable i in this program is unsigned char. In the C language, variables of type unsigned char use 8-bit unsigned binary representation.

What are the smallest and largest values that can be stored in an unsigned char? Hint: Check question #7e (or f).

b. Will the program enter the while loop on line 8? Why?

c. What condition must be true of the value of i in order for the program to exit the while loop?

d. The statement in the body of the while loop adds 1 to the value of i each time the loop iterates. Eventually the value of i will reach the maximum value for its data type. Give the unsigned binary representation of the maximum value of an unsigned char.

e. The next time through the while loop, another 1 will be added to i. Using 8-bit unsigned binary, write out and complete the addition problem for adding 1 to the maximum value from part d. Use a format similar to question #5 and be sure to show all carries that occur.

f. What is the base 10 value of the 8-bit unsigned result you found in part e? Remember that any carry out from the most significant bit will be discarded because we are working 8-bit unsigned binary.

g. Now, using what you learned in parts a-f, write a few sentences explaining in some detail why the while loop in this program is not an infinite loop. Note: It is not sufficient to say “Arithmetic overflow occurred.” You must explain in enough detail to convince me that you understand.

🏆 10. This question explores the implications of arithmetic overflow a little further. Consider the short C language program at [https://repl.it/@braughtg/UnsignedByte#main.c](https://repl.it/%40braughtg/UnsignedByte#main.c). Give an explanation for the behavior of this program. Be sure to include what it seems like the output should be, what the output actually is and a detailed explanation using unsigned binary values for why that output is correct given the datatype being used.

**Another Binary Representation for Unsigned Whole Numbers:**

Binary Coded Decimal (BCD) is another way of representing unsigned whole numbers using just 1’s and 0’s. This page <https://www.computerhope.com/jargon/b/bcd.htm> provides a short introduction to BCD with an example or two. Read that page and then complete the following questions.

🏆 11. What is the BCD representation of the following base 10 number: 572110

🏆 12. The clock below uses BCD to represent the time in HH:MM:SS format.

****

What time is it?

Note: If after seeing that you just have to have one of these clocks for yourself… <https://www.truegether.com/listing.html?id=USER.205a8c99-58d6-452c-9232-9a824b72077d>

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.