

Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible and *show your work*. Write your name at the top of each page. Due at the *beginning of class*. Total points: 58.

1. [12 Points] **Basic Conversions.**

- (a) Convert the binary number 01110000 to decimal.

**Answer:**  $0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 64 + 32 + 16 = 112$

- (b) Convert the decimal number 42 to an 8-bit unsigned binary representation.

**Answer:** We compute the binary via repeated division. We divide the base-10 number by 2 and extract the remainders, which will be a binary digit in our final answer. We repeat this process until the number goes to 0. The first remainder we compute will be the LSD (Least Significant Digit, which goes on the right), and the last remainder will be the MSD (Most Significant Digit). Here is the whole process:

$$\begin{array}{l} 42/2 = 21 \quad \text{remainder} = 0 \\ 21/2 = 10 \quad \text{remainder} = 1 \\ 10/2 = 5 \quad \text{remainder} = 0 \\ 5/2 = 2 \quad \text{remainder} = 1 \\ 2/2 = 1 \quad \text{remainder} = 0 \\ 1/2 = 0 \quad \text{remainder} = 1 \\ 0/2 = 0 \quad \text{remainder} = 0 \end{array}$$

Therefore, the decimal number 42 is represented as 101010. We pad this with one zero to form an 8-bit unsigned binary number, resulting in 0101010.

- (c) Convert the 8-bit 2's complement binary number 10110110 to decimal.

**Answer:** That the high-order bit is 1 implies that this number is negative, so we must take its 2's complement to compute its absolute value. We do this by flipping each bit (*i.e.*, taking the 1's complement) and adding 1:  $01001001 + 1 = 01001010$ . The decimal value of this binary number is computed as follows:

$$0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 64 + 8 + 2 = 74$$

Since the original number was negative, we must negate this number, so the final answer is  $-74$ .

- (d) Convert the decimal number -117 to an 8-bit 2's complement binary representation.

**Answer:** Here we use a slight variation of the technique used in 1(b). Instead of actually computing remainders, we simply observe whether the number at each step is even or odd. You should feel free to use whatever approach is easier for you.

We cannot directly convert  $-117_{10}$  to binary. First, we must convert  $117_{10}$  to binary.

$$117 = a_7 \times 2^7 + a_6 \times 2^6 + a_5 \times 2^5 + a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$$

Since 117 is odd, we know  $a_0 = 1$ . We subtract 1 from both sides of the equation and divide by 2 yielding the following.

$$58 = a_7 \times 2^6 + a_6 \times 2^5 + a_5 \times 2^4 + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0$$

Since 58 is even, we know that  $a_1 = 0$ . We divide by 2 (without first subtracting 1).

$$29 = a_7 \times 2^5 + a_6 \times 2^4 + a_5 \times 2^3 + a_4 \times 2^2 + a_3 \times 2^1 + a_2 \times 2^0$$

Since 29 is odd, we know that  $a_2 = 1$ . We subtract 1 from both sides and divide by 2.

$$14 = a_7 \times 2^4 + a_6 \times 2^3 + a_5 \times 2^2 + a_4 \times 2^1 + a_3 \times 2^0$$

Since 14 is even, we know that  $a_3 = 0$ . We divide by 2 (without first subtracting 1).

$$7 = a_7 \times 2^3 + a_6 \times 2^2 + a_5 \times 2^1 + a_4 \times 2^0$$

Since 7 is odd, we know that  $a_4 = 1$ . We subtract 1 from both sides and divide by 2.

$$3 = a_7 \times 2^2 + a_6 \times 2^1 + a_5 \times 2^0$$

Since 3 is odd, we know that  $a_5 = 1$ . We subtract 1 from both sides and divide by 2.

$$1 = a_7 \times 2^1 + a_6 \times 2^0$$

Since 1 is odd, we know that  $a_6 = 1$ . We subtract 1 from both sides and divide by 2.

$$0 = a_7 \times 2^0$$

Clearly,  $a_7 = 0$ . Putting all these bits together we now know that  $117_{10} = 01110101_2$ . But we want  $-117$ , not  $117$ , so we must find the 2's complement representation of the negative number of the same magnitude. To do this we complement  $01110101_2$  (producing  $10001010_2$ ), to which we add 1. Now we know that  $-117_{10} = 10001011_2$ .

- (e) Convert the 8-bit unsigned binary number 10110010 to hexadecimal.

**Answer:** We could first convert this to decimal, then use the technique used in 1(b) (repeatedly dividing by 16 instead of 2), but it is much easier to observe that each group of four binary digits corresponds to one hexadecimal digit. We know that  $1011_2 = B_{16}$  and  $0010_2 = 2_{16}$ , so  $1011\ 0110_2 = B2_{16}$ .

- (f) Convert the unsigned hexadecimal number BEAD to unsigned 16-bit binary.

**Answer:** Again, we observe that each hexadecimal digit corresponds to 4 binary digits. We know that  $B_{16} = 1011_2$ ,  $E_{16} = 1110_2$ ,  $A_{16} = 1010_2$ , and  $D_{16} = 1101_2$ , so  $BEAD_{16} = 1011\ 1110\ 1010\ 1101_2$ .

2. [12 Points] **Binary Arithmetic and Logical Operations.** Let  $A = 00100110$  and  $B = 11010011$  be 2's complement integers. Compute the following, giving your answers in *both* 8-bit 2's complement and decimal. Use a fixed width of 8 bits (*i.e.*, your answers must be 8 bits). As always, show your work.

- (a)  $A + B$

**Answer:**

$$\begin{array}{r} A = 00100110 \quad (38) \\ + B = 11010011 \quad (-45) \\ \hline 11111001 \quad (-7) \end{array}$$

- (b)  $A \text{ OR } B$

**Answer:**

$$\begin{array}{r}
 A = 00100110 \\
 OR B = 11010011 \\
 \hline
 11110111 \quad (-9)
 \end{array}$$

(c)  $A AND B$

**Answer:**

$$\begin{array}{r}
 A = 00100110 \\
 AND B = 11010011 \\
 \hline
 00000010 \quad (2)
 \end{array}$$

(d)  $B - A$

**Answer:**

We must compute  $-A$  (i.e., 2's complement) and add it to  $B$ . The 2's complement of  $A$  is 00100110 and its 2's complement is 11011010. Now we add.

$$\begin{array}{r}
 B = 11010011 \quad (-45) \\
 + -A = 11011010 \quad (-38) \\
 \hline
 10101101 \quad (-83)
 \end{array}$$

(e)  $A - B$

**Answer:** We compute  $-B$  and add it to  $A$ .  $-B = 00101101$ .

$$\begin{array}{r}
 A = 00100110 \quad (38) \\
 + -B = 00101101 \quad (45) \\
 \hline
 01010011 \quad (83)
 \end{array}$$

(f)  $A + \bar{B} + 1$

**Answer:**

$$\begin{array}{r}
 A = 00011001 \\
 \bar{B} = 00101101 \\
 + \quad 00000001 \\
 \hline
 01010011 \quad (83)
 \end{array}$$

Not surprisingly (since  $-B$  is the complement of  $B$  plus one), this is the same as that which we computed in 2(e).

3. [7 Points] **Logical Operations.** Complete the following truth tables.

(a)

A	$\bar{A}$	$A \text{ OR } \bar{A}$	$A \text{ AND } \bar{A}$
0	1	1	0
1	0	1	0

(b)

A	B	C	$(A \text{ OR } B) \text{ AND } C$	$(A \text{ AND } C) \text{ OR } (B \text{ AND } C)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(c)

A	B	$(\bar{A} \text{ AND } \bar{B})$	$\overline{(A \text{ OR } B)}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

4. [8 Points] **Floating Point.**

- (a) Give an example of a number that has a 32-bit floating point representation (as in Figure 2.2 in the text-book) and cannot be represented as a 32-bit 2's complement integer. Explain why this number cannot be represented as an integer.

**Answer:** There are two possible answers:

- i. Any number that includes a fraction cannot be represented as a 32-bit 2's complement integer. A possible example is  $18.5_{10}$ . The 32 bit representation is 0 1000011 0010100000000000000000.
  - ii. Also, very large numbers cannot be represented in 32 bits. For example, 0 1111110 000000000000000000000000 represents the number  $1.0 \times 2^{127}$  which is far outside the range  $(-2^{31} \text{ to } 2^{31} - 1)$  of a 32-bit 2's complement number.
- (b) Give an example of a number that can be represented as a 32-bit 2's complement integer but cannot be represented exactly as a 32-bit floating point. Explain why this number cannot be represented as a floating point.

**Answer:** We need to find an integer that requires more bits of precision than are available in 32-bit floating point. Here's an obvious example of such an integer.

$$01111111111111111111111111111111_2 = 2147483647_{10}$$

It cannot be represented as a floating point because of the limited fractional bits (23 versus 31). Though the floating point helps in increasing the range of numbers that can be represented, it reduces the precision of the numbers that can be represented.

5. [8 Points] **Limitations of Fixed-Width Arithmetic.** Consider the following 8-bit 2's complement numbers:  $A = 01111111$ ,  $B = 00000101$ , and  $C = 10001011$ . Assume that only 8 bits are available to represent values. Show your work.

- (a) Evaluate  $A + B$ . Give your answer as an 8-bit 2's complement number. Convert this number to decimal. Why doesn't this represent the sum of  $A$  and  $B$ ?

**Answer:** If we add  $A$  (127) and  $B$  (5) and only consider the low-order 8 bits, the result is 10000100, which is  $-124$  in decimal. What happened? The true sum ( $127+5=132$ ) exceeds the range of values that may be expressed in an 8-bit 2's complement representation. As a result, the addition resulted in overflow.

- (b) Evaluate  $C - A$ . Give your answer as an 8-bit 2's complement number. Convert this number to decimal. Why doesn't this represent the difference of  $C$  and  $A$ ?

**Answer:** First, we compute  $-A$  ( $-127$ ) and add it to  $C$  ( $-117$ ). The result (ignoring all but the low-order 8 bits) is 00001100, which in decimal is 12. The true sum ( $-117 + -127 = -244$ ) exceeds the range of values that may be expressed in an 8-bit 2's complement representation. Unlike the previous problem, it is not too large, rather, it is too small. The addition of these two numbers resulted in overflow.

6. [10 Points] **Multiple Interpretations of Bits.** Consider the following sequence of 16 bits: 1100 0110 0011 0001. These bits can be interpreted in many different ways.

- (a) If we interpret these bits as a 16-bit unsigned binary integer, what is the decimal value represented by the bit sequence?

**Answer:**  $1 \times 2^{15} + 1 \times 2^{14} + 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^0 = 32,768 + 16,384 + 1,024 + 512 + 32 + 16 + 1 = 50,737$

- (b) If we interpret these bits as a 16-bit 2's complement integer, what is the decimal value represented by the bit sequence?

**Answer:** That the high-order bit is 1 implies that it is negative. We will take the 2's complement to compute the absolute value. The 2's complement is 0011100111001111. The decimal value of this is computed as follows.

$1 \times 2^{13} + 1 \times 2^{12} + 1 \times 2^{11} + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8192 + 4096 + 2048 + 256 + 128 + 64 + 8 + 4 + 2 + 1 = 14799_{10}$

The original (negative) number is  $-14799$ .

- (c) If we interpret the low-order 8 bits as an ASCII character (see Appendix E in your textbook), what is this character?

**Answer:** The lower-order 8 bits have the value (decimal) value of 49. We look this up in Table E.2 (p. 616), and we see that the ASCII code 49 refers to the character "1".

- (d) If we interpret these bits as a floating point number, what is the decimal value represented by the bit sequence. Assume that the floating point representation devotes 1 bit to the sign, 5 bits to the exponent, and 10 bits to the fraction (similar Figure 2.2 in your textbook). Give the answer in the following form:  $A \times 2^B$ , where  $A$  and  $B$  are *decimal* numbers.

**Answer:** This is  $(-1)^1 \times 1.1000110001^{17-15} = (-1) \times 1.5478515625 \times 2^2 = -6.19140625$ .

- (e) If we interpret these bits as an Red-Green-Blue (RGB) color, what is the color represented by the bit sequence? Assume the high-order bit is always 1, the next 5 bits represent red, the next 5 bits represent green, and the low-order 5 bits represent blue.

**Answer:** If we partition the bits as described above we see that red, green, and blue all have the same value ( $10001_2 = 17_{10}$ ). Since they are all the same, this value represents a color describing a shade of gray.