

## CSC 205: Homework 1

### Connor Baker, January 2017

**Instructions** Solve all problems below, showing ALL work (calculations). Simply writing an answer will result in ZERO credit for the problem. You are encouraged to practice on your own to be certain you can properly calculate the number base conversions and math correctly before attempting these problems.

**Problem 1** Convert  $1282_{10}$  to binary.

*Work* Start by picking an arbitrarily large power of two (not to exceed  $1/2$  the number we are to convert). I'll pick  $2^7$ , which is 128.

Divide 1282 by 128. This yields 10R2 (a quotient of ten with a remainder of two). Since for the choice of divisor I picked  $2^7$ , the binary representation that I'll be appending later won't just be two in base two (because two was our remainder): it'll be two in base two with seven place values (the number of place values must match the power of the base we choose to make our divisor).

Divide 10 by 128. This yields 0R10. Again, since for the choice of divisor I picked  $2^7$ , the binary representation that I'll be appending will be ten in base two with seven place values.

Just like the standard algorithm for change of base, we take the last remainder in the new base and it becomes the leftmost part of the new representation. As such, we'll concatenate  $10_10$  and  $2_10$  after converting to get our binary representation of  $1282_{10}$ .

$10_{10}$  to seven place values in base two is  $0001010_2$ . Likewise,  $2_{10}$  to seven place values in base two is  $0000010_2$ . The concatenation is  $00010100000010_2$ . Leading zeros hold no value, so this is equivalent to  $10100000010_2$ .

**Check** The number  $10100000010_2$  can be written as  $1 \times 2^{10} + 1 \times 2^8 + 1 \times 2^1$ , which is  $1282_{10}$ .

**Problem 2** Convert  $359_{10}$  to octal.

*Work* Start by picking an arbitrarily large power of eight (not to exceed  $1/2$  the number we are to convert). I'll pick  $8^2$ , which is 64.

Divide 359 by 64. This yields 5R39. Since for the choice of divisor I picked  $8^2$ , the octal representation that I'll be appending later won't just be 39 in base eight: it'll be 39 in base eight with two place values (the number of place values must match the power of the base we choose to make our divisor).

It should be noted that one pitfall of this method is that although it cuts the number of computations needed for larger numbers by using higher powers of the base, it also draws on one's ability to recognize smaller numbers in that base. For example, by choosing  $8^2$  as the divisor instead of  $8^1$ , we do roughly half as many calculations. However, we now how to find  $39_{10}$  in octal with two place values, where as if we had chosen  $8^1$ , the remainder would certainly have been under eight (and which makes sense, not least of all because we would be forced to write the remainder with one place value, since the number of place values must match the power of the base).

Divide 5 by 64. This yields 0R5. Again, since for the choice of divisor I picked  $8^2$ , the octal representation that I'll be appending will be five in base eight with two place values.

$5_{10}$  to two place values in octal is  $05_8$ . Likewise,  $39_{10}$  to two place values in octal is  $47_8$ . The concatenation is  $0547_8$ . Again, leading zeros hold no value, so this is equivalent to  $547_8$ .

**Check** The number  $547_8$  can be written as  $5 \times 8^2 + 4 \times 8^1 + 7 \times 8^0$ , which is  $359_{10}$ .

**Problem 3** Convert  $8191_{10}$  to hexadecimal.

*Work* Start by picking an arbitrarily large power of eight (not to exceed  $1/2$  the number we are to convert). I'll pick  $16^2$ , which is 256.

Divide 8191 by 256. This yields 31R255. Since for the choice of divisor I picked  $16^2$ , the hexadecimal representation that I'll be appending later won't just be 255 in hexadecimal: it'll be 255 in hexadecimal with two place values (the number of place values must match the power of the base we choose to make our divisor).

Divide 31 by 256. This yields 0R31. Again, since for the choice of divisor I picked  $16^2$ , the hexadecimal representation that I'll be appending will be 31 in hexadecimal with two place values.

$31_{10}$  to two place values in hexadecimal is  $1F_{16}$ . Likewise,  $255_{10}$  to two place values in hexadecimal is  $FF_{16}$ . The concatenation is  $1FFF_{16}$ .

**Check** The number  $1FFF_{16}$  can be written as  $1 \times 16^3 + 15 \times 16^2 + 15 \times 16^1 + 15 \times 16^0$ , which is  $8191_{10}$ .

**Problem 4** Convert  $2E0C_{16}$  to decimal.

*Work* Writing this problem in exponential notation is a good start:  $2E0C_{16} = 2 \times 16^3 + 14 \times 16^2 + 12 \times 16^0 = 8192 + 3584 + 12 = 11788_{10}$ .

**Problem 5** Convert  $561_8$  to decimal.

*Work* Writing this problem in exponential notation is a good start:  $561_8 = 5 \times 8^2 + 6 \times 8^1 + 1 \times 8^0 = 320 + 48 + 1 = 369_{10}$ .

**Problem 6** Convert  $1110110_2$  to decimal.

*Work* Writing this problem in exponential notation is a good start:  $1110110_2 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 64 + 32 + 16 + 4 + 2 = 118_{10}$ .

**Problem 7** Convert  $67EA_{16}$  to binary DIRECTLY (no intermediate base).

*Work* We can convert from hexadecimal to binary very easily: by splitting the number at each place value, converting the numbers to binary with four place values (since hexadecimal numbers can be represented within four place values in binary – if we were converting to octal, it would be three), and concatenating the result.

$6_{16}$   $7_{16}$   $E_{16}$   $A_{16}$  becomes  $0110_2$   $0111_2$   $1110_2$   $1010_2$ , which when concatenated and the lead zero(s) chopped off becomes  $11001111101010_2$ .

**Problem 8** Convert  $6204_8$  from octal to binary DIRECTLY (no intermediate base).

*Work* We can convert from octal to binary very easily: by splitting the number at each place value, converting the numbers to binary with three place values (since hexadecimal numbers can be represented within three place values in binary), and concatenating the result.

$6_8$   $2_8$   $0_8$   $4_8$  becomes  $110_2$   $010_2$   $000_2$   $100_2$ , which when concatenated becomes  $110010000100_2$ .

**Problem 9** Convert  $111011001100111110101_2$  to hexadecimal DIRECTLY (no intermediate base).

*Work* It's easiest to do this by grouping the binary in chunks of four from right to left.

So we have  $1_2 1101_2 1001_2 1001_2 1111_2 0101_2$ . By evaluating each chunk separately, converting it to hexadecimal, and concatenating the results, we will effectively have converted the original binary to hexadecimal.

Before concatenation, we have  $1_{16} D_{16} 9_{16} 9_{16} F_{16} 5_{16}$ . After concatenation, we find that the original number in hexadecimal is  $1D99F5_{16}$ .

**Problem 10** Convert  $100110111101_2$  to octal DIRECTLY (no intermediate base).

*Work* It's easiest to do this by grouping the binary in chunks of three from right to left.

So we have  $100_2 110_2 111_2 101_2$ . By evaluating each chunk separately, converting it to octal, and concatenating the results, we will effectively have converted the original binary to octal.

Before concatenation, we have  $4_8 6_8 7_8 5_8$ . After concatenation, we find that the original number in octal is  $4675_8$ .

**Problem 11** Represent  $-160_{10}$  in three notations: sign magnitude, 1's complement, and 2's complement.

*Work* The binary representation of  $160_{10}$  is  $10100000_2$ .

*Sign Magnitude* The sign magnitude representation of  $-160_{10}$  is  $110100000_2$  (simply appending a one to the front in the place of the sign bit to indicate that the number is negative).

*1's Complement* The 1's complement of  $-160_{10}$  is  $101011111_2$  (flipping the bit-value of the original number and then appending a one to the front in the place of the sign bit to indicate that the number is negative).

*2's Complement* The 2's complement of  $-160_{10}$  is  $110100001_2$  (flipping the bit-value of the original number, appending a one to the front in the place of the sign bit to indicate that the number is negative, and then adding a one).

**Instructions** Perform the following arithmetic operations on the decimals using signed BINARY numbers and the 2's complement representation.

**Problem 12**  $27 + 19$ 

*Work* Since both numbers are positive, their 2's complement is simply their binary representation with a zero to the left of the most significant bit. However, we should note that if we were to add 27 and 19 without providing additional padding, we would experience overflow (both are less than the next power of two, which would their sum would surpass). As such, we'll add an additional leading zero:  $27_{10} = 0011011_2$  and  $19_{10} = 0010011_2$ .

$$\begin{array}{r} 1 \quad 11 \\ \hline 0011011 \\ + 0010011 \\ \hline 0101110 \end{array}$$

Therefore, our result in 2's complement is  $0101110_2$ .

*Check* To check, simply convert back to base ten (since the result was positive and not negative, we don't have to do additional arithmetic):  $0101110_2 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 46_{10}$ , so we know that the answer is correct.

**Problem 13**  $72 - 85$ 

*Work* We will re-write this operation as addition of a negative. The 2's complement of  $72_{10}$  is  $01001000_2$  ( $72_{10} = 64_{10} + 8_{10} = 2^6 + 2^3 = 1000000_2 + 1000_2$ ). The 2's complement of  $-85_{10}$  is  $10101011_2$  ( $85_{10} = 64_{10} + 16_{10} + 4_{10} + 1_{10} = 2^6 + 2^4 + 2^2 + 2^0 = 1000000_2 + 10000_2 + 100_2 + 1_2 = 1010101_2$ , and we flip the bits so we get  $0101010_2$ , add a sign bit to show that the number is negative  $10101010_2$ , and add one  $10101011_2$ ).

$$\begin{array}{r} 1 \\ \hline 01001000 \\ + 10101011 \\ \hline 11110011 \end{array}$$

Therefore, our result in 2's complement is  $11110011_2$ .

*Check* To check, we can convert back to base ten. Since the result was negative, there's a bit of math we have to do first. First, we'll flip the sign bit, then invert the bit values, and then add one. So,  $11110011_2$  becomes  $00001100_2$ , which becomes  $00001101_2$ , which is equivalent to  $1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 13_{10}$ , so we know that the answer is correct.

**Problem 14**  $31 - 11$ 

*Work* We will re-write this operation as addition of a negative. The 2's complement of  $31_{10}$  is  $011111_2$  (it's just one less than  $32_{10}$  in binary, which is  $100000_2$ , so therefore we have  $11111_2$ , and a sign bit of zero at the front). The 2's complement of  $-11_{10}$  is  $110101_2$  ( $11_{10} = 01011_2$  (we must match the number of bits of the other number), and we flip the bits so we get  $10100_2$ , add a sign bit to show that the number is negative  $110100_2$ , and add one  $110101_2$ ).

$$\begin{array}{r}
 11111 \\
 011111 \\
 +110101 \\
 \hline
 010100
 \end{array}$$

We should note that there was carry-out from the sign bit, but no overflow so our answer should be correct. Therefore, our result in 2's complement is  $010100_2$ .

*Check* To check, simply convert back to base ten (since the result was positive and not negative, we don't have to do additional arithmetic):  $10100_2 = 1 \times 2^4 + 1 \times 2^2 = 20_{10}$ , so we know that the answer is correct.

**Instructions** Answer the following questions regarding BINARY numbers.

**Problem 15** What is the LARGEST positive unsigned value that can be represented using 12 bits?

*Work* Per the formula given in class:

*Theorem* Given an a number  $n$ , the largest positive value that can be stored in the  $n$ -bit binary number can be obtained by  $2^n - 1$ . So, given a 12-bit binary number, the largest positive unsigned value that can be represented is  $2^{12} - 1 = 4095_{10}$ .

**Problem 16** Using 16-bit 2's complement signed numbers, what is the largest POSITIVE magnitude that can be represented? What is the smallest NEGATIVE magnitude (e.g.  $-10$  is smaller than  $-5$ )?

*Work* Per the slide that was shown in class:

*Theorem* Given an a number  $n$ , the range of values for an  $n$ -bit binary number as a 2's complement is  $[-2^{n-1}, (2^{n-1} - 1)]$ . As such, the largest positive value that can be stored is  $2^{15} - 1 = 32767_{10}$ . However, the largest negative value that can be stored is the negative of one more than the largest positive value, which would be  $-32768_{10}$ .