LSU EE 2720-2 Homework 1 Solution Due: 21 September 2011

Solution updated 27 September 2011, 15:30:45 CDT

Problem 1: Perform each of the conversions below.

Convert 812_{10} to Hexadecimal, Binary, and Octal the smart way. The smart way is to convert it to hexadecimal first. From there it is a simple matter to convert the hex to binary and then the binary to octal.

Solution $812_{10} = 32c_{16} = 11\,0010\,1100_2 = 1454_8$.

Convert 812_{16} to decimal.

Solution $812_{16} = 2066_{10}$.

Convert 812_{10} to BCD, Excess-3, and 2421 encoding.

Solution BCD: $812_{10} =$ 0x812 = 1000 0001 0010. Excess-3: $812_{10} =$ 0xb45 = 1011 0100 0101. In 2421 encoding: $812_{10} =$ 0xe12 = 1110 0001 0010.

Convert -812_{10} to 12-bit: signed magnitude, 2's complement, and 1's complement representations.

Solution: Signed-magnitude $812_{10} =$ 1011 0010 1100, 2's complement $812_{10} =$ 1100 1101 0100, 1's complement $812_{10} =$ 1100 1101 0011.

Convert the 8-bit quantity 1010 0101 to decimal assuming it is: binary unsigned, 2's complement signed, 1's complement signed, and BCD unsigned. If the quantity 1010 0101 is not a valid number in any of these representation, then use "not valid" as your answer instead of the decimal number.

Solution: Binary unsigned: 1010 0101 $= 165_{10}$, 2's complement: 1010 0101 $\rightarrow 0101\,1010_2 + 1 \rightarrow -91_{10}$, 1's complement: 1010 0101 $\rightarrow 0101\,1010_2 \rightarrow -90_{10}$, BCD: 1010 0101 The ten's digit in 1010 0101 is 1010, which is greater than 9, so 1010 0101 is not a BCD number. 1010 0101 is not valid BCD so there is nothing to convert.

Note: Assigning an invalid BCD number in the original assignment was a mistake.

Problem 2: Perform the arithmetic indicated below.

- Show the answers in the same representation as the operands (binary, BCD, etc) and also in decimal.
- Show your work.
- Indicate whether there was overflow.

For the problems below do the arithmetic in the indicated representation.

Add the following two 8-bit unsigned binary integers:
0111 0010 + 1001 0011.

Solution: In decimal, $114+147 = 261$. But 261 is not representable as an 8-bit unsigned number, so there is overflow. The calculation in the given representation is 0111 0010 + 1001 0011 = 0000 0101.

Add the following two 9-bit unsigned binary integers (leading zeros omitted):
`111 0010 + 1001 0011.`

Solution: In decimal, $114 + 147 = 261$, this is representable in 9 bits, so there is no overflow. The calculation in the given representation is 0111 0010 + 1001 0011 = 1 0000 0101.

Add the following two 8-bit unsigned BCD integers:
`0111 0010 + 1001 0011.`

Solution: In decimal, $72+93 = 165$. The problem statement said that the answers had to be in the same representation as the operands, which is 8-bit BCD, so the full 3-digit sum, which would be 0001 0110 0101 in 12-bit BCD, is not representable in 8-bit BCD and so there is overflow. With the overflow, the sum is: 0111 0010 + 1001 0011 = 0110 0101.

Add the following two 8-bit 2's complement integers:
`0111 0010 + 1001 0011.`

Solution: For 2's complement one should do the arithmetic in binary and double-check in decimal. In decimal $114-109 = 5$. Since the operands differ in sign there cannot be overflow. 0111 0010 + 1001 0011 = 0000 0101.

Add the following two 9-bit 2's complement integers (leading zeros—and only zeros—omitted):
`111 0010 + 1001 0011.`

Solution: Notice that 1001 0011 is negative in a 8 bit 2's complement representation but positive in 9 bit 2's complement (because bit position 9 is a zero). So just add them as positive numbers. 0111 0010 + 1001 0011 = 1 0000 0101.

Add the following two 8-bit 1's complement integers:
`0111 0010 + 1001 0011.`

Solution: For 1's complement one should to the arithmetic in binary and double-check in decimal. Don't forget to add the carry out to the sum. In decimal $114 - 108 = 6$. Since the operands differ in sign there cannot be overflow. 0111 0010 + 1001 0011 = 0000 0101 + 1 = 0000 0110.

For the problems below, do the arithmetic in any form you like, but show the result in the indicated representation.

Add the following 24-bit ASCII encoded decimal numbers given in hexadecimal:
`0x203337 + 0x203535.`

Solution: First, recall that ASCII encodes characters (alphabetic, numeric, punctuation, etc). So an ASCII-encoded decimal number will consists of characters for the digits. For example a 3 in binary is 11 but the digit 3 in ASCII is $51_{10} = 33_{16} = 0011\,0011_2$. A 24-bit ASCII encoding can hold 3 characters, and so for 0x203337 the digits are 0x20 0x33 0x37 (still in hex), consulting an ASCII table we find that 0x20 is a space, 0x33 is the digit 3, and 0x37 is the digit 7. Therefore 0x203337 represents the number 37. By a similar argument 0x203535 represents 55. In decimal, $37 + 55 = 92$. Encoding the result back into ASCII we get the sum: 0x203337 + 0x203535 = 0x303932.

Add the following 32-bit ASCII encoded numbers in English given in hexadecimal:
`0x20 204f 4e45 + 0x20 2054 574f.`

Solution: Consulting our ASCII table we find `0x20204f4e45 = ONE` and `0x202054574f = TWO`. In decimal, $1+2 = 3$. Representing in English encoded in ASCII we get `0x20204f4e45 + 0x202054574f = 0x5448524545`.