Computer Architecture

Tutorial 3 – Number Representation and Binary Arithmetic

1) Convert the following binary numbers to decimal:
   (a) 0110, (b) 1011, (c) 10101010

2) Convert the following binary numbers to hexadecimal:
   (a) 1110, (b) 11011, (c) 1010111101110010

3) Convert the following decimal numbers to binary and hexadecimal:
   (a) 12, (b) 27, (c) 96

4) For an 8-bit group, work out the representation for -37_{10} in

   a) Sign & Magnitude
   b) One's Complement
   c) Two's Complement
   d) Excess-255 (Note: The n in Excess-n does not have to equal $2^m - 1$, where m is the number of bits in the bit-group)
   e) Excess-128

5) Express 9876510 in Binary Coded Decimal

6) Form the negative equivalent of the following 8-bit Two's Complement numbers

   (a) 00011001, (b) 00011110, (c) 01101000, (d) 01110100

   by comparing the resulting bit-patterns to the originals, can you spot a "short cut" method for the conversion?

Computer Architecture

Tutorial 3 – Number Representation and Binary Arithmetic - Answers

1) Convert the following binary numbers to decimal:
   (a) 0110 = 6, (b) 1011 = 11, (c) 10101010 = 170

2) Convert the following binary numbers to hexadecimal:
   (a) 1110 = E, (b) 11011 = 1B, (c) 1010111101110010 = AF72

3) Convert the following decimal numbers to binary and hexadecimal:
   (a) 12 = 1100 & C, (b) 27 = 11011 & 1B, (c) 96 = 1100000 & 60

4) For an 8-bit group, work out the representation for -37_{10} in

   37_{10} = 100101

   a) Sign & Magnitude 10100101

   b) One's Complement 11011010

   c) Two's Complement 11011011

   d) Excess-255 -37 = -37 + 255 = 218 = 11011010

   e) Excess-128 -37 = -37 + 128 = 91 = 01011011

5) Express 9876510 in Binary Coded Decimal

9	8	7	6	5	1	0
1001	1000	0111	0110	0101	0001	0000

6) Form the negative equivalent of the following 8-bit Two's Complement numbers.

   (a) 00011001, (b) 00011110, (c) 01101000, (d) 01110100

(a) 00011001 = 16 + 8 + 1 = 25_{10}

“invert the bits and add 1” 11100110 + 1 = 11100111

check: 11100111 = -128 + (64 + 32 + 4 + 2 + 1) = -25_{10}

(b) 00011110 = 16 + 8 + 4 + 2 = 30_{10}

"invert the bits and add 1" 11100001 + 1 = 11100010

check: 11100010 = -128 + (64 + 32 + 2) = -30_{10}

(c) 01101000 = 64 + 32 + 8 = 104_{10}

"invert the bits and add 1" 10010111 + 1 = 10011000

check: 10011000 = -128 + (16 + 8) = -104_{10}

(d) 01110100 = 64 + 32 + 16 + 4 = 116_{10}

"invert the bits and add 1" 10001011 + 1 = 10001100

check: 10001100 = -128 + (8 + 4) = -116_{10}

by comparing the resulting bit patterns to the originals, can you spot a "short cut" method for the conversion?

Take another look at the bit patterns:

positive: 00011001   00011110   01101000   01110100
negative: 11100111   11100010   10011000   10001100

"starting from the rightmost bit (lsb), copy each bit unchanged up to and including the first 1 then invert all the remaining bits"