Computer Arithmetic 2016–2017 | **Homework 1 Solutions**

1 Addition in different encodings

The result should be in the same format as the inputs. With $A = 01010101_2$ and $B = 10100110_2$, the answers for the additions are:

	A	B	$A + B$	
			01010101_2	
2′s complement	85_{10}	-90_{10}	10100110_2	
			11111011_2	$= -5_{10}$
			01010101_2	
1′s complement	85_{10}	-89_{10}	10100110_2	
			11111011_2	$= -4_{10}$
			01010101_2	
sign magnitude	85_{10}	-38_{10}	11011010_2	(2's complement of magnitude)
			00101111_2	$= 47_{10}$

While for the subtraction the answers are:

	A	$-B$	$A - B$	
			01010101_2	
2′s complement	85_{10}	$90_{10} = 01011010_2$	01011010_2	
			10101111_2	$= -81_{10} = (175_{10})\mathbf{mod_{256}}$ (an overflow occured)
			01010101_2	
1′s complement	85_{10}	$89_{10} = 01011001_2$	01011001_2	
			10101110_2	$= -81_{10} = (174_{10})\mathbf{mod_{255}}$ (an overflow occured)
			01010101_2	
sign magnitude	85_{10}	$38_{10} = 00100110_2$	00100110_2	
			01111011_2	$= 123_{10}$

2 A non-contiguous digit set

The system in this problem has $\beta = 10$ and the number of digits in the set is also 10. The redundancy index ρ is thus equal to zero and the system is *not* redundant.

The important idea here is to find out the weights of the radix that allow you to represent all the numbers from 0 to 99. Obviously, there is no need to go beyond $\beta^2 = 100$. However, we are faced with the simple question of how to represent a number such as 2 or 3?

The solution is to use three digits so that a number represented by $d_1 d_0 d_{-1}$ has the value $d_1 \times \beta^1 + d_0 \times \beta^0 + d_{-1} \times \beta^{-1}$. An implicit fraction point exists between d_0 and d_1. With that choice, the number 2 is represented as $(0)(0).(20) = 20 \times 10^{-1}$ while 3 is represented as $(0)(1).(20) = 1 \times 10^0 + 20 \times 10^{-1}$.

The requested numbers are represented as:

0= (0)(0).(0)	10= (1)(0).(0)	20= (0)(20).(0)	30= (1)(20).(0)	40= (0)(40).(0)
1= (0)(1).(0)	11= (1)(1).(0)	21= (0)(21).(0)	31= (1)(21).(0)	41= (0)(41).(0)
2= (0)(0).(20)	12= (1)(0).(20)	22= (0)(20).(20)	32= (1)(20).(20)	42= (0)(40).(20)
3= (0)(1).(20)	13= (1)(1).(20)	23= (0)(21).(20)	33= (1)(21).(20)	43= (0)(41).(20)
4= (0)(0).(40)	14= (1)(0).(40)	24= (0)(20).(40)	34= (1)(20).(40)	44= (0)(40).(40)
5= (0)(1).(40)	15= (1)(1).(40)	25= (0)(21).(40)	35= (1)(21).(40)	45= (0)(41).(40)
6= (0)(0).(60)	16= (1)(0).(60)	26= (0)(20).(60)	36= (1)(20).(60)	46= (0)(40).(60)
7= (0)(1).(60)	17= (1)(1).(60)	27= (0)(21).(60)	37= (1)(21).(60)	47= (0)(41).(60)
8= (0)(0).(80)	18= (1)(0).(80)	28= (0)(20).(80)	38= (1)(20).(80)	48= (0)(40).(80)
9= (0)(1).(80)	19= (1)(1).(80)	29= (0)(21).(80)	39= (1)(21).(80)	49= (0)(41).(80)

50= (1)(40).(0)	60= (0)(60).(0)	70= (1)(60).(0)	80= (0)(80).(0)	90= (1)(80).(0)
51= (1)(41).(0)	61= (0)(61).(0)	71= (1)(61).(0)	81= (0)(81).(0)	91= (1)(81).(0)
52= (1)(40).(20)	62= (0)(60).(20)	72= (1)(60).(20)	82= (0)(80).(20)	92= (1)(80).(20)
53= (1)(41).(20)	63= (0)(61).(20)	73= (1)(61).(20)	83= (0)(81).(20)	93= (1)(81).(20)
54= (1)(40).(40)	64= (0)(60).(40)	74= (1)(60).(40)	84= (0)(80).(40)	94= (1)(80).(40)
55= (1)(41).(40)	65= (0)(61).(40)	75= (1)(61).(40)	85= (0)(81).(40)	95= (1)(81).(40)
56= (1)(40).(60)	66= (0)(60).(60)	76= (1)(60).(60)	86= (0)(80).(60)	96= (1)(80).(60)
57= (1)(41).(60)	67= (0)(61).(60)	77= (1)(61).(60)	87= (0)(81).(60)	97= (1)(81).(60)
58= (1)(40).(80)	68= (0)(60).(80)	78= (1)(60).(80)	88= (0)(80).(80)	98= (1)(80).(80)
59= (1)(41).(80)	69= (0)(61).(80)	79= (1)(61).(80)	89= (0)(81).(80)	99= (1)(81).(80)

---

3 Unsigned subtraction

1. The effect of complementing the bits of B and adding the carry-in signal is to form the two's complement of B. Hence the result of the addition is in fact

$$\begin{aligned} Sum &= A + (2^n - B) && (1) \\ &= 2^n + (A - B) && (2) \end{aligned}$$

   If $(A - B)$ is positive the 2^n term leads to an output carry. On the other hand, when $(A - B)$ is negative it subtracts from the 2^n and no carry is generated. Hence, the absence of a carry indicates a negative result.

2. It is in two's complement.

---

4 A subtracter

1. The truth table is

a_i	b_i	c_i	t_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

   and the logical equations are $t_{i+1} = a_i b_i \vee a_i \bar{c}_i \vee b_i \bar{c}_i$ and $s_i = a_i \oplus b_i \oplus c_i$.

2. We put n cells in a row and label them from 0 at the least significant side to $n - 1$ at the most significant side. The bits of Y should be connected to the c input and the t output of a cell should be connected to the b input of the adjacent cell of the higher mathematical weight.

   The input b_0 is set to zero. R has $n + 1$ bits: all the s bits from the n cells and the t output from the most significant cell.

3. The bits of R are all negatively valued except for the most significant bit. Hence the equation is $R = r_n 2^n - \sum_{i=0}^{n-1} r_i 2^i$.

---

5 Multiply by 2 and 5

1. In the following truth table, 'd' indicates a don't care value, t_4 is the carry into the higher digit in the case of multiplicaition by 2, and $f_6 f_5 f_4$ are the bits indicating the carry into the higher digit in the case of multiplication by 5.

Original bits	Multiplied by 2		Multiplied by 5	
$b_3b_2b_1b_0$	t_4	$t_3t_2t_1t_0$	$f_6f_5f_4$	$f_3f_2f_1f_0$
0000	0	0000	000	0000
0001	0	0010	000	0101
0010	0	0100	001	0000
0011	0	0110	001	0101
0100	0	1000	010	0000
0101	1	0000	010	0101
0110	1	0010	011	0000
0111	1	0100	011	0101
1000	1	0110	100	0000
1001	1	1000	100	0101
1010	d	dddd	ddd	dddd
1011	d	dddd	ddd	dddd
1100	d	dddd	ddd	dddd
1101	d	dddd	ddd	dddd
1110	d	dddd	ddd	dddd
1111	d	dddd	ddd	dddd

2. Based on the above table and using the don't care values for logic minimization,

$$
\begin{aligned}
t_4 &= b_3 + b_2b_1 + b_2b_0 && (3)\\
t_3 &= b_3b_0 + b_2\bar{b}_1\bar{b}_0 && (4)\\
t_2 &= b_1b_0 + \bar{b}_2b_1 + b_3\bar{b}_0 && (5)\\
t_1 &= \bar{b}_3\bar{b}_2b_0 + b_2b_1\bar{b}_0 + b_3\bar{b}_0 && (6)\\
t_0 &= 0 && (7)\\
f_6 &= b_3 && (8)\\
f_5 &= b_2 && (9)\\
f_4 &= b_1 && (10)\\
f_3 &= 0 && (11)\\
f_2 &= b_0 && (12)\\
f_1 &= 0 && (13)\\
f_0 &= b_0 && (14)\\
& && (15)
\end{aligned}
$$

3. It is evident that in the case of multiplication by 2, the least significant bit of a digit is always zero. Hence, the carry into a digit can be directly added to this bit position without generating further carries into higher bit positions.

   Thus the total delay for the multiplication by 2 for any number of digits is just the delay of the above logic equations. This delay is about 3 gate delays (inverter, 3-input AND, 3-input OR).

   For the multiplication by 5, the three bits $f_6f_5f_4$ representing the carry into a digit may be added to either 0000 or 0101 depending on bit b_0 of that digit, i.e. they are added to b_00b_0. It is important to note that the carry bits are just the most significant three bits of the adjacent

lower digit without any logic involved to compute them. The time delay of the addition is that of a three bit adder. If the time delay of one full adder is estimated to be 2 gate delays then the total delay is at most 6 gate delays.

A shorter delay may be achieved by using a direct gate implementation for this very specific addition:

$$\begin{array}{cccccc} & & & f_6 & f_5 & f_4 \\ + & 0 & & b_0 & 0 & b_0 \\ \hline & f_6b_0 + f_5f_4b_0 & & f_6 \oplus b_0 \oplus (f_5f_4b_0) & f_5 \oplus (f_4b_0) & f_4 \oplus b_0 \end{array}$$

This direct implementation may take only 2 gate delays (3-input AND then an XOR) to produce the multiplication by 5 for any number of digits.
