Solutions - Homework 2

(Due date: February 2nd @ 11:59 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (31 PTS)

- In ALL these problems (a, b, c, d), you MUST show your conversion procedure. **No procedure ≡ zero points**.

a) Convert the following decimal numbers to their 2's complement representations: binary and hexadecimal. (6 pts)

✓ -97.125, 63.3125, -71.25.

- 97.125 = 01100001.001 → -97.125 = 10011110.111 = 0x9E.E
- 63.3125 = 0111111.0101 = 0x3F.5
- 71.25 = 01000111.01 → -71.25 = 10111000.11 = 0xB8.C

b) We want to represent integer numbers between (and including) -32768 to 32768 using the 2C representation. What is the minimum number of bits required? (3 pts)

Range of signed integer with n bits: $[-2^{n-1}, 2^{n-1}-1]$

$\Rightarrow 2^{n-1}-1 \leq 32768 \rightarrow 2^{n-1} \leq 32769 \rightarrow n-1 \geq \log_2 32769 \rightarrow n \geq 16.0000440269 \rightarrow n = 17$

$\therefore$ The minimum required number of bits is $n = 17$.

c) Complete the following table. The decimal numbers are unsigned: (4 pts)

Decimal	BCD	Binary	Reflective Gray Code
269	001001101001	100001101	110001011
102	000100000010	1100110	1010101
110	000100010000	1101110	1011001
687	011010000111	1010101111	1111111000

d) Complete the following table. Use the fewest number of bits in each case: (18 pts)

	REPRESENTATION		
Decimal	**Sign-and-magnitude**	**1's complement**	**2's complement**
-16	110000	101111	10000
-129	110000001	101111110	101111111
-32	1100000	1011111	100000
64	01000000	01000000	01000000
0	00	111111	0
-33	1100001	1011110	1011111
-31	1011111	100000	100001

PROBLEM 2 (20 PTS)

a) What is the minimum number of bits required to represent: (2 pts)

✓ 65,537 symbols?
$\lceil \log_2 65{,}537 \rceil = 17$ bits

✓ Numbers between (and including) 35,000 and 39,096?
$\lceil \lceil 39096 - 35000 + 1 \rceil \rceil = 13$ bits

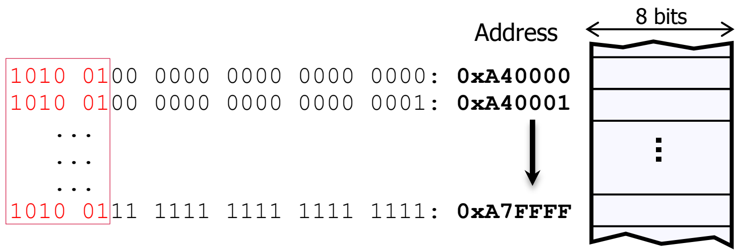
b) A microprocessor has a 24-bit address line. The size of the memory contents of each address is 8 bits. The memory space is defined as the collection of memory positions the processor can address. (6 pts)

- What is the address range (lowest to highest, in hexadecimal) of the memory space for this microprocessor? What is the size (in bytes, KB, or MB) of the memory space? 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes. (2 pts)
  Address Range: 0x000000 to 0xFFFFFF
  With 24 bits, we can address 2^{24} bytes, thus we have $2^4 2^{20}$ = 16 MB

- A memory device is connected to the microprocessor. Based on the memory size, the microprocessor has assigned the addresses 0xA40000 to 0xA7FFFF to this memory device.
  - What is the size (in bytes, KB, or MB) of this memory device?

- What is the minimum number of bits required to represent the addresses only for this memory device?

  As per the figure, we only need 18 bits for the addresses in the given range (where the memory device is located). Thus, the size of the memory device is 2^{18} = 256 KB.



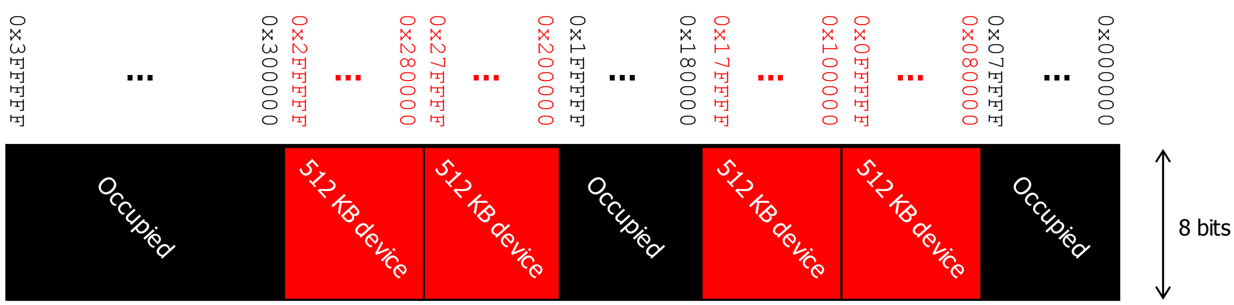
c) The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. (12 pts)

- What is the size (in bytes, KB, or MB) of the memory space? What is the address bus size of the microprocessor? (2 pts)
  Address Range: `0x0000000` to `0x3FFFFF`. To represent all these addresses, we require 22 bits. So, the address bus size of the microprocessor is 22 bits. The size of the memory space is then 2^{22} = 4MB.

- If we have a memory chip of 512KB, how many bits do we require to address 512KB of memory?
  512KB memory device: 512KB = $2^9 2^{10}$ = 2^{19} bytes. Thus, we require 19 bits to address the memory device.

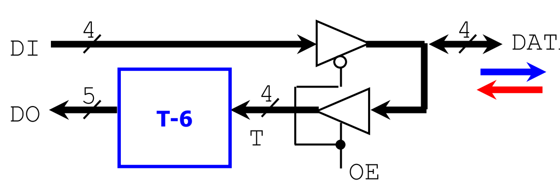
- We want to connect the 512KB memory chip to the microprocessor. For optimal implementation, we must place those 512KB in an address range where every single address shares some MSBs (e.g.: `0x000000` to `07FFFF`). Provide a list of all the possible address ranges that the 512KB memory chip can occupy. You can only use the non-occupied portions of the memory space as shown below. (8 pts)

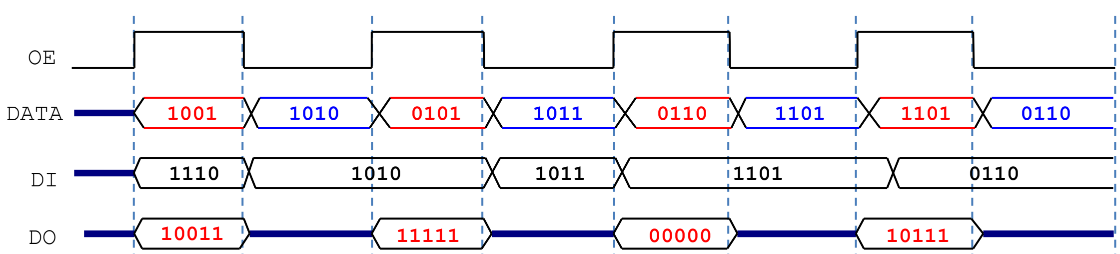
  The 19-bit address range for an 512KB memory would be: `0x000000` to `0x7FFFF`. To place this range within the 22-bit memory space in the figure, we have four options:
  `0x080000 to 0x0FFFFF`
  `0x100000 to 0x17FFFF`
  `0x200000 to 0x27FFFF`
  `0x280000 to 0x2FFFFF`



PROBLEM 3 (12 PTS)

- Complete the timing diagram (signals DO and $DATA$) of the following circuit. The circuit in the blue box computes the signed (2C) operation T-6, with the result having 5 bits. T is a 4-bit signed (2C) number.
  For example: if T=1010 → DO = 1010 – 0110 = 11010 + 11010 = 10100.





PROBLEM 4 (37 PTS)

a) Perform the following additions and subtractions of the following unsigned integers. Use the fewest number of bits n to represent both operators. Indicate every carry (or borrow) from c_0 to c_n (or b_0 to b_n). For the addition, determine whether there is an overflow. For the subtraction, determine whether we need to keep borrowing from a higher bit. (8 pts)

Example (n=8):

- ✓ 210 + 54
- ✓ 77 - 194

```
c8=1 c7=1 c6=1 c5=1 c4=0 c3=1 c2=1 c1=0 c0=0

 54 = 0x36 =  0 0 1 1 0 1 1 0 +
210 = 0xD2 =  1 1 0 1 0 0 1 0
-------------------------------
Overflow!   1 0 0 0 0 1 0 0 0
```

```
Borrow out!  b8=1 b7=0 b6=0 b5=0 b4=0 b3=0 b2=1 b1=0 b0=0

 77 = 0x4D =  0 1 0 0 1 1 0 1 -
194 = 0xC2 =  1 1 0 0 0 0 1 0
-------------------------------
              1 0 0 0 1 0 1 1
```

- ✓ 165 + 89
- ✓ 109 + 53
- ✓ 130 - 43
- ✓ 93 - 129

```
No Overflow   c8=0 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=1 c0=0

165 = 0xA5 =  1 0 1 0 0 1 0 1 +
 89 = 0x59 =  0 1 0 1 1 0 0 1
-------------------------------
254 = 0xFE =  1 1 1 1 1 1 1 0
```

```
No Borrow Out   b8=0 b7=1 b6=1 b5=1 b4=1 b3=1 b2=0 b1=1 b0=0

194 = 0xC2 =  1 1 0 0 0 0 1 0 -
125 = 0x7D =  0 1 1 1 1 1 0 1
-------------------------------
 69 = 0x45 =  0 1 0 0 0 1 0 1
```

```
c7=1 c6=1 c5=1 c4=1 c3=1 c2=0 c1=1 c0=0

109 = 0x6D =  1 1 0 1 1 0 1 +
 53 = 0x35 =  0 1 1 0 1 0 1
-------------------------------
Overflow!   1 0 1 0 0 0 1 0
```

```
Borrow out!  b8=1 b7=0 b6=0 b5=0 b4=0 b3=0 b2=0 b1=0 b0=0

 93 = 0x5D =  0 1 0 1 1 1 0 1 -
129 = 0x81 =  1 0 0 0 0 0 0 1
-------------------------------
     0xDC =  1 1 0 1 1 1 0 0
```

b) We need to perform the following operations, where numbers are represented in 2's complement (2C): (20 pts)

- ✓ 358 + 157
- ✓ 109 - 146
- ✓ -91 + 125
- ✓ -66 -127
- ✓ 87 - 46

- For each case:
  - ✓ Determine the minimum number of bits n required to represent both summands. You might need to sign-extend one of the summands, since for proper summation, both summands must have the same number of bits.
  - ✓ Perform the signed (2C) binary addition, i.e., complete all the carries (c_0 to c_n) and the summation bits (s_0 to s_{n-1}).
  - ✓ Determine whether there is overflow by:
    - i. Using c_n, c_{n-1} (carries).
    - ii. Performing the operation in the decimal system and checking whether the result is within the allowed range for n bits, where n is the minimum number of bits for the summands.
  - ✓ If we want to avoid overflow, what is the minimum number of bits required to represent both the summands and the result?

n = 10 bits

```
c10⊕c9=1     c10=0 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=0 c1=0 c0=0
Overflow!

157 = 0 0 1 0 0 1 1 1 0 1 +
358 = 0 1 0 1 1 0 0 1 1 0
---------------------------
      1 0 0 0 0 0 0 0 1 1

157 + 358 = 515 ∉ [-2^9, 2^9-1] → overflow!
```

To avoid overflow:
n= 11 bits (sign extension)

```
c11⊕c10=0    c11=0 c10=0 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=0 c1=0 c0=0
No Overflow

156 = 0 0 0 1 0 0 1 1 1 0 0 +
359 = 0 0 1 0 1 1 0 0 1 1 1
-----------------------------
515 = 0 1 0 0 0 0 0 0 0 1 1

156 + 359 = 515 ∈ [-2^10, 2^10-1] → no overflow
```

n = 8 bits

```
c8⊕c7=1      c8=1 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=0 c0=0
Overflow!

-127 = 1 0 0 0 0 0 0 1 +
 -66 = 1 0 1 1 1 1 1 0
------------------------
       0 0 1 1 1 1 1 1

-127-66 = -193 ∉ [-2^7, 2^7-1] → overflow!
```

To avoid overflow:
n = 9 bits (sign extension)

```
c9⊕c8=0      c9=1 c8=1 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=0 c0=0
No Overflow

-127 = 1 1 0 0 0 0 0 0 1 +
 -66 = 1 0 1 1 1 1 1 1 0
--------------------------
-193 = 1 0 0 1 1 1 1 1 1

-127-66 = -193 ∈ [-2^8, 2^8-1] → no overflow
```

n = 9 bits

$c_9 \oplus c_8 = 0$
No Overflow

```
        c9=0 c8=0 c7=1 c6=1 c5=0 c4=1 c3=1 c2=0 c1=0 c0=0
 109 = 0 0 1 1 0 1 1 0 1 +
-146 = 1 0 1 1 0 1 1 1 0
-------------------------
 -37 = 1 1 1 0 1 1 0 1 1
```

$109 - 146 = -37 \in [-2^8, 2^8-1] \rightarrow$ no overflow

n = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

```
       c8=1 c7=1 c6=0 c5=1 c4=0 c3=1 c2=1 c1=0 c0=0
 87 = 0 1 0 1 0 1 1 1 +
-46 = 1 1 0 1 0 0 1 0
----------------------
 41 = 0 0 1 0 1 0 0 1
```

$87 - 46 = 41 \in [-2^7, 2^7-1] \rightarrow$ no overflow

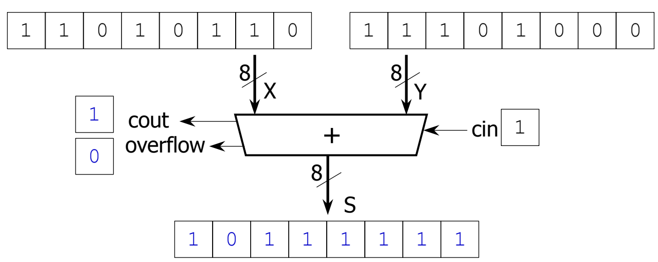
n = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

```
        c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=0 c1=1 c0=0
 125 = 0 1 1 1 1 1 0 1 +
 -91 = 1 0 1 0 0 1 0 1
-----------------------
  34 = 0 0 1 0 0 0 1 0
```

$126 - 91 = 34 \in [-2^7, 2^7-1] \rightarrow$ no overflow

c) For the following 8-bit 2's complement adder, complete all the outputs (S, cout, overflow) given the input values. (3 pts)



d) Get the multiplication results of the following numbers that are represented in 2's complement arithmetic with 4 bits. (6 pts)

✓ 0101×0101, 1011×0111, 1010×1110.

```
      0 1 0 1 x
      0 1 0 1
      -------
      0 1 0 1
    0 0 0 0
  0 1 0 1
0 0 0 0
---------------
0 0 0 1 1 0 0 1
```

```
1 0 1 1 x   ➡      0 1 0 1 x
0 1 1 1            0 1 1 1
                   -------
                   0 1 0 1
                 0 1 0 1
               0 1 0 1
             0 0 0 0
             ---------------
             0 0 1 0 0 0 1 1
                   ⬇
             1 1 0 1 1 1 0 1
```

```
1 0 1 0 x   ➡      0 1 1 0 x
1 1 1 0            0 0 1 0
                   -------
                   0 0 0 0
                 0 1 1 0
               0 0 0 0
             0 0 0 0
             ---------------
             0 0 0 0 1 1 0 0
```