Solutions - Homework 2

(Due date: January 31st @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (32 PTS)

- In ALL these problems (a, b, c), 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. (12 pts)

✓ -93.3125, 172.65625, -64.5078125, -71.25.

- 93.3125 = 01011101.0101 → -93.3125 = 10100010.1011 = 0xA2.B
- 172.65625 = 010101100.10101 = 0x0AC.A8
- 64.5078125 = 01000000.1000001 → -64.5078125 = 10111111.0111111 = 0xBF.7E
- 71.25 = 01000111.01 → -71.25 = 10111000.11

b) Complete the following table. The decimal numbers are unsigned: (8 pts.)

Decimal	BCD	Binary	Reflective Gray Code
299	001010011001	100101011	110111110
587	010110000111	1001001011	1101101110
1587	0001010110000111	11000110011	10100101010
128	000100101000	10000000	11000000
166	000101100110	10100110	11110101
114	000100010100	1110010	1001011
399	001110011001	110001111	101001000
819	100000011001	1100110011	1010101010

c) Complete the following table. Use the fewest number of bits in each case: (12 pts.)

	REPRESENTATION		
Decimal	**Sign-and-magnitude**	**1's complement**	**2's complement**
-126	11111110	10000001	10000010
-103	11100111	10011000	10011001
31	011111	011111	011111
-70	11000110	10111001	10111010
-128	110000000	101111111	10000000
-77	11001101	10110010	10110011

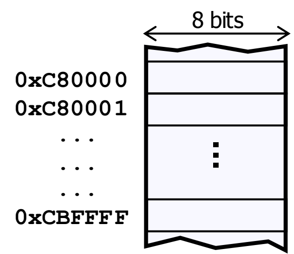
PROBLEM 2 (20 PTS)

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

✓ 125,000 colors?
$\lceil \log_2 125000 \rceil = 17$ bits

✓ Numbers between 64,000 and 68,096?
$\lceil \log_2 (68096 - 64000 + 1) \rceil = 13$ bits

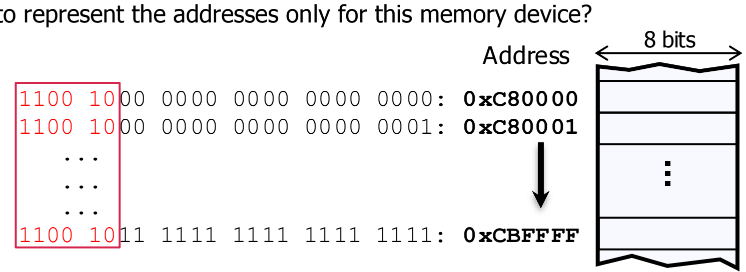
b) A microprocessor has a 24-bit address line. The size of the memory contents of each address is 8 bits. The memory space is defined as the collection of memory positions the processor can address. (6 pts)



- 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
  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 size of the memory, the microprocessor has assigned the addresses 0xC80000 to 0xCBFFFF 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.



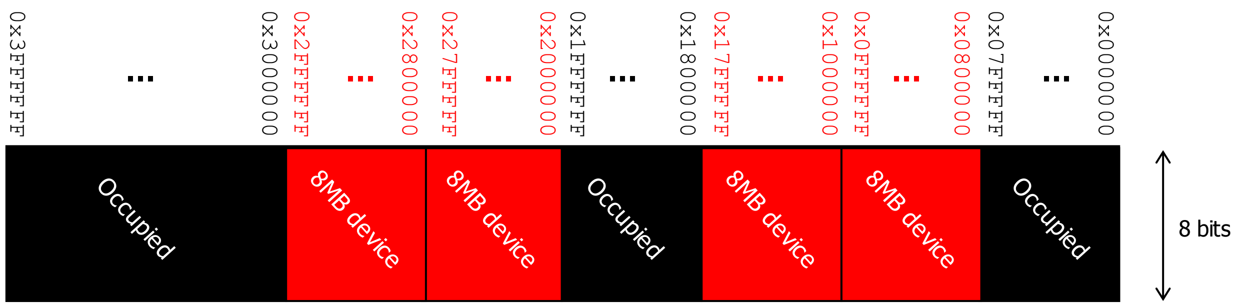
c) The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. (12 pts)

- What is the size (in bytes, KB, or MB) of the memory space? What is the address bus size of the microprocessor?
  Address Range: `0x0000000` to `0x3FFFFFF`. To represent all these addresses, we require 26 bits. So, the address bus size of the microprocessor is 26 bits. The size of the memory space is then 2^{26} = 64MB.

- If we have a memory chip of 8 MB, how many bits do we require to address 8 MB of memory?
  8MB memory device: 8MB = $2^3 2^{20} = 2^{23}$ bytes. Thus, we require 23 bits to address the memory device.

- We want to connect the 8 MB memory chip to the microprocessor. Provide a list of all the possible address ranges that the 8 MB memory chip can occupy. You can only use the non-occupied portions of the memory space as shown below.

  The 23-bit address range for an 8MB memory would be: `0x000000` to `0x7FFFFF`. To place this range within the 26-bit memory space in the figure, we have four options:
  `0x0800000` to `0x0FFFFFF`
  `0x1000000` to `0x17FFFFF`
  `0x2000000` to `0x27FFFFF`
  `0x2800000` to `0x2FFFFFF`



PROBLEM 3 (38 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 byte. (8 pts)

Example (n=8):

✓ 54 + 210

```
                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
```

✓ 77 - 194

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

✓ 271 + 137
✓ 111 + 75

✓ 43 - 97
✓ 128 - 43

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

271 = 0x10F =      1    0    0    0    0    1    1    1    1   +
137 = 0x089 =      0    1    0    0    0    1    0    0    1
--------------------------------------------------------------
408 = 0x198 =      1    1    0    0    1    1    0    0    0
```

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

43 = 0x2B =          0    0    1    0    1    0    1    1   -
97 = 0x61 =          0    1    1    0    0    0    0    1
-------------------------------------------------------------
      0xCA =         1    1    0    0    1    0    1    0
```

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

111 = 0x6F =        1    1    0    1    1    1    1   +
 75 = 0x4B =        1    0    0    1    0    1    1
-----------------------------------------------------
Overflow! -->  1    0    1    1    1    0    1    0
```

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

128 = 0x80 =        1    0    0    0    0    0    0    0   -
 43 = 0x2B =        0    0    1    0    1    0    1    1
------------------------------------------------------------
 85 = 0x55 =        0    1    0    1    0    1    0    1
```

b) We need to perform the following operations, where numbers are represented in 2's complement: (24 pts)

✓ -97 + 256	✓ 99 - 62
✓ 413 + 617	✓ -127 - 37
✓ -122 - 26	✓ -2 - 64

- For each case:
  - ✓ Determine the minimum number of bits 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 binary addition in 2's complement arithmetic. The result must have the same number of bits as the summands.
  - ✓ 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

$c_{10} \oplus c_9 = 0$
No Overflow

		$c_{10}=1$	$c_9=1$	$c_8=0$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$
-97	=		1	1	1	0	0	1	1	1	1	1 +
256	=		0	1	0	0	0	0	0	0	0	0
159	=		0	0	1	0	0	1	1	1	1	1

$-97+256 = 159 \in [-2^9, 2^9-1] \rightarrow$ no overflow

n = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

		$c_8=1$	$c_7=1$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=1$	$c_1=0$	$c_0=0$
99	=		0	1	1	0	0	0	1	1 +
-62	=		1	1	0	0	0	0	1	0
37	=		0	0	1	0	0	1	0	1

$99-62 = 37 \in [-2^7, 2^7-1] \rightarrow$ no overflow

n = 11 bits

$c_{11} \oplus c_{10} = 1$
Overflow!

		$c_{11}=0$	$c_{10}=1$	$c_9=1$	$c_8=1$	$c_7=1$	$c_6=1$	$c_5=1$	$c_4=1$	$c_3=0$	$c_2=0$	$c_1=1$	$c_0=0$
413	=		0	0	1	1	0	0	1	1	1	0	1 +
617	=		0	1	0	0	1	1	0	1	0	0	1
			1	0	0	0	0	0	0	0	1	1	0

$413+617 = 1030 \notin [-2^{10}, 2^{10}-1] \rightarrow$ overflow!

To avoid overflow:
n = 12 bits (sign-extension)

$c_{12} \oplus c_{10} = 0$
No Overflow

		$c_{12}=0$	$c_{11}=0$	$c_{10}=1$	$c_9=1$	$c_8=1$	$c_7=1$	$c_6=1$	$c_5=1$	$c_4=1$	$c_3=0$	$c_2=0$	$c_1=1$	$c_0=0$
413	=		0	0	0	1	1	0	0	1	1	1	0	1 +
617	=		0	0	1	0	0	1	1	0	1	0	0	1
1030	=		0	1	0	0	0	0	0	0	0	1	1	0

$413+617 = 1030 \in [-2^{11}, 2^{11}-1] \rightarrow$ no overflow

n = 8 bits

$c_8 \oplus c_7 = 1$
Overflow!

		$c_8=1$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=1$	$c_1=1$	$c_0=0$
-127	=		1	0	0	0	0	0	0	1 +
-37	=		1	1	0	1	1	0	1	1
			0	1	0	1	1	1	0	0

$-127-37 = -164 \notin [-2^7, 2^7-1] \rightarrow$ overflow!

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

$c_9 \oplus c_8 = 0$
No Overflow

		$c_9=1$	$c_8=1$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=1$	$c_1=1$	$c_0=0$
-127	=		1	1	0	0	0	0	0	0	1 +
-37	=		1	1	1	0	1	1	0	1	1
-164	=		1	0	1	0	1	1	1	0	0

$-127-37 = -164 \in [-2^8, 2^8-1] \rightarrow$ no overflow

n = 8 bits

$c_8 \oplus c_7 = 1$
Overflow!

		$c_8=1$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=1$	$c_2=1$	$c_1=0$	$c_0=0$
-122	=		1	1	0	0	0	0	1	0 +
-26	=		1	1	1	0	0	1	1	0
			0	1	1	0	1	1	0	0

$-122-26 = -148 \notin [-2^7, 2^7-1] \rightarrow$ overflow!

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

$c_9 \oplus c_8 = 0$
No Overflow

		$c_9=1$	$c_8=1$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=1$	$c_2=1$	$c_1=0$	$c_0=0$
-122	=		1	1	1	0	0	0	0	1	0 +
-26	=		1	1	1	1	0	0	1	1	0
			1	0	1	1	0	1	1	0	0

$-122-26 = -148 \in [-2^8, 2^8-1] \rightarrow$ no overflow

n = 7 bits

$c_7 \oplus c_6 = 1$
Overflow!

		$c_7=1$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$
-2	=		1	1	1	1	1	1	0 +
-64	=		1	0	0	0	0	0	0
			0	1	1	1	1	1	0

$-2 -64 = -66 \notin [-2^6, 2^6-1] \rightarrow$ overflow!

To avoid overflow:
n = 8 bits (sign-extension)

$c_8 \oplus c_7 = 1$
Overflow!

		$c_8=1$	$c_7=1$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$
-2	=		1	1	1	1	1	1	1	0 +
-64	=		1	1	0	0	0	0	0	0
			1	0	1	1	1	1	1	0

$-2 -64 = -66 \in [-2^7, 2^7-1] \rightarrow$ overflow!

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

✓ `0100×0101, 1000×0110, 1001×1001.`

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

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

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

PROBLEM 4 (10 PTS)

- Complete the timing diagram (signals DO and $DATA$) of the following circuit. The circuit in the blue box computes the unsigned summation `P+8`, with the result having 5 bits.
  For example: if `P=1001 → DO=1001 + 1000 = 10001. If P=1100 → DO=1100+1000 = 10100.`