Solutions - Homework 2

(Due date: February 5^{th} @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (20 PTS)

- In these problems, you MUST show your conversion procedure.
  a) Convert the following decimal numbers to their 2's complement representations: binary and hexadecimal. (6 pts)
    ✓ -393.3125, 37.65625, -128.5078125, -31.25.
    - 393.3125 = 0110001001.0101 → -393.3125 = 1001110110.1011 = 0xE76.B
    - 37.65625 = 0100101.10101 = 0x25.A8
    - 128.5078125 = 010000000.1000001 → -128.5078125 = 101111111.0111111 = 0xF7F.7E
    - 31.25 = 011111.01 → -31.25 = 100000.11 = 0xE0.C

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

Decimal	BCD	Binary	Reflective Gray Code
397	001110010111	110001101	101001011
634	011000110100	1001111010	1101000111
835	100000110101	1101000011	1011100010
256	001101010110	100000000	110000000
232	001000110010	11101000	10011100
114	000100010100	1110010	1001011
401	010000000001	110010001	101011001
295	001010010101	100100111	110110100

  c) Complete the following table. Use the fewest number of bits in each case: (6 pts.)

	REPRESENTATION		
Decimal	Sign-and-magnitude	1's complement	2's complement
-123	11111011	10000100	10000101
-256	1100000000	1011111111	100000000
-77	11001101	10110010	10110011
-51	1110011	1001100	1001101
-165	110100101	101011010	101011011
217	011011001	011011001	011011001

PROBLEM 2 (15 PTS)

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

✓ 100,000 symbols?
$log2100000 = 17$

✓ Numbers between 35,000 and 43,192?
$\lceil \log_2(43192 - 35000 + 1) \rceil = \lceil \log_2 8193 \rceil = 14$

b) A microprocessor has a 32-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. (5 pts)

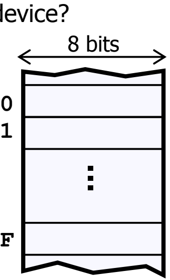
- What is the address range (lowest to highest, in hexadecimal) of the memory space for this microprocessor? What is the size (in bytes, KB, or MB) of the memory space? 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes

  Address range: 0x00000000 to 0xFFFFFFFF.
  With 32 bits, we can address 2^{32} bytes, thus we have $2^2 2^{30}$ = 4 GB of address space

- A memory device is connected to the microprocessor. Based on the size of the memory, the microprocessor has assigned the addresses 0xB1C00000 to 0xB1FFFFFF 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 below, we only need 22 bits for the addresses in the given range. Thus, the size of the memory device is 2^{22} = 4MB.

```
                                                     Address
1011 0001 1100 0000 0000 0000 0000 0000: 0xB1C00000
1011 0001 1100 0000 0000 0000 0000 0001: 0xB1C00001
   ...
   ...
   ...
1011 0001 1111 1111 1111 1111 1111 1111: 0xB1FFFFFF
```



c) The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. (8 pts)

- What is the size (in bytes, KB, or MB) of the memory space? What is the address bus size of the microprocessor?

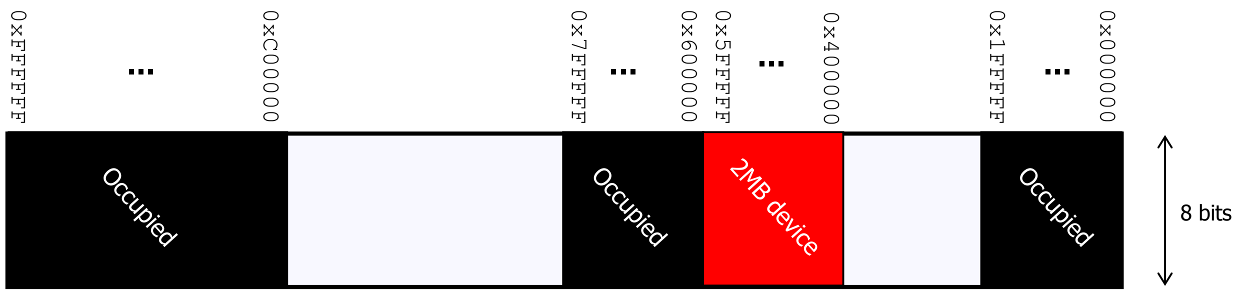
  Address size: `0x000000` to `0xFFFFFF`. To represent all these addresses, we require 24 bits. So, the address bus size of the microprocessor is 24 bits. The size of the memory space is then 2^{24} = 16 MB.

- If we have a memory chip of 2MB, how many bits do we require to address 2MB of memory?

  2 MB memory device: 2MB = $2 \times 2^{20} = 2^{21}$ bytes. Thus, we require 21 bits to address the memory device.

- We want to connect the 2MB memory chip to the microprocessor. Provide an address range so that 2MB of memory is properly addressed. You can only use the non-occupied portions of the memory space as shown in the figure below.

  2MB of memory require 21 bits. The 21-bit address range would be from `0x000000` to `0x1FFFFF`. Within the entire 24-bit memory space, there are four options to place those 2MB in the figure. In particular, we picked the address range from `0x400000` to `0x5FFFFF`.



PROBLEM 3 (30 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

  $c_8=1$, $c_7=1$, $c_6=1$, $c_5=1$, $c_4=0$, $c_3=1$, $c_2=1$, $c_1=0$, $c_0=0$

```
 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! → $b_8=1$, $b_7=0$, $b_6=0$, $b_5=0$, $b_4=0$, $b_3=0$, $b_2=1$, $b_1=0$, $b_0=0$

```
 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
```

- ✓ 271 + 137
- ✓ 111 + 75

- ✓ 43 - 97
- ✓ 128 - 43

No Overflow: $c_9=0$, $c_8=0$, $c_7=0$, $c_6=0$, $c_5=0$, $c_4=1$, $c_3=1$, $c_2=1$, $c_1=1$, $c_0=0$

```
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! → $b_8=1$, $b_7=1$, $b_6=0$, $b_5=0$, $b_4=0$, $b_3=0$, $b_2=0$, $b_1=0$, $b_0=0$

```
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
```

$c_7=1$, $c_6=0$, $c_5=0$, $c_4=0$, $c_3=1$, $c_2=1$, $c_1=0$, $c_0=0$

```
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: $b_8=0$, $b_7=1$, $b_6=1$, $b_5=1$, $b_4=1$, $b_3=1$, $b_2=1$, $b_1=1$, $b_0=0$

```
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: (16 pts)

- ✓ `-97 + 256`
- ✓ `99 - 62`
- ✓ `413 + 617`
- ✓ `-127 - 37`

- 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 $\in [-2^9, 2^9-1]$ → 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 $\in [-2^7, 2^7-1]$ → 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 $\notin [-2^{10}, 2^{10}-1]$ → 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 $\notin [-2^7, 2^7-1]$ → 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 $\in [-2^{11}, 2^{11}-1]$ → no 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 $\in [-2^8, 2^8-1]$ → no 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.`

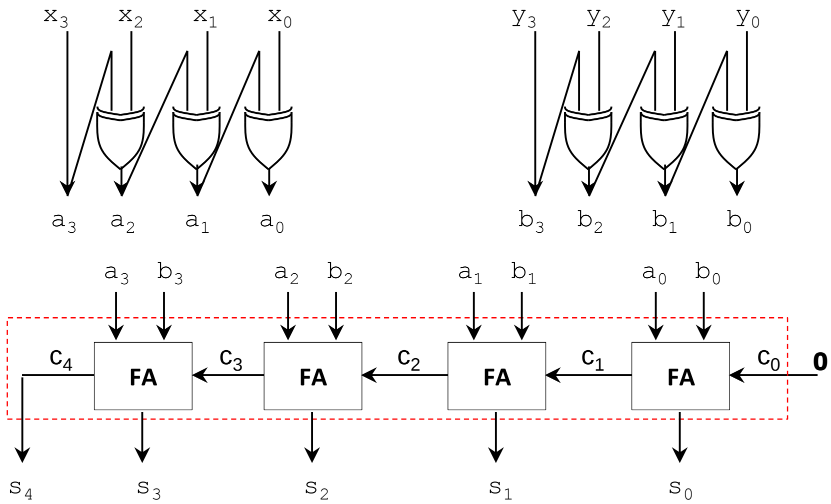
```
         0 1 0 0 x
         0 1 0 1
         -------
         0 1 0 0
       0 0 0 0
     0 1 0 0
   0 0 0 0
 ---------------
 0 0 0 1 0 1 0 0
```

```
 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
```

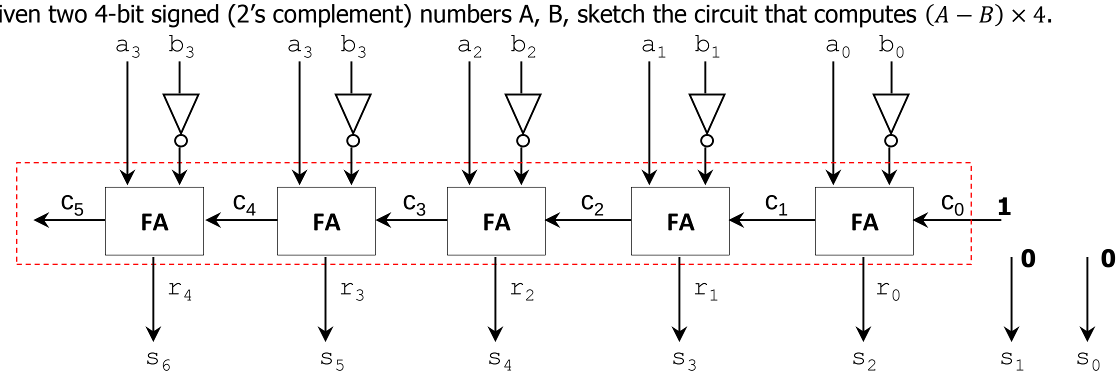
```
 1 0 0 1 x   ➡         0 1 1 1 x
 1 0 0 1               0 1 1 1
                       -------
                       0 1 1 1
                     0 1 1 1
                   0 1 1 1
                 0 0 0 0
               ---------------
               0 0 1 1 0 0 0 1
```

PROBLEM 4 (15 PTS)

- In these problems, you can use full adders and logic gates. Make sure your circuit works for all cases. If there is overflow, design your circuit so that the final answer is always the correct one with the correct number of bits.
  a) Given two 4-bit numbers provided in gray code, sketch the circuit that computes the summation of the unsigned decimal numbers these gray codes represent.



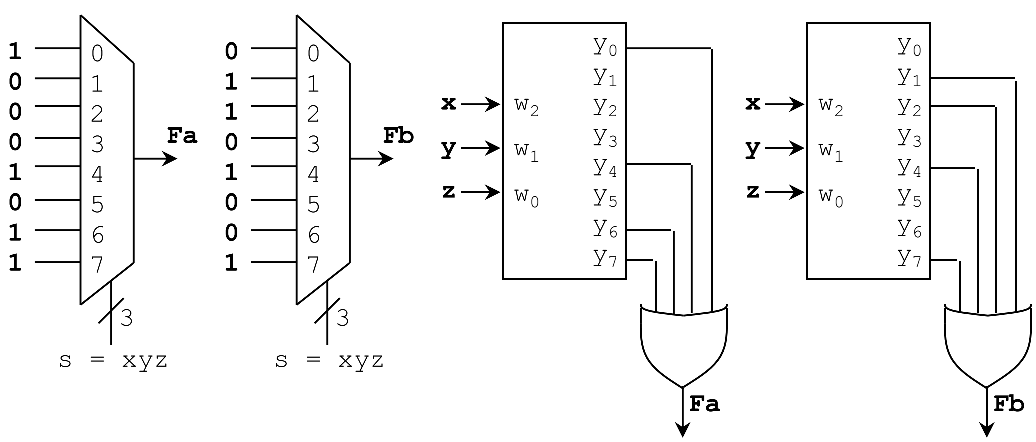
  b) Given two 4-bit signed (2's complement) numbers A, B, sketch the circuit that computes $(A - B) \times 4$.



PROBLEM 5 (20 PTS)

a) Implement the following functions using i) decoders (and OR gates) and ii) multiplexers: (5 pts)
- ✓ $F_a = \overline{Y+Z} + XY$
- ✓ $F_b = \bar{X} \oplus Y \oplus \bar{Z}$

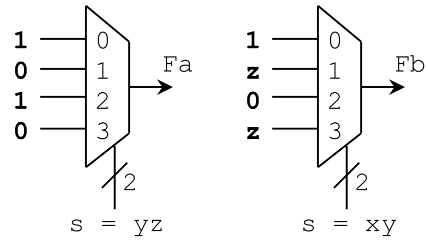
w_2 x	w_1 y	w_0 z	Fa	Fb
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1



b) Using only a 4-to-1 MUX, implement the following functions. (5 pts)

✓ $F_a(X,Y,Z) = \sum(m_0, m_2, m_4, m_6)$ ✓ $F_b(X,Y,Z) = \prod(M_2, M_4, M_5, M_6)$

x	y	z	Fa	Fb
0	0	0	1	1
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	1



c) Complete the timing diagram of the circuit shown below: (10 pts)