Solutions - Homework 2

(Due date: October 3rd (001), October 2nd (006) @ 5:30 pm)
Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (28 PTS)

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

- ✓ 141,000 symbols? $\lceil \log_2 141000 \rceil = 18$
- ✓ Numbers between 0 to 16384? $\lceil \log_2 16385 \rceil = 15$

b) A microprocessor has a 28-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)

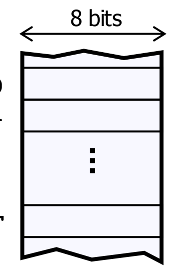
- What is the address range (lowest to highest, in hexadecimal) of the memory space for this microprocessor? What is the size (in bytes, KB, or MB) of the memory space? 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes
  Address Range: `0x0000000` to `0xFFFFFFF`
  With 28 bits, we can address 2^{28} bytes, thus we have $2^8 2^{20}$ = 256 MB of address space

- A memory device is connected to the microprocessor. Based on the size of the memory, the microprocessor has assigned the addresses `0xD040000` to `0xD07FFFF` 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 address in the given range (where the memory device is located). Thus, the size of the memory is 2^{18} = 256KB.

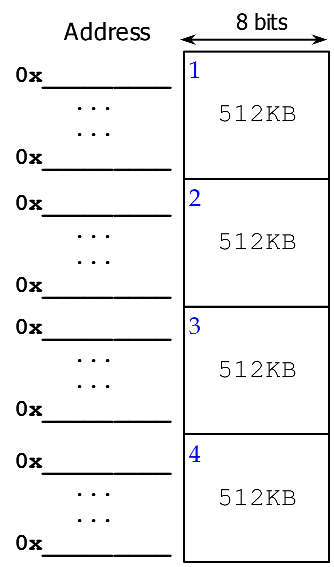
Address

```
1101 0000 0100 0000 0000 0000 0000: 0xD040000
1101 0000 0100 0000 0000 0000 0001: 0xD040001
  ...
  ...
  ...
1101 0000 0111 1111 1111 1111 1111: 0xD07FFFF
```



c) A microprocessor has a memory space of 2 MB. Each memory address occupies one byte. (8 pts)

- What is the address bus size (number of bits of the address) of this microprocessor?
  Since 2 MB = 2^{21} bytes, the address bus size is 21 bits.
- What is the range (lowest to highest, in hexadecimal) of the memory space for this microprocessor?
  With 21 bits, the address range is `0x00000` to `0x1FFFF`.
- The figure to the right shows four memory chips that are placed in the given positions:
  - Complete the address ranges (lowest to highest, in hexadecimal) for each of the memory chips.



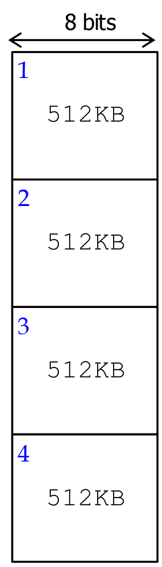
Address

```
0 0000 0000 0000 0000 0000: 0x000000
0 0000 0000 0000 0000 0001: 0x000001
  ...                       ...
0 0111 1111 1111 1111 1111: 0x07FFFF

0 1000 0000 0000 0000 0000: 0x080000
0 1000 0000 0000 0000 0001: 0x080001
  ...                       ...
0 1111 1111 1111 1111 1111: 0x0FFFFF

1 0000 0000 0000 0000 0000: 0x100000
1 0000 0000 0000 0000 0001: 0x100001
  ...                       ...
1 0111 1111 1111 1111 1111: 0x17FFFF

1 1000 0000 0000 0000 0000: 0x180000
1 1000 0000 0000 0000 0001: 0x180001
  ...                       ...
1 1111 1111 1111 1111 1111: 0x1FFFFF
```



d) 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?

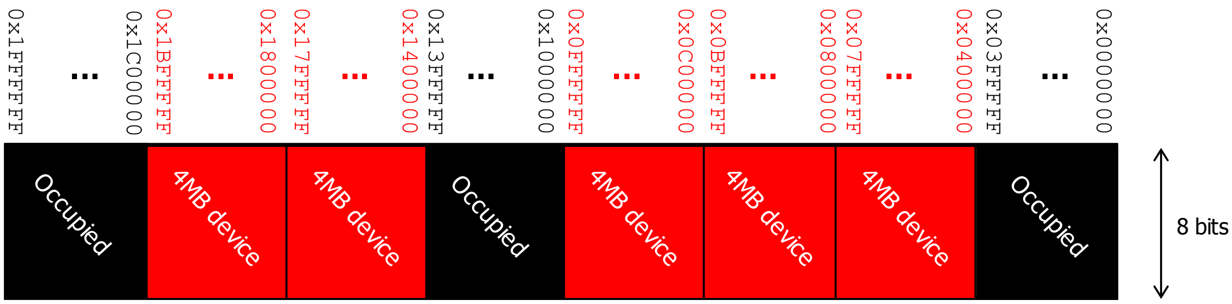
  Address space: `0x0000000` to `0x1FFFFFF`. To represent all these addresses, we require 25 bits. So, the address bus size of the microprocessor is 25 bits. The size of the memory space is then 2^{25}=32 MB.

- If we have a memory chip of 4MB, how many bits do we require to address 4MB of memory?

  4MB = 2^{22} bytes. Thus, we require 22 bits to address only the memory device.

- We want to connect the 4MB memory chip to the microprocessor. For optimal implementation, we must place those 4MB in an address range where every single address shares some MSBs (e.g.: `0x1C00000` to `0x1FFFFFF`). Provide a list of all the possible address ranges that the 4MB memory chip can occupy. You can only use the non-occupied portions of the memory space as shown below.

```
0x0400000 to 0x07FFFFF
0x0800000 to 0x0BFFFFF
0x0C00000 to 0x0FFFFFF
0x1400000 to 0x17FFFFF
0x1800000 to 0x1BFFFFF
```



PROBLEM 2 (30 PTS)

- In ALL these problems, 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)

✓ -137.625, 92.3125, -128.6875, -37.65625.

- 137.625 = 010001001.1010 → -137.625 = 101110110.0110 = 0xF76.6
- 92.3125 = 01011100.0101 = 0x5C.5
- 128.6875 = 010000000.1011 → -128.6875 = 101111111.0101 = 0xF7F.5
- 37.65625 = 0100101.10101 → -37.65625 = 1011010.01011 = 0xDA.58

b) Complete the following table. The decimal numbers are unsigned: (6 pts)

Decimal	BCD	Binary	Reflective Gray Code
137	000100110111	10001001	11001101
171	000101110001	10101011	11111110
588	010110001000	1001001100	1101101010
92	10010010	1011100	1110010
265	001001100101	100001001	110001101
957	100101010111	1110111101	1001100011

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**
-237	111101101	100010010	100010011
-56	1111000	1000111	1001000
-32	1100000	1011111	100000
-21	110101	101010	101011
81	01010001	01010001	01010001
-128	110000000	101111111	10000000

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 bit. (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
```

✓ 191 + 201
✓ 210 + 69

```
             c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=1 c1=1 c0=0
191 = 0xBf =        1    0    1    1    1    1    1    1   +
201 = 0xC9 =        1    1    0    0    1    0    0    1
-------------------------------------------------------
Overflow! →   1     1    0    0    0    1    0    0    0
```

```
             c8=1 c7=1 c6=0 c5=0 c4=0 c3=0 c2=0 c1=0 c0=0
 69 = 0x45 =        0    1    0    0    0    1    0    1   +
210 = 0xD2 =        1    1    0    1    0    0    1    0
-------------------------------------------------------
Overflow! →   1     0    0    0    1    0    1    1    1
```

✓ 130 - 142
✓ 241 - 36

```
Borrow out! → b8=1 b7=1 b6=1 b5=1 b4=1 b3=1 b2=0 b1=0 b0=0
130 = 0x82 =        1    0    0    0    0    0    1    0   -
142 = 0x8E =        1    0    0    0    1    1    1    0
-------------------------------------------------------
      0xF5 =        1    1    1    1    0    1    0    0
```

```
No Borrow Out b8=0 b7=0 b6=0 b5=0 b4=1 b3=1 b2=0 b1=0 b0=0
241 = 0xF1 =        1    1    1    1    0    0    0    1   -
 36 = 0x24 =        0    0    1    0    0    1    0    0
-------------------------------------------------------
205 = 0xCD =        1    1    0    0    1    1    0    1
```

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

✓ 489 + 23
✓ 256 - 87
✓ -129 + 126
✓ -255 - 231
✓ -35 + 66
✓ 985 + 122

- 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

$c_{10} \oplus c_9 = 1$
Overflow!

```
      c10=0 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=1 c1=1 c0=0
489 =         0    1    1    1    1    0    1    0    0    1   +
 23 =         0    0    0    0    0    1    0    1    1    1
-----------------------------------------------------------
              1    0    0    0    0    0    0    0    0    0
```

489+23 = 512 $\notin$ [-2^9, 2^9-1] → overflow!

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

$c_{11} \oplus c_{10} = 0$
No Overflow

```
      c11=0 c10=0 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=1 c2=1 c1=1 c0=0
489 =          0     0    1    1    1    1    0    1    0    0    1   +
 23 =          0     0    0    0    0    0    1    0    1    1    1
-----------------------------------------------------------------
512 =          0     1    0    0    0    0    0    0    0    0    0
```

490+22 = 512 $\in$ [-2^{10}, $2^{10}-1$] → no overflow

n = 9 bits

$c_9 \oplus c_8 = 1$
Overflow!

```
       c9=1 c8=0 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=1 c0=0
-255 =        1    0    0    0    0    0    0    0    1   +
-231 =        1    0    0    0    1    1    0    0    1
-------------------------------------------------------
              0    0    0    0    1    1    0    1    0
```

-255-231 = -486 $\notin$ [-2^8, 2^8-1] → overflow!

To avoid overflow:
n = 10 bits (sign-extension)

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

```
       c10=1 c9=1 c8=0 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=1 c0=0
-255 =          1    1    0    0    0    0    0    0    0    1   +
-231 =          1    1    0    0    0    1    1    0    0    1
-------------------------------------------------------------
-486 =          1    0    0    0    0    1    1    0    1    0
```

-255-231 = -486 $\in$ [-2^9, 2^9-1] → no overflow

n = 10 bits

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

$c_{10}=1$, $c_9=1$, $c_8=0$, $c_7=0$, $c_6=0$, $c_5=0$, $c_4=0$, $c_3=0$, $c_2=0$, $c_1=0$, $c_0=0$

```
 -87 = 1 1 1 0 1 0 1 0 0 1 +
 256 = 0 1 0 0 0 0 0 0 0 0
 -------------------------
 169 = 0 0 1 0 1 0 1 0 0 1
```

$-87+256 = 169 \in [-2^9, 2^9-1] \rightarrow$ no overflow

n = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

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

```
 -35 = 1 1 0 1 1 1 0 1 +
  66 = 0 1 0 0 0 0 1 0
 ---------------------
  31 = 0 0 0 1 1 1 1 1
```

$-35+66 = 31 \in [-2^7, 2^7-1] \rightarrow$ no overflow

n = 9 bits

$c_9 \oplus c_8 = 0$
No Overflow

$c_9=0$, $c_8=0$, $c_7=1$, $c_6=1$, $c_5=1$, $c_4=1$, $c_3=1$, $c_2=1$, $c_1=0$, $c_0=0$

```
 -129 = 1 0 1 1 1 1 1 1 1 +
  126 = 0 0 1 1 1 1 1 1 0
 ------------------------
   -1 = 1 1 1 1 1 1 1 0 1
```

$-129+126 = -3 \in [-2^8, 2^8-1] \rightarrow$ no overflow

n = 11 bits

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

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

```
 985 = 0 1 1 1 1 0 1 1 0 0 1 +
 122 = 0 0 0 0 1 1 1 1 0 1 0
 ---------------------------
       1 0 0 0 1 0 1 0 0 1 1
```

$985+122 = 1107 \notin [-2^{10}, 2^{10}-1] \rightarrow$ overflow!

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

$c_{12} \oplus c_{11} = 0$
No Overflow

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

```
  985 = 0 0 1 1 1 1 0 1 1 0 0 1 +
  122 = 0 0 0 0 0 1 1 1 1 0 1 0
 -----------------------------
 1107 = 0 1 0 0 0 1 0 1 0 0 1 1
```

$98+122 = 1107 \in [-2^{11}, 2^{11}-1] \rightarrow$ no overflow

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

✓ 0101×0101, 1001×0110, 1000×1010.

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

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

PROBLEM 4 (8 PTS)

- The following circuit includes a 4-bit bidirectional port. Complete the timing diagram (signals DO and $DATA$) of the following circuit. The 4-bit binary to gray decoder treats input data as unsigned numbers.