Solutions - Homework 2

(Due date: February 6th @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (18 PTS)

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

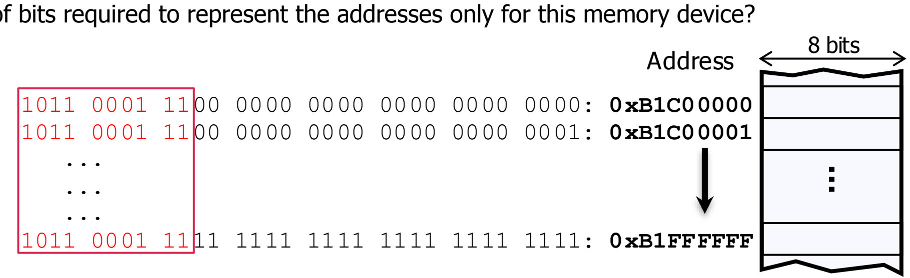
- ✓ 100,000 symbols?
  $\lceil \log_2 100{,}000 \rceil = 17$
- ✓ Numbers between 0 and (including) 32678?
  $\lceil \log_2 (32678 - 0 + 1) \rceil = 16$

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. (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: 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 memory size, 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.



c) The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. (10 pts)

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

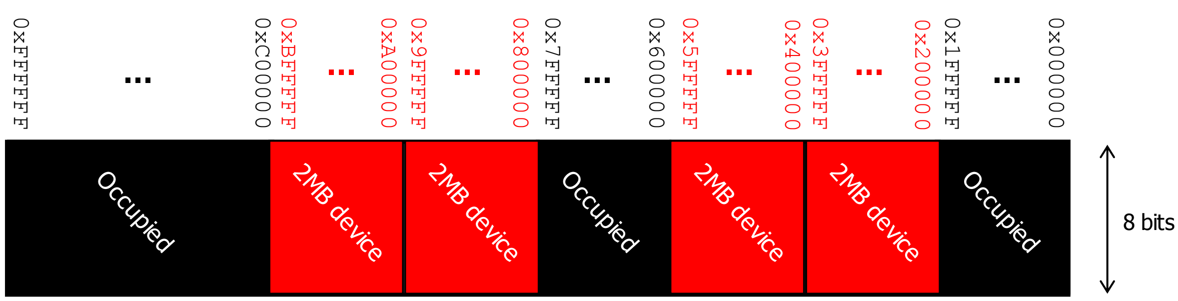
  Address space: 0x000000 to 0xFFFFFF. To represent all these addresses, we require 24 bits. So, the address bus size of the microprocessor is 24 bits. The size of the memory space is then 2^{24} = 16 MB.

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

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

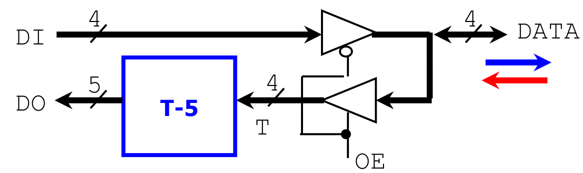
- We want to connect the 2MB memory chip to the microprocessor. For optimal implementation, we must place those 2MB in an address range where every single address shares some MSBs (e.g.: 0x000000 to 0xFFFFFF). Provide a list of all the possible address ranges that the 2MB memory chip can occupy. You can only use the non-occupied portions of the memory space as shown below.

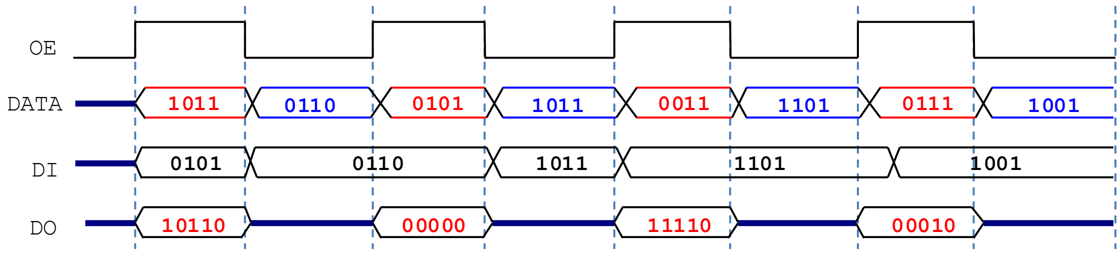
```
0x200000 to 0x3FFFFF
0x400000 to 0x5FFFFF
0x800000 to 0x9FFFFF
0xA00000 to 0xBFFFFF
```



PROBLEM 2 (10 PTS)

- Complete the timing diagram (signals DO and $DATA$) of the following circuit. The circuit in the blue box computes the signed operation T-5, with the result having 5 bits. T is a 4-bit signed number.
  For example: if T=1010 → DO = 1010 – 0101 = 1010 + 1011 = 10101.





PROBLEM 3 (34 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. (12 pts)
  ✓ -137.3125, 37.65625, -128.5078125, -31.25.
    - 137.3125 = 010001010.0101 → -137.3125 = 101110101.1011 = 0xF75.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) We want to represent integer numbers between -1024 to 1024 using the 2C representation. What is the minimum number of bits required? (2 pts)

  Range of signed integer with n bits: $[-2^{n-1}, 2^{n-1}-1]$
  $\Rightarrow 2^{n-1}-1 \leq 1024 \rightarrow 2^{n-1} \leq 1025 \rightarrow n-1 \geq \log_2 1025 \rightarrow n \geq 10.0014 \rightarrow n = 11$
  $\therefore$ The minimum required number of bits is $n = 11$.

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

Decimal	BCD	Binary	Reflective Gray Code
397	001110010111	110001101	101001011
634	011000110100	1001111010	1101000111
835	100000110101	1101000011	1011100010
114	000100010100	1110010	1001011
401	010000000001	110010001	101011001
295	001010010101	100100111	110110100

  d) Complete the following table. Use the fewest number of bits in each case: (14 pts)

	REPRESENTATION		
Decimal	**Sign-and-magnitude**	**1's complement**	**2's complement**
-129	110000001	101111110	101111111
-512	11000000000	10111111111	1000000000
-64	11000000	10111111	1000000
107	01101011	01101011	01101011
0	00	11111	0
-165	110100101	101011010	101011011
-51	1110011	1001100	1001101

PROBLEM 4 (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
```

✓ 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 (2C): (24 pts)

✓ 413 + 617
✓ -97 + 256
✓ 93 - 128
✓ -127 - 37
✓ 99 - 62
✓ -255 - 69

- 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 signed (2C) binary addition. 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 = 11 bits

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

```
       c11=0 c10=1 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=0 c2=0 c1=1 c0=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!

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

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

```
        c12=0 c11=0 c10=1 c9=1 c8=1 c7=1 c6=1 c5=1 c4=1 c3=0 c2=0 c1=1 c0=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

n = 8 bits

$c_8 \oplus c_7 = 1$
Overflow!

```
        c8=1 c7=0 c6=0 c5=0 c4=0 c3=0 c2=1 c1=1 c0=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 = 9 bits (sign-extension)

$c_9 \oplus c_8 = 0$
No Overflow

```
        c9=1 c8=1 c7=0 c6=0 c5=0 c4=0 c3=0 c2=1 c1=1 c0=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

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 = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

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

```
   93 = 0 1 0 1 1 1 0 1 +
 -128 = 1 0 0 0 0 0 0 0
 ----------------------
  -35 = 1 1 0 1 1 1 0 1
```

$93-128 = -35 \in [-2^7, 2^7-1] \rightarrow$ no overflow

n = 9 bits

$c_9 \oplus c_8 = 1$
Overflow!

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

```
 -255 = 1 0 0 0 0 0 0 0 1 +
  -69 = 1 1 0 1 1 1 0 1 1
 ------------------------
        0 1 0 1 1 1 1 0 0
```

$-255-69 = -324 \notin [-2^8, 2^8-1] \rightarrow$ overflow!

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

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

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

```
 -255 = 1 1 0 0 0 0 0 0 0 1 +
  -69 = 1 1 1 0 1 1 1 0 1 1
 --------------------------
 -324 = 1 0 1 0 1 1 1 1 0 0
```

$-255-69 = -324 \in [-2^9, 2^9-1] \rightarrow$ no overflow

c) Get the multiplication results of the following numbers that are represented in 2's complement arithmetic with 4 bits. (6 pts)
✓ 0100×0101, 0110×1010, 1011×1001.

```
        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
```