Solutions - Homework 2

(Due date: October 3rd @ 11:59 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (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
```

✓ 23 + 403
✓ 103 + 204

✓ 77 - 128
✓ 199 - 107

n = 9 bits

No Overflow

```
                 c8=0 c8=0 c7=0 c6=0 c5=1 c4=0 c3=1 c2=1 c1=1 c0=0
 23 =  0x17 =      0  0  0  0  1  0  1  1  1 +
403 = 0x193 =      1  1  0  0  1  0  0  1  1
-----------------------------------------------
423 = 0x1A7 =      1  1  0  1  0  1  0  1  0
```

n = 8 bits

```
Borrow out! -->  b8=1 b7=0 b6=0 b5=0 b4=0 b3=0 b2=0 b1=0 b0=0
 77 = 0x4D =          0  1  0  0  1  1  0  1 -
128 = 0x80 =          1  0  0  0  0  0  0  0
-----------------------------------------------
      0xCD =          1  1  0  0  1  1  0  1
```

n = 8 bits

```
                 c8=1 c7=1 c6=0 c5=0 c4=1 c3=1 c2=0 c1=0 c0=0
103 = 0x67 =          0  1  1  0  0  1  1  1 +
204 = 0xCC =          1  1  0  0  1  1  0  0
-----------------------------------------------
Overflow! -->      1  0  0  1  1  0  0  1  1
```

n = 8 bits

No Borrow Out

```
                 b8=0 b7=1 b6=1 b5=1 b4=1 b3=0 b2=0 b1=0 b0=0
199 = 0xC7 =          1  1  0  0  0  1  1  1 -
107 = 0x6B =          0  1  1  0  1  0  1  1
-----------------------------------------------
 92 = 0x5C =          0  1  0  1  1  1  0  0
```

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

✓ -61 + 128
✓ 225 + 31
✓ 256 - 257
✓ -126 + 263
✓ -511 - 167
✓ 137 + 886

- 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 = 9 bits

$c_9 \oplus c_8 = 1$
Overflow!

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

```
 -61 = 1 1 1 0 0 0 0 1 1 +
 128 = 0 1 0 0 0 0 0 0 0
 -----------------------
  67 = 0 0 1 0 0 0 0 1 1
```

$-61+128 = 67 \in [-2^8, 2^8-1] \rightarrow$ no overflow

n = 10 bits

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

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

```
 -126 =  1 1 1 0 0 0 0 0 1 0 +
  263 =  0 1 0 0 0 0 0 1 1 1
 --------------------------
  137 =  0 0 1 0 0 0 1 0 0 1
```

$-126+263 = 137 \in [-2^9, 2^9-1] \rightarrow$ no overflow

n = 9 bits

$c_9 \oplus c_8 = 1$
Overflow!

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

```
 225 =  0 1 1 1 0 0 0 0 1 +
  31 =  0 0 0 0 1 1 1 1 1
 ------------------------
        1 0 0 0 0 0 0 0 0
```

$225+31 = 256 \notin [-2^8, 2^8-1] \rightarrow$ overflow!

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

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

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

```
 225 =  0 0 1 1 1 0 0 0 0 1 +
  31 =  0 0 0 0 0 1 1 1 1 1
 --------------------------
 256 =  0 1 0 0 0 0 0 0 0 0
```

$225+31 = 256 \in [-2^9, 2^9-1] \rightarrow$ no overflow

n = 10 bits

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

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

```
 -511 =  1 0 0 0 0 0 0 0 0 1 +
 -167 =  1 1 0 1 0 1 1 0 0 1
 --------------------------
         0 1 0 1 0 1 1 0 1 0
```

$-511-167 = -678 \notin [-2^9, 2^9-1] \rightarrow$ overflow!

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

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

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

```
 -511 =  1 1 0 0 0 0 0 0 0 0 1 +
 -167 =  1 1 1 0 1 0 1 1 0 0 1
 ----------------------------
 -678    1 0 1 0 1 0 1 1 0 1 0
```

$-511-167 = -678 \in [-2^{10}, 2^{10}-1] \rightarrow$ no overflow

n = 10 bits

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

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

```
 -257 =  1 0 1 1 1 1 1 1 1 1 +
  256 =  0 1 0 0 0 0 0 0 0 0
 --------------------------
   -1 =  1 1 1 1 1 1 1 1 1 1
```

$-257+256 = -1 \in [-2^9, 2^9-1] \rightarrow$ no overflow

n = 11 bits

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

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

```
  137 =  0 0 0 1 0 0 0 1 0 0 1 +
  886 =  0 1 1 0 1 1 1 0 1 1 0
 ----------------------------
 1023 =  0 1 1 1 1 1 1 1 1 1 1
```

$137+886 = 1023 \in [-2^{10}, 2^{10}-1] \rightarrow$ no overflow

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

✓ 0101×0111, 0101×1001, 1100×1010

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

        ↓

  0 1 0 0 0 1 1
```

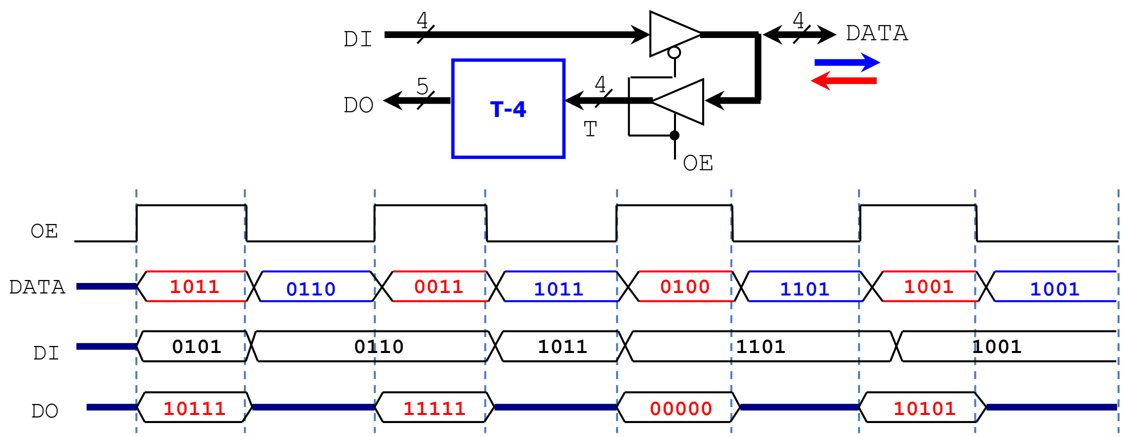
```
  0 1 0 1 x   ➡        0 1 0 1 x
  1 0 0 1              0 1 1 1
                       -------
                       0 1 0 1
                     0 1 0 1
                   0 1 0 1
                 -------------
                 0 1 0 0 0 1 1

                       ↓

                 1 0 1 1 1 0 1
```

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

                       ↓

                 0 1 1 0 0 0
```

PROBLEM 2 (7 PTS)

- Complete the timing diagram (signals DO and $DATA$) of the following circuit. The circuit in the blue box computes the signed operation T-4, with the result having 5 bits. T is a 4-bit signed (2C) number.

  For example: if T=1010 → DO = 1010 – 0100 = 11010 + 11100 = 10110.



PROBLEM 3 (29 PTS)

- In 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. (9 pts.)

✓ -255.6875, 31.625, -128.6875.

- +255.6875 = 011111111.1011 → -255.6875 = 100000000.0101 = 0xF00.5
- +31.625 = 011111.1010 = 0x1F.A
- +128.6875 = 010000000.1011 → -128.6875 = 101111111.0101 = 0xF7F.5

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

Decimal	BCD	Binary	Reflective Gray Code
127	000100100111	1111111	1000000
186	000110000110	10111010	11100111
512	010100010010	1000000000	1100000000
230	001000110000	11100110	10010101
234	001000110100	11101010	10011111
875	100001110101	1101101011	1011011110

c) 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**
-120	11111000	10000111	10001000
-88	11011000	10100111	10101000
465	0111010001	0111010001	0111010001
-64	11000000	10111111	1000000
-15	1001111	10000	10001
-64	11000000	10111111	1000000
-125	11111101	10000010	10000011

PROBLEM 4 (26 PTS)

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

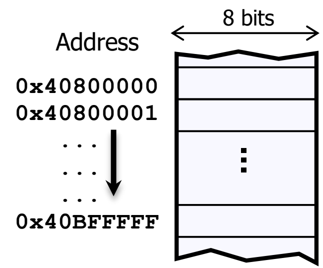
✓ 32678 memory addresses in a computer?	✓ Numbers between 0 and 2048?
✓ $\lceil \log_2 32678 \rceil = 15$	✓ $\lceil \log_2 (2048+1) \rceil = 12$

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

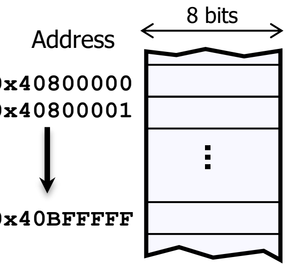
  Address Range: `0x00000000` to `0xFFFFFFFF`.
  With 32 bits, we can address 2^{32} bytes, thus we have $2^2 2^{30}$ = 4GB of address space

- A memory device is connected to the microprocessor. Based on the size of the memory, the microprocessor has assigned the addresses `0x40800000` to `0x40BFFFFF` 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 22 bits for the address in the given range (where the memory device is located).
Thus, the size of the memory device is 2^{22} = 4MB.

```
0100 0000 1000 0000 0000 0000 0000 0000: 0x40800000
0100 0000 1000 0000 0000 0000 0000 0001: 0x40800001
   ...
   ...
   ...
0100 0000 1011 1111 1111 1111 1111 1111: 0x40BFFFFF
```

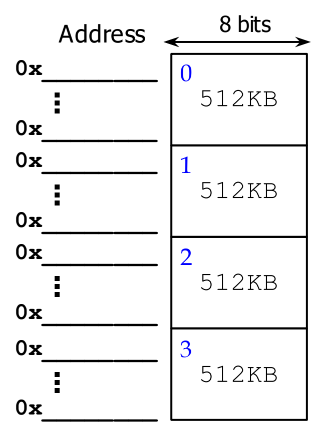


c) A microprocessor has a memory space of 2 MB. The size of the memory contents of each address is 8 bits (1 byte). (7 pts)

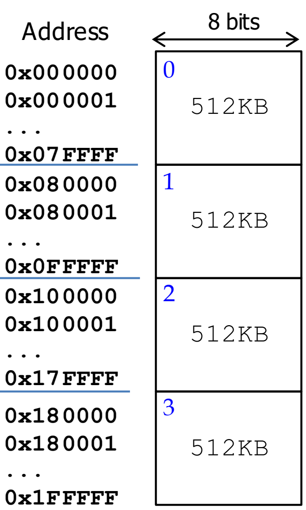
✓ What is the address bus size (number of bits of the address) of this microprocessor?
Since 2 MB = 2^{21} bytes, the address bus size is 21 bits.

✓ What is the range (lowest to highest, in hexadecimal) of the memory space for this microprocessor?
With 21 bits, the address range is 0x000000 to 0x1FFFFF.

✓ The figure (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. (5 pts)




```
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. (11 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 `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}=64 MB.

- If we have a memory chip of 8MB, how many bits do we require to address 8MB of memory? (1 pt.)

  8MB = 2^{23} bytes. Thus, we require 23 bits to address only the memory device.

- We want to connect the 8MB memory chip to the microprocessor. For optimal implementation, we must place those 8MB in an address range where every single address share some MSBs (e.g.: `0x0000000` to `0x07FFFFF`). Provide a list of all the possible address ranges that the 8MB memory chip can occupy. You can only use any of the non-occupied portions of the memory space as shown below. (8 pts)

`0x0800000` to `0x0FFFFFF`	`0x1800000` to `0x1FFFFFF`	`0x2000000` to `0x27FFFFF`	`0x2800000` to `0x2FFFFFF`