Solutions - Homework 2

(Due date: February 4th @ 11:59 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)

- ✓ 341,000 symbols?
  $\lceil \log_2 341{,}000 \rceil = 19$
- ✓ Numbers between (and including) 0 and 8,192?
  $\lceil \log_2 8{,}193 \rceil = 14$

b) A microprocessor has a memory space of 4 MB. Each memory address occupies one byte. (8 pts)

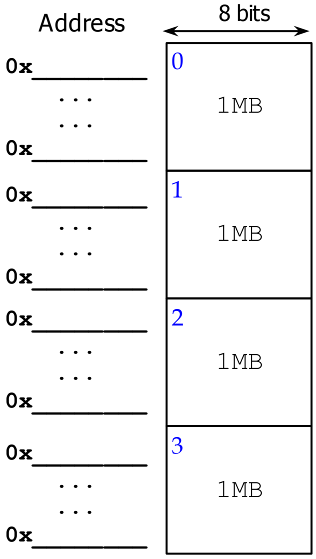
- What is the address bus size (number of bits of the address) of this microprocessor?

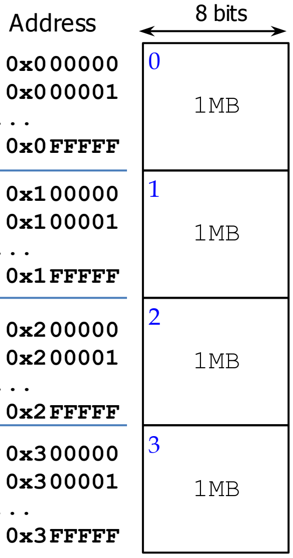
  Since 4MB = 2^{22} bytes, the address bus size is 22 bits.

- What is the range (lowest to highest, in hexadecimal) of the memory space for this microprocessor?

  With 22 bits, the address range is `0x000000` to `0x3FFFFF`.

- 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.



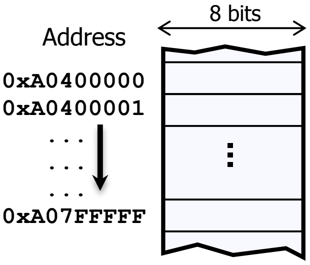


```
00 0000 0000 0000 0000 0000: 0x000000
00 0000 0000 0000 0000 0001: 0x000001
   ...                       ...
00 1111 1111 1111 1111 1111: 0x0FFFFF

01 0000 0000 0000 0000 0000: 0x100000
01 0000 0000 0000 0000 0001: 0x100001
   ...                       ...
01 1111 1111 1111 1111 1111: 0x1FFFFF

10 0000 0000 0000 0000 0000: 0x200000
10 0000 0000 0000 0000 0001: 0x200001
   ...                       ...
10 1111 1111 1111 1111 1111: 0x2FFFFF

11 0000 0000 0000 0000 0000: 0x300000
11 0000 0000 0000 0000 0001: 0x300001
   ...                       ...
11 1111 1111 1111 1111 1111: 0x3FFFFF
```

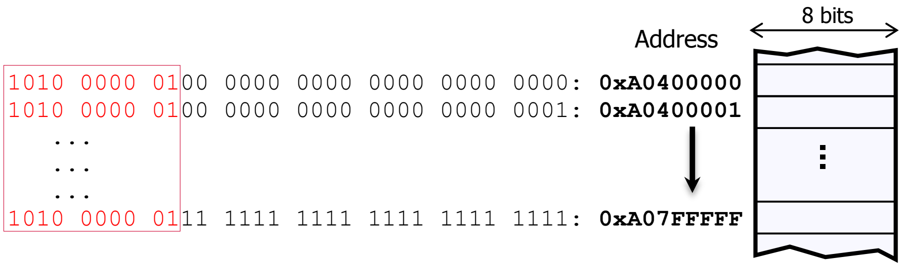
c) A microprocessor has a 32-bit address line. The size of the memory contents of each address is 8 bits. The memory space is defined as the collection of memory positions the processor can address. (6 pts)

- 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. (2 pts)
  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 `0xA0400000` to `0xA07FFFFF` 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.



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.

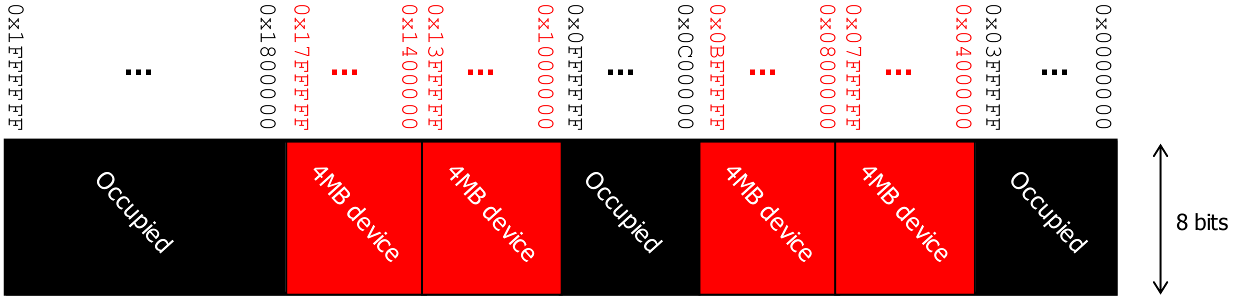
- If we have a memory chip of 4MB, how many bits do we require to address 4MB of memory? (2 pts)

  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 any of the non-occupied portions of the memory space as shown below. (8 pts)

  The 22-bit address range for an 4MB memory would be: `0x0000000` to `0x03FFFFF`. To place this range within the 25-bit memory space in the figure, we have four options:

`0x0400000` to `0x07FFFFF`	`0x0800000` to `0x0BFFFFF`	`0x1000000` to `0x13FFFFF`	`0x1400000` to `0x17FFFFF`



PROBLEM 2 (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)

✓ `101.3125, -64.6875, -31.65625.`

- `101.3125 = 01100101.0101 = 0x65.5`
- `+64.6875 = 01000000.1011 → -64.6875 = 1110111111.0101 = 0xFBF.5`
- `31.65625 = 011111.10101 → -31.6525 = 100000.01011 = 0xE0.51`

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

Decimal	BCD	Binary	Reflective Gray Code
278	001001111000	100010110	110011101
171	000101110001	10101011	11111110
217	001000010111	11011001	10110101
186	000110000110	10111010	11100111
265	001001100101	100001001	110001101
957	100101010111	1110111101	1001100011

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**
-257	1100000001	1011111110	1011111111
-64	11000000	10111111	1000000
-256	1100000000	1011111111	100000000
-39	1100111	1011000	1011001
145	010010001	010010001	010010001
-128	110000000	101111111	10000000
-125	11111101	10000010	10000011

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 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
```

- ✓ 173 + 75
- ✓ 69 + 211

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

173 = 0xAD =   1 0 1 0 1 1 0 1 +
 75 = 0x4B =   0 1 0 0 1 0 1 1
264 = 0xF8 =   1 1 1 1 1 0 0 0
```

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

 69 = 0x45 =   0 1 0 0 0 1 0 1 +
211 = 0xD3 =   1 1 0 1 0 0 1 1
Overflow! →  1 0 0 0 1 1 0 0 0
```

- ✓ 87 - 256
- ✓ 241 - 37

```
Borrow out! →  b9=1 b8=0 b7=0 b6=0 b5=0 b4=0 b3=0 b2=0 b1=0 b0=0

 87 = 0x057 =   0 0 1 0 1 0 1 1 1 -
256 = 0x100 =   1 0 0 0 0 0 0 0 0
      0x157 =   1 0 1 0 1 0 1 1 1
```

```
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 -
 37 = 0x25 =   0 0 1 0 0 1 0 1
204 = 0xCC =   1 1 0 0 1 1 0 0
```

b) We need to perform the following operations, where numbers are represented in 2's complement: (16 pts)

- ✓ -87 + 256
- ✓ 490 + 22
- ✓ -35 + 65
- ✓ -255 - 230

- 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

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

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

n = 8 bits

$c_8 \oplus c_7 = 0$
No Overflow

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

 -35 = 1 1 0 1 1 1 0 1 +
  65 = 0 1 0 0 0 0 0 1
  30 = 0 0 0 1 1 1 1 0
```

$-35+65 = 30 \in [-2^7, 2^7-1]$ → no overflow

n = 10 bits

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

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

```
490 = 0 1 1 1 1 0 1 0 1 0 +
 22 = 0 0 0 0 0 1 0 1 1 0
      -------------------
      1 0 0 0 0 0 0 0 0 0
```

$490+22 = 512 \notin [-2^9,\ 2^9-1] \rightarrow$ overflow!

To avoid overflow:

n = 11 bits (sign-extension)

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

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

```
490 = 0 0 1 1 1 1 0 1 0 1 0 +
 22 = 0 0 0 0 0 0 1 0 1 1 0
      ---------------------
512 = 0 1 0 0 0 0 0 0 0 0 0
```

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

n = 9 bits

$c_9 \oplus c_8 = 1$ Overflow!

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

```
-255 = 1 0 0 0 0 0 0 0 1 +
-230 = 1 0 0 0 1 1 0 1 0
       -----------------
       0 0 0 0 1 1 0 1 1
```

$-255-230 = -485 \notin [-2^8,\ 2^8-1] \rightarrow$ overflow!

To avoid overflow:

n = 10 bits (sign-extension)

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

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

```
-255 = 1 1 0 0 0 0 0 0 0 1 +
-230 = 1 1 0 0 0 1 1 0 1 0
       -------------------
-485 = 1 0 0 0 0 1 1 0 1 1
```

$-255-230 = -485 \in [-2^9,\ 2^9-1] \rightarrow$ no overflow

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

- ✓ `0111×0110, 1100×0101, 1011×1010.`

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

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

PROBLEM 4 (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-7, with the result having 5 bits. T is a 4-bit signed (2C) number.
- For example: if T=1010 → DO=1010-0111 = 11010 + 11001 = 10011