Solutions - Homework 2

(Due date: October 6th @ 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)

- ✓ 16385 symbols?
  $\lceil log_2 16385 \rceil = 15$
- ✓ Memory addresses from 0 to 131072?
  $\lceil log_2(131073) \rceil = 18$

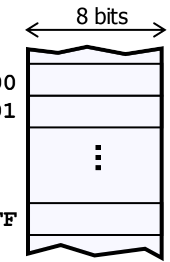
b) A microprocessor has a 32-bit address line. The size of the memory contents of each address is 8 bits. The memory space is defined as the collection of memory positions the processor can address. (6 pts)

- 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 (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 `0xFE400000` to `0xFE7FFFFF` 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 is 2^{22} = 4 MB.

```
                                                  Address
1111 1110 0100 0000 0000 0000 0000 0000: 0xFE400000
1111 1110 0100 0000 0000 0000 0000 0001: 0xFE400001
   ...
   ...
   ...
1111 1110 0111 1111 1111 1111 1111 1111: 0xFE7FFFFF
```

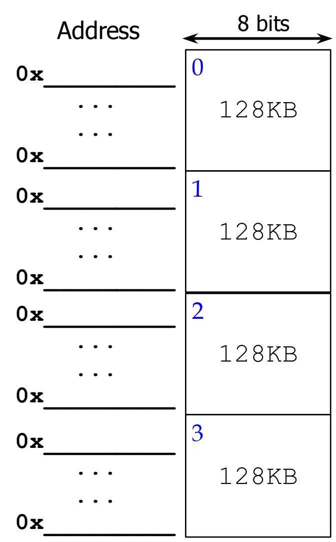


c) A microprocessor has a memory space of 512 KB. Each memory address occupies one byte. (8 pts)

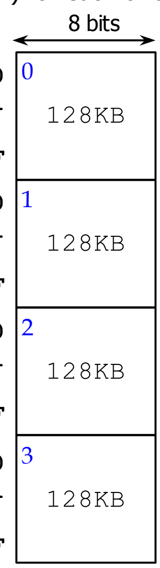
- What is the address bus size (number of bits of the address) of this microprocessor?
  Since 512 KB = 2^{19} bytes, the address bus size is 19 bits.

- What is the range (lowest to highest, in hexadecimal) of the memory space for this microprocessor?
  With 19 bits, the address range is `0x00000` to `0x7FFFF`.

- 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
000 0000 0000 0000 0000: 0x00000
000 0000 0000 0000 0001: 0x00001
  ...                      ...
001 1111 1111 1111 1111: 0x1FFFF
-----------------------------------
010 0000 0000 0000 0000: 0x20000
010 0000 0000 0000 0001: 0x20001
  ...                      ...
011 1111 1111 1111 1111: 0x3FFFF
-----------------------------------
100 0000 0000 0000 0000: 0x40000
100 0000 0000 0000 0001: 0x40001
  ...                      ...
101 1111 1111 1111 1111: 0x5FFFF
-----------------------------------
110 0000 0000 0000 0000: 0x60000
110 0000 0000 0000 0001: 0x60001
  ...                      ...
111 1111 1111 1111 1111: 0x7FFFF
```



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?

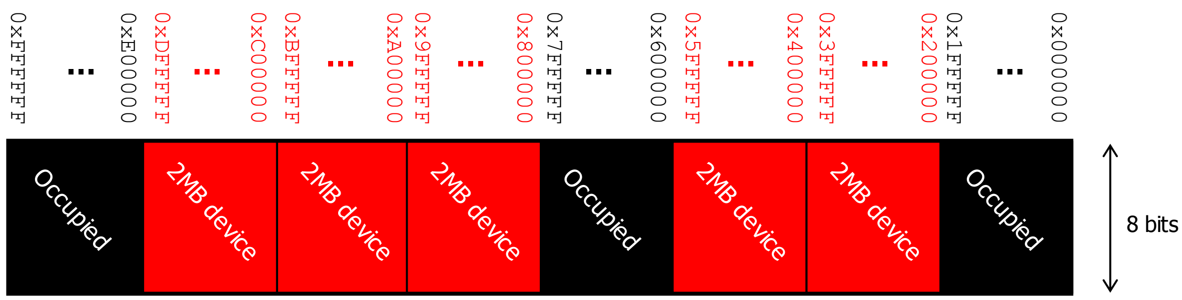
  Address space: `0x000000` to `0xFFFFFF`. To represent all these addresses, we require 24 bits. So, the address bus size of the microprocessor is 24 bits. The size of the memory space is then 2^{24} = 16 MB.

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

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

- We want to connect the 2MB memory chip to the microprocessor. Recall that a memory chip must be placed in an address range where every single address shares some MSBs (e.g.: `0x600000` to `0x7FFFFF`). Provide a list of all the possible address ranges that the 2MB memory chip can occupy. You can only use any of the non-occupied portions of the memory space as shown below. (8 pts)

```
0x200000 to 0x3FFFFF
0x400000 to 0x5FFFFF
0x800000 to 0x9FFFFF
0xA00000 to 0xBFFFFF
0xC00000 to 0xDFFFFF
```



PROBLEM 2 (28 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. (8 pts)

✓ `-136.6875, 207.65625, -128.5078125`

- `207.65625 = 011001111.10101 = 0x0CF.A8`
- `+128.5078125 = 010000000.1000001 → -128.5078125 = 101111111.0111111 = 0xF7F.7E`

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

Decimal	BCD	Binary	Reflective Gray Code
278	00100111	100010110	110011101
171	000101110001	10101011	11111110
507	010100000111	111111011	100000110
995	100110010101	1111100011	1000010010
217	001000010111	11011001	10110101
84	10000100	01010100	01111110

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**
-129	110000001	101111110	101111111
84	01010100	01010100	01010100
-88	11011000	10100111	10101000
0	00	11111	0
-64	11000000	10111111	1000000
-39	1100111	1011000	1011001
-1	11	10	11111

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

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

✓ 209 + 68
✓ 34 + 437

✓ 129 - 37
✓ 42 - 98

n = 8 bits
Overflow!

	$c_8=1$	$c_7=1$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$	
209 = 0xD1 =		1	1	0	1	0	0	0	1	+
68 = 0x44 =		0	1	0	0	0	1	0	0	
277 =0x115 =	1	0	0	0	1	0	1	0	1	

n = 8 bits
No Borrow Out

	$b_8=0$	$b_7=1$	$b_6=1$	$b_5=1$	$b_4=1$	$b_3=1$	$b_2=0$	$b_1=0$	$b_0=0$	
129 = 0x81 =		1	0	0	0	0	0	0	1	-
37 = 0x25 =		0	0	1	0	0	1	0	1	
92 = 0x5C =		0	1	0	1	1	1	0	0	

n = 9 bits
No Overflow

	$c_9=0$	$c_8=0$	$c_7=0$	$c_6=1$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$	
437 = 0x1B5 =		1	1	0	1	1	0	1	0	1	+
34 = 0x022 =		0	0	0	1	0	0	0	1	0	
471 = 0x1D7 =		1	1	1	0	1	0	1	1	1	

n = 7 bits

Borrow out! →	$b_7=1$	$b_6=0$	$b_5=0$	$b_4=0$	$b_3=0$	$b_2=0$	$b_1=0$	$b_0=0$	
42 = 0x2A =		0	1	0	1	0	1	0	-
98 = 0x62 =		1	1	0	0	0	1	0	
0x48 =		1	0	0	1	0	0	0	

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

✓ -98 + 256
✓ 206 + 309
✓ -257 + 256

✓ 105 - 62
✓ -127 - 36
✓ 246 + 31

- 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

	$c_{10}=1$	$c_9=1$	$c_8=0$	$c_7=0$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$	
-98 =		1	1	1	0	0	1	1	1	1	0	+
256 =		0	1	0	0	0	0	0	0	0	0	
158 =		0	0	1	0	0	1	1	1	1	0	

-98+256 = 158 ∈ $[-2^9, 2^9-1]$ → no overflow

n = 8 bits
$c_8 \oplus c_7=0$
No Overflow

	$c_8=1$	$c_7=1$	$c_6=0$	$c_5=0$	$c_4=0$	$c_3=0$	$c_2=0$	$c_1=0$	$c_0=0$	
105 =		0	1	1	0	1	0	0	1	+
-62 =		1	1	0	0	0	0	1	0	
43 =		0	0	1	0	1	0	1	1	

105=62 = 43 ∈ $[-2^7, 2^7-1]$ → no overflow

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=0 c1=0 c0=0

206 =   0 0 1 1 0 0 1 1 1 0 +
309 =   0 1 0 0 1 1 0 1 0 1
        -------------------
        1 0 0 0 0 0 0 0 1 1
```

206+309 = 515 $\notin [-2^9, 2^9-1]$ → 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=1

206 = 0 0 0 1 1 0 0 1 1 1 0 +
309 = 0 0 1 0 0 1 1 0 1 0 1
      ---------------------
515 = 0 1 0 0 0 0 0 0 0 1 1
```

206+309 = 515 $\in [-2^{10}, 2^{10}-1]$ → 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=0 c1=0 c0=0

-127 = 1 0 0 0 0 0 0 1 +
 -36 = 1 1 0 1 1 1 0 0
       ---------------
       0 1 0 1 1 1 0 1
```

-127-36 = -163 $\notin [-2^7, 2^7-1]$ → 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=0 c1=0 c0=0

-127 = 1 1 0 0 0 0 0 0 1 +
 -36 = 1 1 1 0 1 1 1 0 0
       -----------------
-163 = 1 0 1 0 1 1 1 0 1
```

-127-36 = -163 $\in [-2^8, 2^8-1]$ → no overflow

n = 10 bits

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

```
       c10=0 c9=0 c8=0 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=0 c0=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 $\in [-2^9, 2^9-1]$ → no overflow

n = 9 bits

$c_9 \oplus c_8 = 1$ Overflow!

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

246 = 0 1 1 1 1 0 1 1 0 +
 31 = 0 0 0 0 1 1 1 1 1
      -----------------
277 = 1 0 0 0 1 0 1 0 1
```

246+31 = 277 $\notin [-2^8, 2^8-1]$ → overflow!

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

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

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

246 = 0 0 1 1 1 1 0 1 1 0 +
 31 = 0 0 0 0 0 1 1 1 1 1
      -------------------
277 = 0 1 0 0 0 1 0 1 0 1
```

246+31 = 277 $\in [-2^9, 2^9-1]$ → no overflow

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

✓ 1001x1011, 1010x0101, 1110x0110.

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

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

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

PROBLEM 4 (6 PTS)

- Complete the timing diagram (signals DO and $DATA$) of the following circuit. The circuit in the blue box is a 4-bit Binary to Gray Decoder. For example, if T=1100, then DO=1010.