6.004 Fall 2019 Tutorial Problems
L01 – Binary Encoding and Arithmetic

Problem 1. Encoding positive integers

1. What is the 5-bit binary representation of the decimal number 21?

   10101

2. What is the hexadecimal representation for decimal 219 encoded as an 8-bit binary number?

   1101_1011 → 0xDB

3. What is the hexadecimal representation for decimal 51 encoded as a 6-bit binary number?

   110011 → 0x33

4. The hexadecimal representation for an 8-bit unsigned binary number is 0x9E. What is its decimal representation?

   158

5. What is the range of integers that can be represented with a single unsigned 8-bit quantity?

   0 - 255

6. Since the start of official pitching statistics in 1988, the highest number of pitches in a single game has been 172. Assuming that remains the upper bound on pitch count, how many bits would we need to record the pitch count for each game as an unsigned binary number?

   ceil(log2(172)) = 8

7. Compute the sum of these two 4-bit unsigned binary numbers. Express the result in hexadecimal.

```
  1101
+ 0110
 10011   == 0x13
```

Problem 2. Two's complement representation

1. What is the 6-bit two's complement representation of the decimal number -21?

   21 = 16 + 4 + 1 = 0b010101, (the prefix 6b denotes using 6 bit binary)
   -21 = 0b101010 + 1 = **0b101011**

2. What is the hexadecimal representation for decimal -51 encoded as an 8-bit two's complement number?

   51 = 32+16+2+1 = 0b0011_0011, (using 8-bit binary)
   -51 = 0b1100_1101 = **0xCD**

3. The hexadecimal representation for an 8-bit two's complement number is 0xD6. What is its decimal representation?

   0xD6 = 0b1101_0110 = -128+64+16+4+2 = **-42**

   Alternative (*may* be easier):
   0xD6 = 0b1101_0110, which is negative, so use -0xD6 = 0b0010_1010 = 42, which gives +0xD6 = **-42**

4. Using a 5-bit two's complement representation, what is the range of integers that can be represented with a single 5-bit quantity?

   -2^4 to (2^4)-1
   -16 to 15

5. Can the value of the sum of two 2's complement numbers 0xB3 + 0x47 be represented using an 8-bit 2's complement representation? If so, what is the sum in hex? If not, write NO.

   Yes: negative + positive is always within range.
   0xB3 + 0x47 =
   0b1011_0011 +
   0b0100_0111 =
   0b1111_1010 = **0xFA** (in decimal: -6, can you see why it's a very small negative?)

6. Can the value of the sum of two 2's complement numbers 0xB3 + 0xB1 be represented using an 8-bit 2's complement representation? If so, what is the sum in hex? If not, write NO.

   No: negative + negative gives us positive, cannot be represented in 8 bits.
   0xB3 + 0xB1 =
   0b1011_0011 +
   0b1011_0001 =
   0b0110_0100 :((("9th bit" is dropped, we only have 8 bits)

7. Please compute the value of the expression 0xBB – 8 using 8-bit two's complement arithmetic and give the result in decimal (base 10).

   0xBB - 8 = 8b1011_1011 + -0b0000_1000 =
   0b1011_1011 +
   0b1111_1000 =
   0b1011_0011 =
   -77 (negative, so positive is 0b0100_1101 = 77) =

   Note that negative - positive is always within range (same as positive-negative).

8. Consider the following subtraction problem where the operands are 5-bit two's complement numbers. Compute the result and give the answer as a decimal (base 10) number.

$$\begin{array}{r} 10101 \\ -\underline{00011} \end{array}$$

```
  0b10101      0b10101          0b10101
- 0b00011 => + 0b11100 + 1 => + 0b11101
                              = 0b10010 = -(0b01101 + 1) = -14
```

Problem 3. Multiples of 4

1. Given an unsigned n-bit binary integer $= \boldsymbol{b_{n-1}} \ldots \boldsymbol{b_1 b_0}$, prove that $\boldsymbol{v}$ is a multiple of 4 if and only if $\boldsymbol{b_0 = 0}$ and $\boldsymbol{b_1 = 0}$.

   To solve this problem, remember that $\boldsymbol{v = \sum_{k=0}^{n-1} b_k 2^k}$

   - Powers of 2 greater than or equal to 4 are multiples of 4 (for all k>=2, 2^k = 4*2^(k-2) and 2^(k-2) is an integer)
   - The sum of numbers divisible by 4 is divisible by 4.
     Therefore, any number of the form $b_{n-1} \ldots 00$ is a multiple of 4.

   If a number ends in one of 01, 10, or 11 we are adding 1, 2, or 3 respectively to a multiple of 4. Therefore, a number is a multiple 4 only if it ends in 00.

2. Does the same relation hold for two's complement encoding?

   Yes. The above proof still works: in two's complement notation the highest-order bit uses base -2^(n-1) instead of +2^(n-1), and -2^(n-1) is divisible by 4.

Problem 4. Encoding text

There are multiple standards to encode characters and strings using binary values. ASCII is a classic standard to encode English alphabet characters (modern formats like UTF support other alphabets, but are typically based on ASCII). ASCII encodes each character using an 8-bit (1-byte) value. The table below shows ASCII's mapping of characters to values.

ASCII (1977/1986), adapted from https://en.wikipedia.org/wiki/ASCII

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL 0x00	SOH 0x01	STX 0x02	ETX 0x03	EOT 0x04	ENQ 0x05	ACK 0x06	BEL 0x07	BS 0x08	HT 0x09	LF 0x0A	VT 0x0B	FF 0x0C	CR 0x0D	SO 0x0E	SI 0x0F
1_	DLE 0x10	DC1 0x11	DC2 0x12	DC3 0x13	DC4 0x14	NAK 0x15	SYN 0x16	ETB 0x17	CAN 0x18	EM 0x19	SUB 0x1A	ESC 0x1B	FS 0x1C	GS 0x1D	RS 0x1E	US 0x1F
2_	space 0x20	! 0x21	" 0x22	# 0x23	$ 0x24	% 0x25	& 0x26	' 0x27	(0x28	) 0x29	* 0x2A	+ 0x2B	, 0x2C	- 0x2D	. 0x2E	/ 0x2F
3_	0 0x30	1 0x31	2 0x32	3 0x33	4 0x34	5 0x35	6 0x36	7 0x37	8 0x38	9 0x39	: 0x3A	; 0x3B	< 0x3C	= 0x3D	> 0x3E	? 0x3F
4_	@ 0x40	A 0x41	B 0x42	C 0x43	D 0x44	E 0x45	F 0x46	G 0x47	H 0x48	I 0x49	J 0x4A	K 0x4B	L 0x4C	M 0x4D	N 0x4E	O 0x4F
5_	P 0x50	Q 0x51	R 0x52	S 0x53	T 0x54	U 0x55	V 0x56	W 0x57	X 0x58	Y 0x59	Z 0x5A	[0x5B	\ 0x5C	] 0x5D	^ 0x5E	_ 0x5F
6_	` 0x60	a 0x61	b 0x62	c 0x63	d 0x64	e 0x65	f 0x66	g 0x67	h 0x68	i 0x69	j 0x6A	k 0x6B	l 0x6C	m 0x6D	n 0x6E	o 0x6F
7_	p 0x70	q 0x71	r 0x72	s 0x73	t 0x74	u 0x75	v 0x76	w 0x77	x 0x78	y 0x79	z 0x7A	{ 0x7B	| 0x7C	} 0x7D	~ 0x7E	DEL 0x7F

Letter Number Punctuation Symbol Other/non-printable

Computers often store variable-length text as a null-terminated string: a sequence of bytes, where each byte denotes a different character, terminated by the value 0x00 (null) to denote the end of the string. For example, the string "6.004" is encoded as the 6-byte sequence 0x362E30303400.

1. Encode your name as a null-terminated ASCII string (use the best approximation if your name contains non-English characters)

0x 59 6F 75 72 20 4E 61 6D 65 20 48 65 72 65 20 3A 29 00

2. Decode the following null-terminated ASCII string:

0x 52 49 53 43 2D 56 20 69 73 20 63 6F 6D 69 6E 67 21 00

RISC-V is coming!