PROBLEM SET 3 - SOLUTIONS
W1004 Spring 2014

Exercises:
Chapter 4: 1, 11, 19, 25, 26
Chapter 5: 2, 18, 19, 21, Challenge Work 1

4.1 (6 points)
a. $(133)_4 = 1 \times 4^2 + 3 \times 4^1 + 3 \times 4^0 = \mathbf{31}$
b. $(367)_8 = 3 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 = \mathbf{247}$
c. $(1BA)_{16} = 1 \times 16^2 + 11 \times 16^1 + 10 \times 16^0 = \mathbf{442}$

4.11 (6 points)
a. 0 111000000 0 00111 = $+ .111000000 \times 2^{00111} = 0.875 \times 2^7 = \mathbf{112}$
b. 1 010001000 1 00001 = $-.010001000 \times 2^{-00001} = 0.265625 \times 2^{-1} =$ **-0.1328125**

Note: Is there something unusual about this representation? If so, what is it?
Although this is the correct answer, the mantissa is not normalized. In most computers this value would be normalized so the first digit of the mantissa is a 1, and the exponent would be adjusted accordingly.

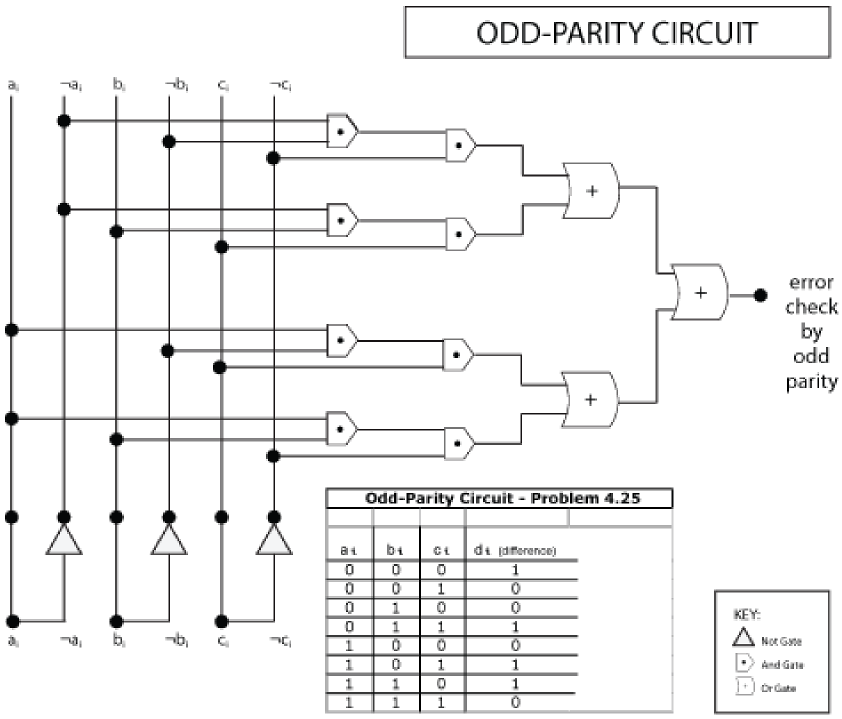
4.19 (12 points)
a. True
b. True
c. False
d. False
e. False

4.25 (16 points)

a	**b**	**c**	**Output (F)**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1

1	1	0	1
1	1	1	0

$$F = \bar{a}.\bar{b}.\bar{c} + \bar{a}.b.c + a.\bar{b}.c + a.b.\bar{c}$$



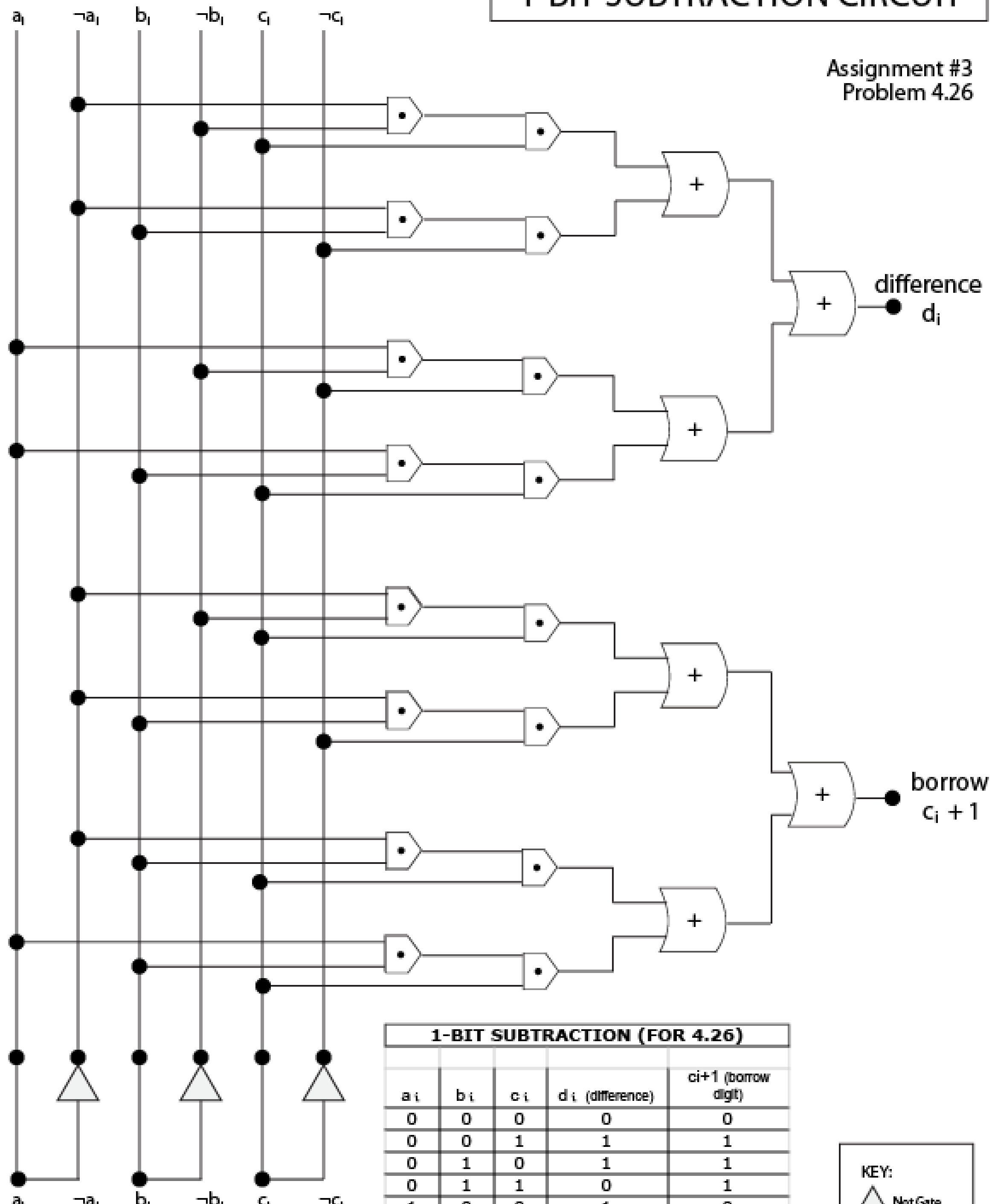
4.26 (16 points)

a_i	b_i	c_i	d_i	c_{i+1} (borrow)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0

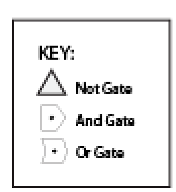
1	1	1	1	1

1-BIT SUBTRACTION CIRCUIT

Assignment #3
Problem 4.26



1-BIT SUBTRACTION (FOR 4.26)				
a_i	b_i	c_i	d_i (difference)	ci+1 (borrow digit)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



5.2 (8 points)
2^N bytes requires N bits.

a) 1 million bytes: lg(1000000) = 19.93 = **20 bits**
b) 10 million bytes: lg(10000000) = 23.25 = **24 bits**
c) 100 million bytes: lg(100000000) = 26.57 = **27 bits**
d) 1 billion bytes: lg(1000000000) = 29.89 = **30 bits**

5.18 (6 points)

500 MIPS / 56,000 bits/second = 500,000,000 instructions/second * 1/56,000 seconds/bit = 8928.57 instructions/bit = **8928 instructions/bit**

5.19 (6 points)

op code	address 1	address 2
6 bits	18 bits	18 bits

a. Maximum number of op codes: 2^6 = **64**
b. Maximum memory size = 2^18 bytes = **262144 bytes = 256 KB**
c. Bytes required for each operation: 6 + 18 + 18 bits = 42 bits = 5.25 bytes = **6 bytes used per operation**

5.21 (16 points)

a.	Memory Location	Op Code	Address Field	Comment
	50	LOAD	202	Register R now contains the value of x
	51	SUBTRACT	203	R now contains the x - y
	52	ADD	204	R now contains the value of x - y + z
	53	STORE	200	Store the result into v
	54			The next instruction begins here
b	Memory Location	Op Code	Address Field	Comment
	50	LOAD	201	Register R now contains the value

				of w
	51	ADD	202	R now contains sum w + x
	52	STORE	201	Store the result in 201
	53	LOAD	203	Register R now contains the value of z.
	54	ADD	204	R now contains the value of w + x - y
	55	STORE	204	R now contains the value of w + x - y - z
	56	LOAD	201	Load the result stored in 201
	57	SUBTRACT	204	R now contains the difference
	58	STORE	200	Store the result into v
c	Memory Location	Op Code	Address Field	Comment
	50	COMPARE	200, 201	Compare v and w and set condition codes
	51	JUMPLT	55	Jump to address 55 if v < w
	52	LOAD	203	Load R with the value of y
	53	STORE	202	And store it into x
	54	JUMP	57	Jump to address 57
	55	LOAD	204	Load R with the value of z
	56	STORE	202	And store that result into x
	57			The next instruction begins here
d	Memory Location	Op Code	Address Field	Comment
	50	COMPARE	203, 204	Compare y and z and set condition codes
	51	JUMPEQ	61	Jump to end if y = z

	52	JUMPGT	61	Jump to end if y > z
	53	LOAD	203	Load R with the value of y
	54	ADD	201	R now contains sum y + w
	55	ADD	204	R now contains sum y + w + z
	56	STORE	203	Store the result back into y
	57	LOAD	204	Load R with the value of z
	58	ADD	200	R now contains sum z + v
	59	STORE	204	Store the result into z
	60	JUMP	50	Jump back to test loop condition again
	61			The next instruction begins here

Challenge Work 1: (8 points)

- Break the algorithm into 50 pairs, and then have 50 processors compute the sums of each of these pairs. This leaves 50 new numbers.
- Split the resulting numbers into 25 pairs, and use 25 processors to find the sums of these pairs.
- Leave behind one number (call it X), split the remaining 24 into 12 pairs, and use 12 processors to compute the sums of each of the pairs.
- Split the 12 resulting numbers into 6 pairs and compute the sums of each pair using 6 processors.
- Split the 6 resulting numbers into 3 pairs, and use 3 processors to compute the sums of these pairs.
- Pair up two of the remaining 3 numbers, and pair the 3rd number with X (previously leftover) to give 2 pairs, and use two processors to compute the sums of these pairs.
- Use one last processor to compute the final sum.

This algorithm takes *7 units of time*, since in each step listed above, all the processors work together, and there are 7 steps. Compare this to the sequential algorithm, which takes 99 units of time (it’s fine if they say 100). Clearly, the parallel algorithm is much faster.

This algorithm only uses 50 processors at most, since all the steps after the first one can just reuse the previous processors.
It’s not possible to use more than 50 processors for a parallel algorithm for this specific problem (and hence not more than 100).