Java Problem Set 3

Name: Shikhar Bakhda
UNI: ssb2189

Chapter 4

15.a

$$bits = \frac{40000\ samples}{sec} \times \frac{16\ bits}{sample} \times 180$$

$$\boxed{= 115{,}200{,}000\ bits}$$

If compressed, the number of bits needed will be as follows:

$$\frac{5}{1} = \frac{uncompressed\ bits}{compressed\ bits} = \frac{115{,}200{,}000}{compressed\ bits}$$

$$compressed\ bits = \frac{115200000}{5}$$

$$\boxed{= 23{,}040{,}000\ bits}$$

15.b

$$uncompressed\ bits = \frac{(1200 \times 800)\ pixels}{\text{color}} \times \frac{3\ colors}{pixel} \times \frac{8\ bits}{color}$$

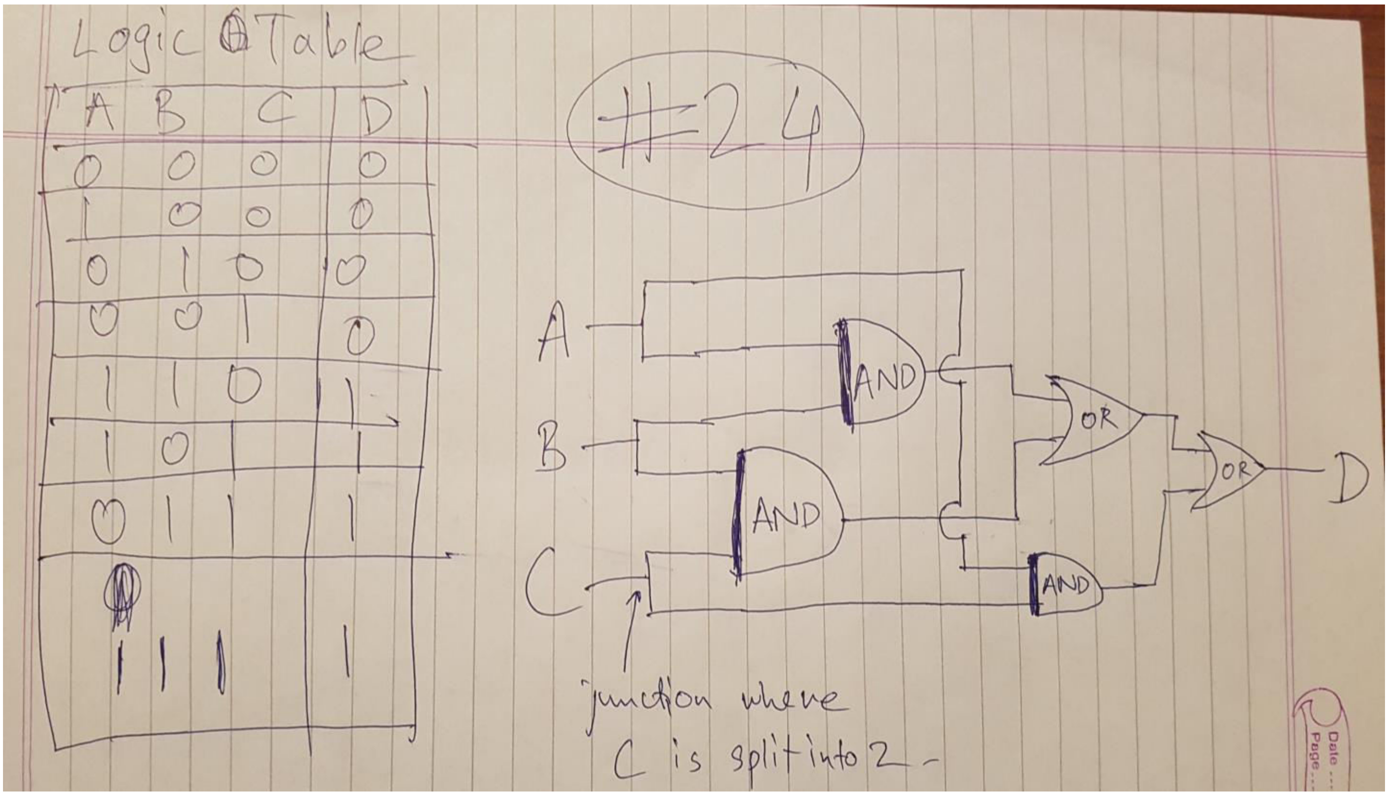
$$\boxed{= 23{,}040{,}000\ bits}$$

$$compression\ ratio = \frac{uncompressed\ bits}{compressed\ bits} = \frac{23040000}{2.4\ \times\ 10^6} = \frac{9.6}{1}$$

$$\boxed{\text{compression ratio} = 96 : 10}$$

24

At least 2 inputs out of 3 must be true. If the inputs were A, B, and C, the possible combinations where at least 2 inputs are true are AB, BC, and AC. If either combinations are met, the output must be true. Thus I decided to put AND gates between A and B, B and C, and A and C. OR gates connect to the output of the AND gates. The diagram below shows the circuit with inputs A, B and C, and the output D.



26

Here is the truth table for the binary subtraction circuit. *a* and *b* are the two numbers. *c* is the borrow digit from the previous column. diff = *(a-b)* is the difference, and *e* is the new borrow digit that propagates to the next column.

a	b	c	diff	e
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

Now, we will figure out the circuit using logical notation.
Finding (a-b):

$$\text{diff} = a'b'c + a'bc' + ab'c' + abc$$
$$= c(a'b' + ab) + c'(a'b + ab')$$

We can show that

$$(ab + a'b')' = (ab)'(a'b')' = (a' + b')(a + b)$$
$$= aa' + a'b + ab' + bb'$$
$$= a'b + ab'$$

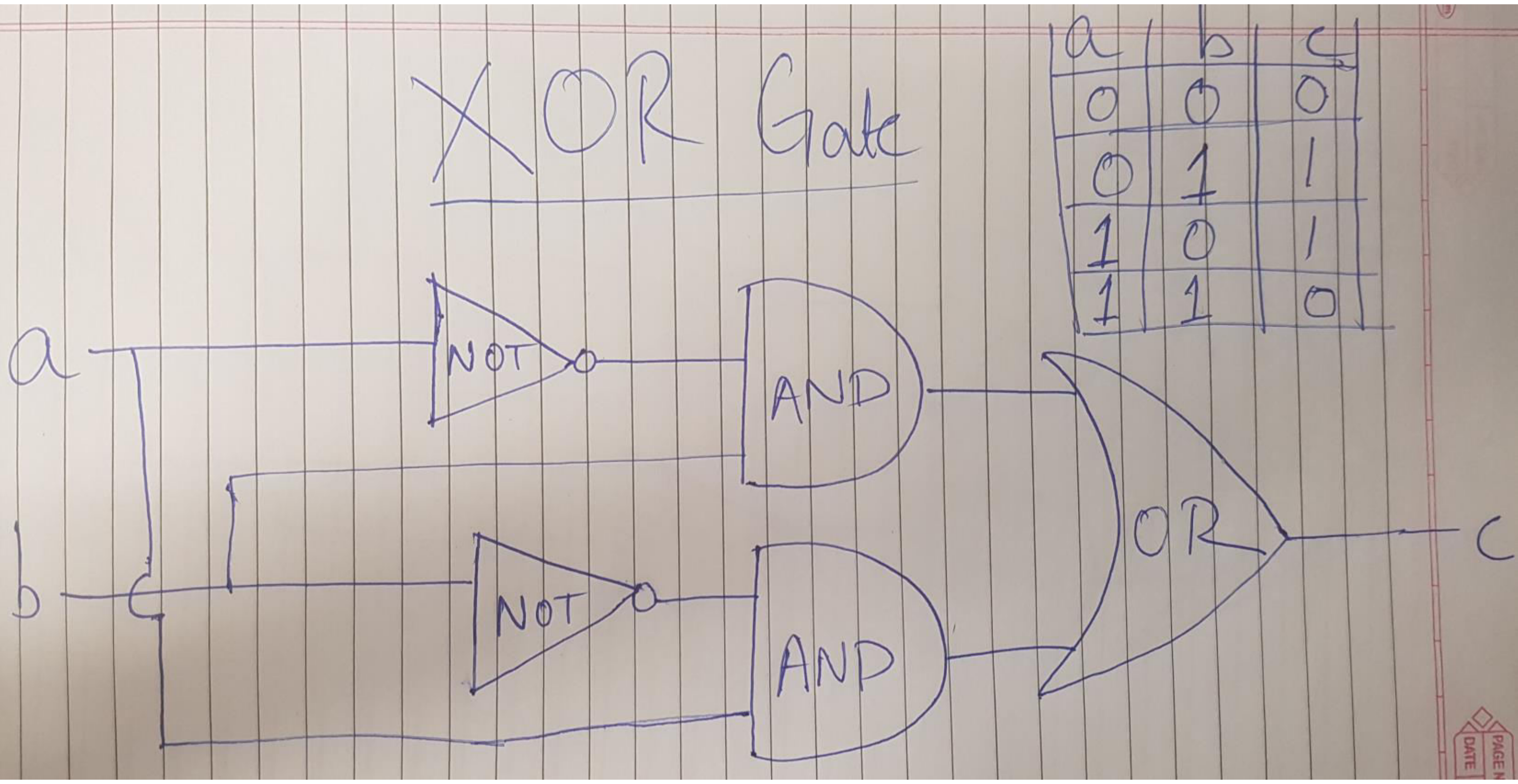
Thus, we substitute $(a'b' + ab)$ with $(a'b + ab')'$

$$\text{diff} = c(a'b + ab')' + c'(a'b + ab')$$

The structure AB' + A'B is that of the XOR circuit, which looks like the following, and has the truth table for any inputs *a* and *b* and ouput *c*:

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Here is the circuit for XOR:



We thus deduce that

$$a'b + ab' = a\ XOR\ b$$

and

$$c(ab' + a'b)' + c'(a'b + ab') = c\ XOR\ (a'b + ab')$$

Thus, we obtain the following answer for diff:

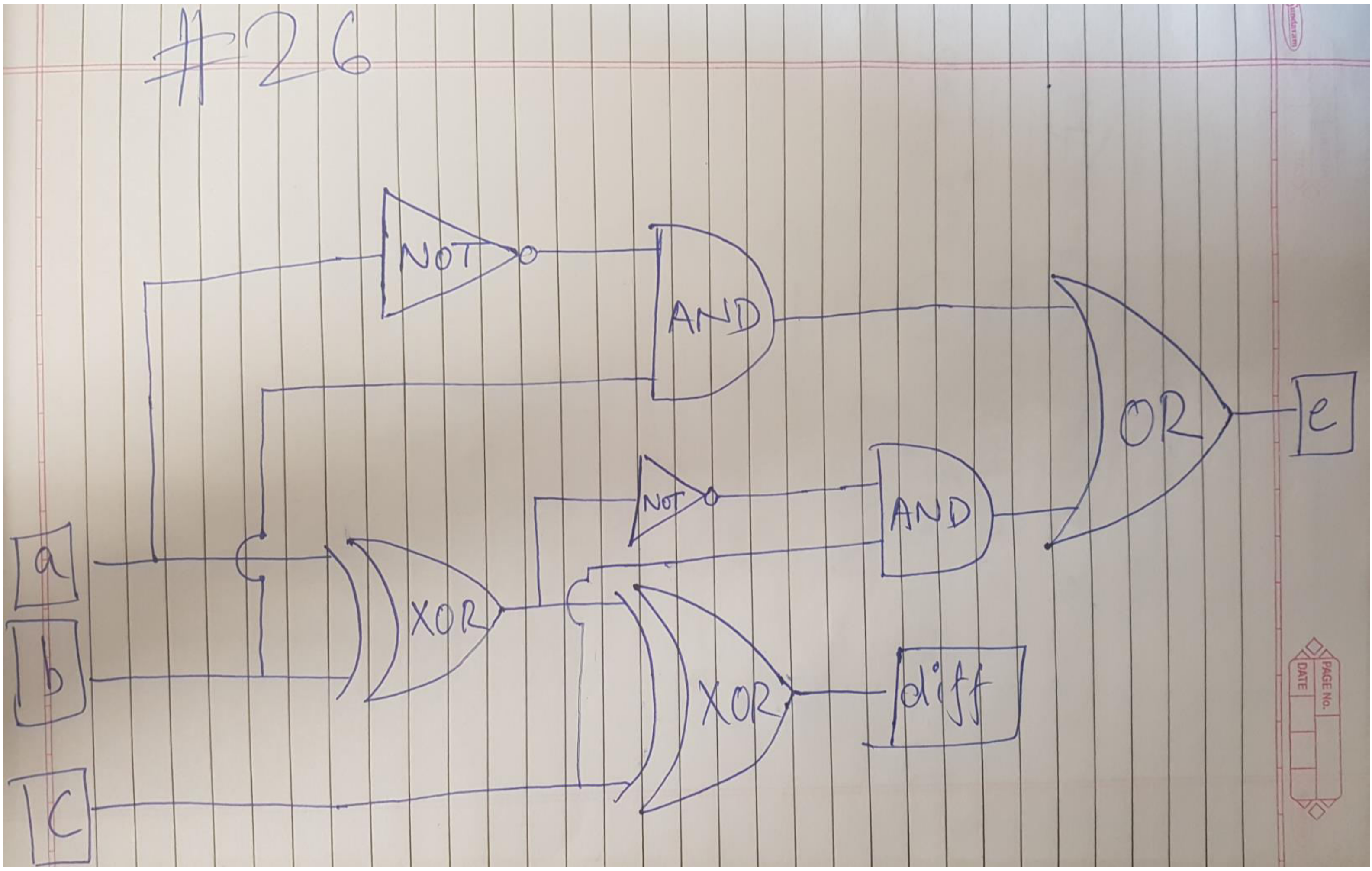
$$\boxed{\text{diff} = c \text{ XOR } (a \text{ XOR } b)}$$

Similarly, solving for e

$$\begin{aligned} e &= a'b'c + a'bc' + a'bc + abc \\ &= a'b(c' + c) + c(a'b' + ab) \\ &= c(a'b + ab')' + a'b \end{aligned}$$

$$\boxed{e = \left(c \text{ AND } \left(\text{NOT}(a \text{ XOR } b)\right)\right) \text{ OR } \left((\text{NOT } a) \text{AND } b\right)}$$

The circuit structure we thus get is as follows:



Chapter 5

2.a

$$N\ bits\ can\ respresent\ 2^N\ memory\ locations.$$

$$N = \log_2 1{,}000{,}000 = 19.9$$

$$\therefore N \text{ should be at least } 20$$

2.b

$$N\ bits\ can\ respresent\ 2^N\ memo\square y\ locations.$$

$$N = \log_2 10{,}000{,}000 = 23.3$$

$$\therefore N \text{ should be at least } 24$$

2.c

$$N\ bits\ can\ respresent\ 2^N\ memory\ locations.$$

$$N = \log_2 100{,}000{,}000 = 26.6$$

$$\therefore N \text{ should be at least } 27$$

2.d

$$N\ bits\ can\ respresent\ 2^N\ memory\ locations.$$

$$N = \log_2 1{,}000{,}000{,}000 = 29.9$$

$$\therefore N \text{ should be at least } 30$$

19.a

$$\text{Max Number of operations} = 2^6 = 64$$

19.b

$$\text{Max Memory Spaces} = 2^{18} = 262144$$

Assuming that every memory space is 8 bits = 1 byte,

$$\text{Max Memory Size} = 262144 \text{ bytes}$$

19.c

$$total\ bits\ = 6 + 18 + 18 = 42$$

$$\frac{42}{8} = 5.25$$

$$6\ bytes\ are\ required$$

22 a

Variable/Value	Location
a	300
b	301
+1	400
-1	401
0	402

Address	Contents	Explanation
50	LOAD 300	Put the value of a in register R
51	ADD 401, 301	Add -1 to b, and stores the result (b-1) in b
52	ADD 301	Adds b to register R. It now holds (a+b-1)
53	STORE 300	Store the contents of register R into a
54	HALT	Stops the program from executing further

22 b

Address	Contents	Explanation
50	COMPARE 300, 402	Compare a and 0, and set condition codes
51	JUMPGT 53	Go to location 53 if a > 0
52	JUMP 55	If a <= 0, skip to instruction 55
53	LOAD 400	Put the value of +1 in register R
54	STORE 301	Store the contents of register R into b
55	HALT	Stops the program from executing further