Introduction to Computer Science
COMS W1004
Problem Set 3 Solutions

Prepared by: Alexander Roth & Ethan Adams

Due: 2016 – 10 – 03

Chapter 4

Exercise 15 **a.** How many bits does it take to store a 3-minute song using an audio encoding method that samples at the rate of 40,000 bits/second, has a bit depth of 16, and does not use compression? What if it uses a compression scheme with a compression ratio of 5:1?

Solution.

$$3 \text{ minutes} \times \frac{60 \text{ seconds}}{1 \text{ minute}} \times \frac{40,000 \text{ samples}}{1 \text{ second}} \times \frac{16 \text{ bits}}{1 \text{ sample}} = 115,200,000 \text{ bits}$$

Thus, there are 115,200,000 bits that we need to store in a 3 minute song. With a compression ration of 5 : 1, we have

$$\frac{1.152 \times 10^8}{5} = 2.304 \times 10^7 \text{ bits}$$

That is, there are 2.304×10^7 bits in the compressed format.

b. How many bits does it take to store an uncompressed $1,200 \times 800$ RGB color image? If we found out that the image actually takes only 2.4 Mbits, what is the compression ratio?

Solution. For the uncompressed image, we have the following calculation:

$$(1200 \times 800) \text{ pixels} \times \frac{24 \text{ bits}}{1 \text{ pixel}} = 23,040,000 \text{ bits}$$

Thus, there are 23,040,000 bits that need to be stored in the uncompressed version. If we find that image takes only 2.4Mbits, we have the following calculations.

$$2.4 \text{ Mbit} = 2.4 \times \frac{1000^2 \text{ bits}}{1 \text{ Mbit}} = 2,400,000$$
$$2.4 \text{ Mbit} = 2.4 \times \frac{1024^2 \text{ bits}}{1 \text{ Mbit}} = 2,516,582$$

In either case, we can approximate both results to 2,400,000. Thus, we have

$$\frac{23,040,000 \text{ bits}}{2,400,000 \text{ bits}} \approx 9.6$$

Thus, the compression ratio is 9.6 : 1.

Exercise 24 Build a *majority-rules circuit.* This is a circuit that has three inputs and one output. The value of its output is 1 if an only if two or more of its inputs are 1; otherwise, the output of the circuit is 0. For example, if the three inputs are 0, 1, 1, your circuit should output a 1. If its three inputs are 0, 1, 0, it should output a 0. This circuit is frequently used in **fault-tolerant computing** – environments where a computer must keep working correctly no matter what, for example as on a deep-space vehicle where making repairs is impossible. In these conditions, we might choose to put three computers on board and have all three do every computation; if two or more systems produce the same answer, we accept it. Thus, one of the machines could fail and the system would still work properly.

Solution. The truth table for the majority-rules circuit is given below:

A	B	C	*Output*
0	0	0	0
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	1

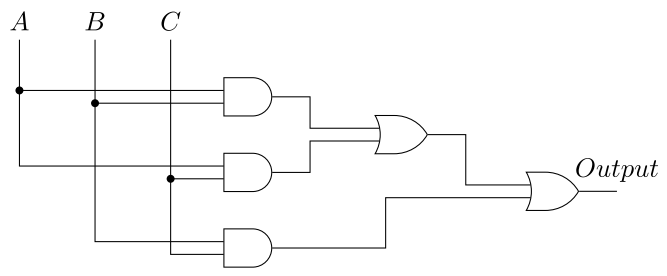
We observe the the equations to generate an output of 1 are

$$Output = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

We can further optimize this circuit by examining the equation above. We see that for each possible combination, we require that only two of the Boolean variables are true. The value of the third negated variable is not necessary to this circuit. For example, $\overline{A}BC + ABC$ can be reduced to just BC. Thus, our optimized Boolean equation is

$$Output = AB + AC + BC$$

We shall use the optimized circuit for our circuit diagram.



Exercise 26 Design a *1-bit subtraction circuit.* This circuit takes three inputs – two binary digits a and b and a borrow digit from the previous column. The circuit has two outputs – the difference $(a - b)$, including the borrow, and a new borrow digit that propagates to the next column. Create the truth table and build the circuit. This circuit can be used to build N-bit subtraction circuits.

Solution. **N.B.** This solution was prepared with the help of Sean Liu.

Review of Subtraciton:

Subtraction works from right to left. At any column, we may need to borrow from the next, higher column (the place on the left) to complete a subtraction. If we choose to borrow from the next column, we also need to remember to subtract that away when we move on to the next column. Essentially, the current input borrow (what this current column has loaned) is always set to the previous output borrow (what that previous column borrowed).

We should be familiar with decimal subtraction. Let's walk through a simple example of $10 - 5$. Starting from the ones' column, we see that we need help to carry out $0 - 5$. Therefore, we **borrow** from the tens' column, and because this is base ten, that means we borrow a value of 10. Now we can successfully carry out $0 - 5 + 10$ to get the answer 5. Moving on to the tens column, we need to perform $1 - 0$. However, we remember that we previously lent a digit to the ones column, so we have to subtract 1 away: $1 - 0 - 1 = 0$. Now, we're done with the problem and we have the complete answer 05.

The same principles apply to binary subtraction. Whenever we borrow from the next place, we borrow a value of 2 because we are in base two.

Here is the complete truth table:

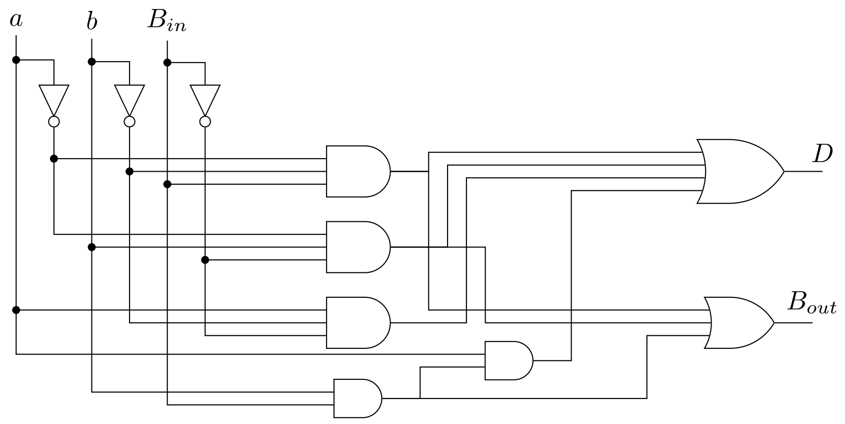
a	b	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From this truth table, we can derive the boolean expressions for our desired outputs:

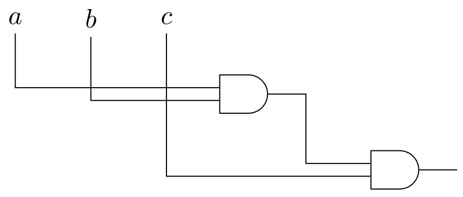
$$D = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + a\overline{b}\overline{B_{in}} + abB_{in}$$

$$B_{out} = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + \overline{a}bB_{in} + abB_{in} = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + bB_{in}$$

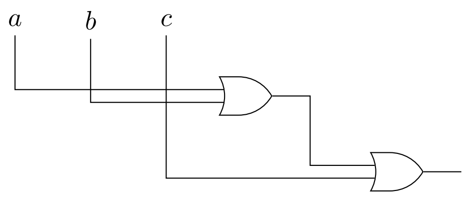
One possible circuit diagram for this circuit is



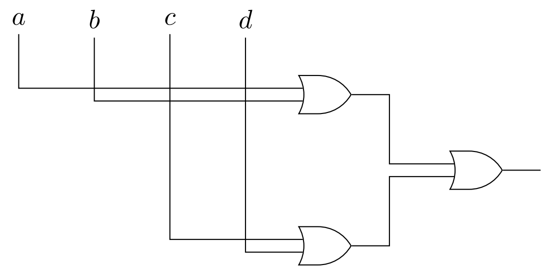
An example of a three input AND gate is



An example of a three input OR gate is



An example of a four input OR gate is



Chapter 5

Exercise 2 At a minimum, how many bits are needed in the MAR with each of the following memory sizes?

a. 1 million bytes

Solution. From *Invitation* pg. 231: "The memory address register (MAR) holds the addresses of the cell to be accessed. Because the MAR must be capable of holding any address, it must be at least N bits wide, where 2^N is the address space of the computer." Thus, we take log base 2 and round up to get N.

$$log_2 1,000,000 = 19.931...$$

$$roundup(19.931) = 20\,\text{bits} = N$$

b. 10 million bytes

Solution.

$$log_2 10,000,000 = 23.253...$$

$$roundup(23.253) = 24\,\text{bits} = N$$

c. 100 million bytes

Solution.

$$log_2 100,000,000 = 26.575...$$

$$roundup(26.575) = 27\,\text{bits} = N$$

d. 1 billion bytes

Solution.

$$log_2 1,000,000,000 = 29.897...$$

$$roundup(29.897) = 30\,\text{bits} = N$$

Exercise 19 Consider the following structure of the instruction register.

Op code	***Address-1***	***Address-2***
6 bits	18 bits	18 bits

a. What is the maximum number of distinct operation codes that can be recognized and executed by the processor on this machine?

Solution. The number of distinct operation codes for this processor is $2^6 = 64$ operation codes. We calculated this number by taking 2 to the size of the opcode field.

b. What is the maximum memory size on this machine?

Solution. The maximum memory of this machine is $2^{18} = 262,144$ bits

c. How many bytes are required for each operation?

Solution. The total number of bites for an operation is calculated as follows:

$$6 \text{ bits} + 18 \text{ bits} + 18 \text{ bits} = 42 \text{ bits}$$

We then determine the number of bytes need to hold these bits.

$$42 \text{ bits} \times \frac{1 \text{ byte}}{8 \text{ bits}} = 5.25 \text{ bytes}$$

We then take the ceiling of 5.25 because we cannot have a quarter of a byte. Thus, 6 bytes are required for each operation.

Exercise 22 Assume that the variables a and b are stored in memory locations 300 and 301, respectively.Also assume that the three integer values +1, −1, and 0 are stored in memory locations 400, 401, and 402, respectively. Finally assume that the code sequence you are writing begins in memory location 50. Using any machine language instructions shown in Section 5.2.4, translate the following algorithmic operations into their machine language equivalents.

a. Set a to the value of $a + b - 1$.

Memory Location	Op-Code	Address Field	Comment
50	ADD	301, 300	$a = a + b$
51	SUBTRACT	400, 300	$a = a - 1$

b. if $a > 0$
Set b to the value +1.

Memory Location	Op-Code	Address Field	Comment
50	COMPARE	300, 402	Compare a and 0; set condition
51	JUMPLE	53	Jump to end if $a <= 0$
52	MOVE	400, 301	$b = 1$
53	HALT		Stop program.