CS 1203: Data Structures **Semester:** Monsoon 2023

Assignment #1

Instructor: Subhamoy Mitra
Due: September 8, 2023

Student Name: Varsha Baisane
varsha.baisane_ug25#@ashoka.edu.in

Question 1

Broad topics discussed:
(1) The boolean functions: The number of boolean functions that can be constructed given there are n inputs.
(2) Bits and Bytes: 8 bits constitute a byte. Data types have data units in bytes such as char which is 1 byte.
(3) The Big O notation: Big O notation can express the best, worst, and average-case time complexities of a algorithm. It does so by comparing two algorithms based on changes in their performance as the input size is increased or decreased.
(4) Space and time complexities: Necessary to consider in-order to keep our program efficient and also not exceed the memory space of the given device.

Question 2

No. of Boolean functions:

(a) For one variable p, 4 functions can be constructed, given the formula for n inputs as 2^n.
A function maps each input value of a variable to one and only one output value.
1. The False(p) function maps each value of p to 0 (False).
2. The identity(p) function maps each value of p to the identical value.
3. The negation(p) function maps False to True and True to False.
4. The True(p) function maps each value of p to 1 (True).

As a result, for a variable p, the number of boolean functions that can be constructed is 2^{2^n} where n represents the number of input variables.

(b) For two variables p and q, 16 Boolean functions can be constructed.
One way to show this is by listing the 4 truth assignments to the variables:

P	Q	$f(P,Q)$
0	0	a
0	1	b
1	0	c
1	1	d

Now, since the assignments a,b,c,d can have either 0 or 1 as an output, we have 2 choices per 4 assignments. That being said, the number of n inputs 1 output function is 2^{2^2}=16.

Therefore, there are 2^{2^n} different Boolean functions on n given Boolean variables.

Source: https://cs.fit.edu

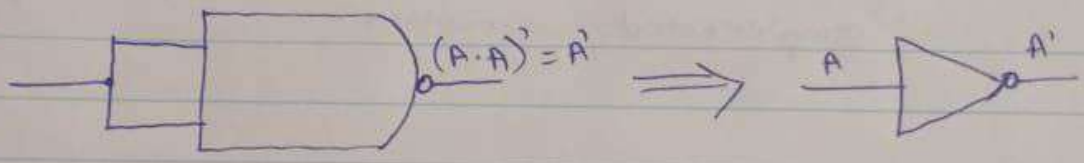
SOURCE:- www.uop.edu.pk

Q. NAND is an universal gate :-

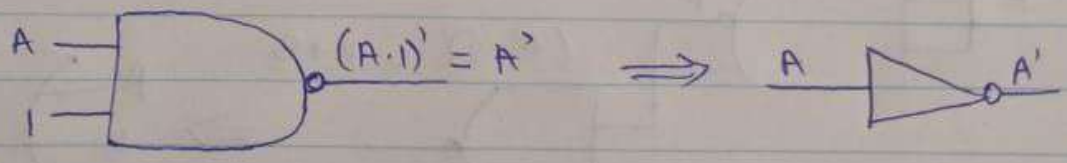
To say that a gate is universal, it has to be able to implement any Boolean function without ~~ae~~ using any other gate type.
So, to show that a NAND gate is universal, we build boolean function using NAND gate alone.

(a) NAND gate can be used as an inverter (NOT gate)

All NAND input pins connected to the input signal A gives an output A'.



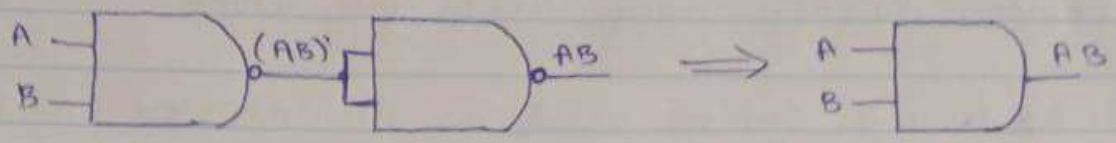
~~Usin~~ for two inputs,
One NAND input pin is connected to the input signal A while the other is connected to logic 1, The output is A'



As a result, NAND can implement a NOT gate

* Implementing AND gate

An AND gate can be implemented from a NAND gate by complementing a NAND gate, with NAND inverter.
output



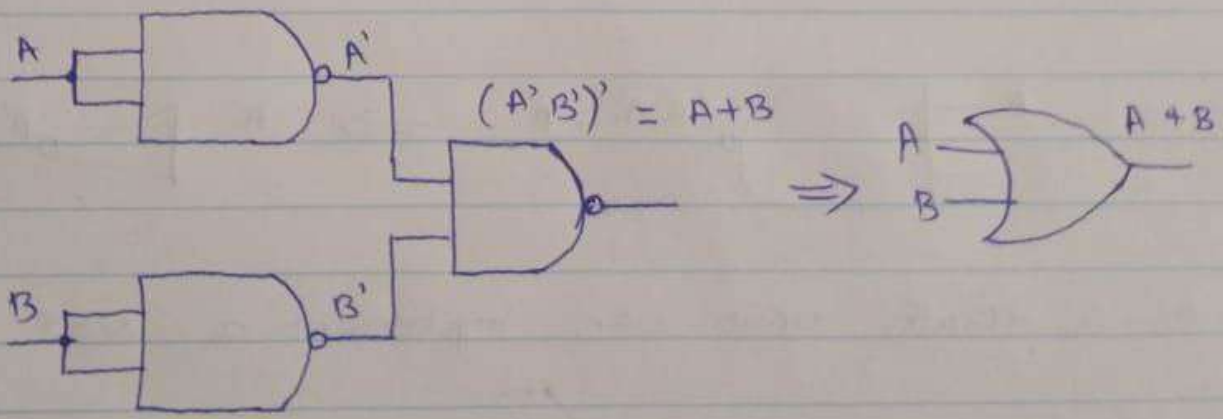
A Nand gate's output complemented

with a nand inverter

gives us an AND gate

* Implementing OR gate

An OR gate can be implemented by complementing all inputs of a NAND gate by NAND gate inverters.



Therefore, the NAND gate is a universal gate since it can implement the AND, OR and NOT boolean functions.

Listing 1: Printing Value and Address of a variable in C

```c
#include <stdio.h>

int main(void)
{

    int a;
    a = 10;
    printf("value of a: %d\n", a);
    printf("Address of a: %p\n", &a);

}
```

Listing 1: Finding Max and Min of an array in C

```c
#include <stdio.h>
#define SIZE 50

int array[SIZE];
int i, max, min, size, pos1, pos2;

printf("Enter size of the array: ");
scanf("%d", &size);

printf("Enter elements in the array: ");
for(i=0; i<size; i++) {
    scanf("%d", &array[i]);
}

max = array[0];
min = array[0];

for(i=1; i<size; i++) {

    if(array[i] > max) {
        max = array[i];
        pos1 = i + 1;
    }

    if(array[i] < min) {
        min = array[i];
        pos2 = i + 1;
    }
}

printf("Maximum element = %d is at position %d\n", max,
    pos1);
printf("Minimum element = %d is at position %d\n", min,
    pos2);
```