C501: Computer Architecture
Assessed Coursework

Due date: **15th November 2017**
(Hard-copy to SAO by 16:00)

1. **Boolean Algebra and Digital Circuits**

   (a) Simplify the following Boolean expressions to its simplest form. The symbols $\bullet$, $+$, and $'$ represent "AND", "OR" and "NOT" operations respectively. Please show the sequence of steps and state the reduction rules used.

   (i) $E = A \bullet B + B \bullet (A' + A \bullet B)$ (5 marks)

   Answer

   $E = A \bullet B + B \bullet (A' + A \bullet B)$
   $E = B \bullet (A + A' + A \bullet B)$ (Distributed Rule)
   $E = B \bullet (1 + A \bullet B)$ (Negation Rule)
   $E = B \bullet 1$ (Simplification Rule)
   $E = B$

   (ii) $E = (A + B)' \bullet (C + D + F)' + (A' \bullet B')$ (10 marks)

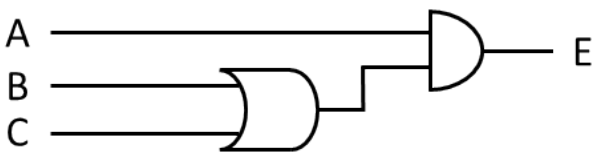
   Answer

   $E = (A + B)' \bullet (C + D + F)' + (A' \bullet B')$
   $E = A' \bullet B' \bullet C' \bullet D' \bullet F' + A' \bullet B'$ (De Morgan's Rule)
   $E = (A' \bullet B') \bullet (C' \bullet D' \bullet F' + 1)$ (Distributed Rule)
   $E = (A' \bullet B') \bullet 1$ (Simplification Rule)
   $E = A' \bullet B'$

   (b) Use Boolean Algebra to simplify the following expression and then draw the logic circuit for the simplified expression. $E = A \bullet (B + A \bullet B) + A \bullet C$ (10 marks)

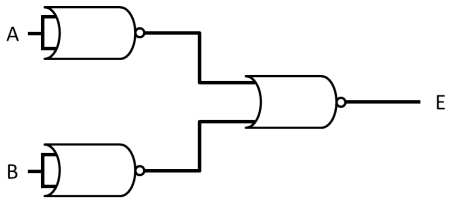
   Answer

   $E = A \bullet (B + A \bullet B) + A \bullet C$
   $E = A \bullet B + A \bullet C$ (Simplification Rule)
   $E = A \bullet (B + C)$ (Distributed Rule)



   (c) NOR gates as well as NAND gates are considered as universal gates. Build a circuit for the following Boolean expression $E = A \bullet B$ using only NOR gates. Prove that your circuit works, either by using truth tables or Boolean Algebra reduction rules. (15 marks)

   Answer



Looking at the circuit, we get:

$E = ((A + A)' + (B + B)')'$

$E = ((A + A)') \bullet ((B + B)')'$ (De Morgan's Rule)

$E = (A + A) \bullet (B + B)$ (Negation Rule)

$E = A \bullet B$ (Idempotent Rule)

This shows that the circuit above gives us $E = A \bullet B$ using only NOR gates.

2. **Binary Arithmetic**

Show your working clearly.

(a) Assume that you use 5-bits to represent a number. Using 2's complement representation for numbers, show the calculations of:

(i) 9 - 11. (5 marks)

Answer

While using 2's complement, 9 - 11 can be calculated as 9 + (-11). Converting to binary representations for 9 and (-11), we get:

Binary value of 9 = 01001
Binary value of 11 = 01011
1s complement of 11 = 10100
2s complement of 11 = 10100 + 1 = 10101
Putting this together:

01001 +
10101
———
11110
———

We get a two's complement number. Converting back into positive, we get 00010 = 2.

(ii) -12 - 10. (10 marks)

While using 2's complement, -12 - 10 can be calculated as -12 + (-10). Converting to binary representations for -12 and (-10), we get:

Binary value of 12 = 01100
1s complement of 12 = 10011
2s complement of 12 = 10011 + 1 = 10100
Binary value of 10 = 01010
1s complement of 10 = 10101
2s complement of 10 = 10101 + 1 = 10110

Putting this together:

10100 +
10110
———
101010
———

Discarding the overflow, we have 01010, but this is a positive number. In this case two numbers whose signs were different were subtracted, but the result has the opposite sign and the sign of the subtrahend, which is an Overflow and hence a wrong value.

(b) Evaluate $\frac{189}{27}$ using Binary Arithmetic. (15 marks)

Answer

Binary value of 189 = 10111101
Binary value of 27 = 11011
Using long-division method (any other method is also acceptable)

```
        0111
      ---------------
11011 |10111101
        11011
        --------
         101000
          11011
        --------
           11011
           11011
        --------
               0 = Reminder
        --------
```

Answer = 111

3. **Floating Point Numbers**

Show your working clearly.

(a) Convert the decimal number, -31.1 into IEEE Single Precision format and its corresponding hexadecimal value. (15 marks)

Answer

We start off by ignoring the sign.
Breaking up the number into before and after decimal, we get
Binary of 31 = 11111
Binary of 0.1 is = 0.0001100110011001100110011001 (recurring)
Binary of 31.1 is = 11111.0001100110011001100110011001 (recurring)
Normalising, we get: 1.11110001100110011001101 (rounding up) * 2^4
Significand is thus: 11110001100110011001101

Exponent field is (4 + 127 = 131): 10000011

We can now add the sign field, which would be 1 here.

Stored as: 1100 0001 1111 1000 1100 1100 1100 1101

Hexadecimal: C1F8CCCD

(b) Using the IEEE Single Precision format, convert the following hexadecimal number, 40F109D5 into binary and decimal. (15 marks)

Answer

40F109D5 is stored in binary as: 0100 0000 1111 0001 0000 1001 1101 0101
Sign bit is: 0 (number is positive)
Exponent field is: 10000001 (129 - 127 = 2)
Significand is thus: 11100010000100111010101
Normalised number is thus: $1.11100010000100111010101 * 2^2$
Number is: 111.10001000100111010101

Converting to Decimal, we get:
$(2^2 + 2^1 + 2^0).(2^{-1} + 2^{-5} + 2^{-10} + 2^{-13}\ 2^{-14} + 2^{-15} + 2^{-17} + 2^{-19} + 2^{-21})$
= (approximately) 7.532450199127197
or = 7.53245 (Rounded values are accepted)