ITM207 Tip Sheet: Midterm Review (includes main calculations)

For the Midterm, you must review main concepts from Professor's slides and textbook

By: Priyanshi Patel

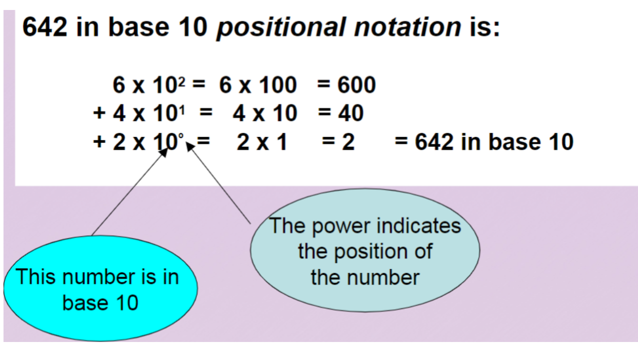
Binary Values and Number System

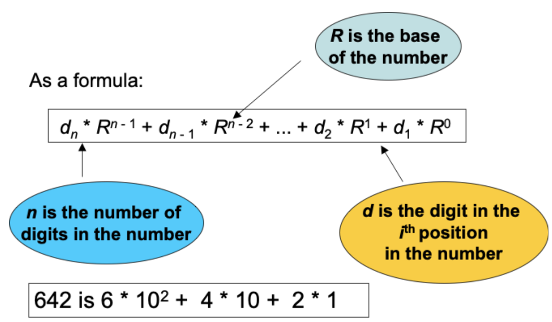
Numbers

- **Natural Numbers:** Zero and any number obtained by repeatedly adding one to it
  - E.g: 100, 0, 45645, 32
- **Negative Numbers:** A value less than 0, with a – sign
  - E.g: –24, –1, –45645, –32
- **Integers:** A natural number, a negative number
  - E.g: 249, 0, –45645, –32
- **Rational Numbers:** An integer or the quotient of two integers
  - E.g: –249, –1, 0, 3/7, –2/5

Positional Notation

- **Base of a number determines the number of different digit symbols (numerals) and the values of digit positions.**





Bases

- **Decimal** is base 10 and has 10 digit symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **Binary** is base 2 and has 2 digit symbols: 0, 1
- **Octal** is base 8 and has 8 digit symbols: 0,1,2,3,4,5,6,7
- **Hexadecimal** is base 16 and has 16 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F

Hexadecimal to Decimal Conversion Table (cuemath THE MATH EXPERT)

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

For a number to exist in a given base, it can only contain the digits in that base, which range from 0 up to (but not including) the base.

Arithmetic in Binary

- **Binary Addition**

  Remember that there are only 2 digit symbols in binary, 0 and 1

  1 + 1 is 0 with a carry

  ```
   1011111   <-- Carry Values
    1010111
   +1001011
   10100010
  ```

  -

- **Binary Subtraction**
  - **Simple Subtraction**

    ```
     012
       02
     1010111
    - 111011
     0011100
    ```

  - **Using 2's complement**

    ```
      10001100
    - 00010111     → take bottom number and convert using
                     2's complement
                     00010111  (+)
                     11101000  invert
                            1  +1
         Simply  {   11101001  (-)
         add

      10001100
    + 11101001
     101110101
     ↓        9 digits, there is an overflow,
              must only be 8
    we cross out the extra
      X01110101
         ↓ 01110101 ]- that's the answer

    ∴ 10001100 - 00010111 = 01110101
    ```

Converting to different bases

- ### Octal to Decimal

What is the decimal equivalent of the octal number 642?

$$
\begin{aligned}
& 6 \times 8^2 = 6 \times 64 = 384 \\
+\ & 4 \times 8^1 = 4 \times 8 = 32 \\
+\ & 2 \times 8^0 = 2 \times 1 = 2 \\
& = 418 \text{ in base } 10
\end{aligned}
$$

- ### Hexadecimal to Decimal

What is the decimal equivalent of the hexadecimal nb DEF?

$$
\begin{aligned}
& D \times 16^2 = 13 \times 256 = 3328 \\
+\ & E \times 16^1 = 14 \times 16 = 224 \\
+\ & F \times 16^0 = 15 \times 1 = 15 \\
& = 3567 \text{ in base } 10
\end{aligned}
$$

- ### Binary to Decimal

What is the decimal equivalent of the binary number 1101110?

$$
\begin{aligned}
& 1 \times 2^6 = 1 \times 64 = 64 \\
+\ & 1 \times 2^5 = 1 \times 32 = 32 \\
+\ & 0 \times 2^4 = 0 \times 16 = 0 \\
+\ & 1 \times 2^3 = 1 \times 8 = 8 \\
+\ & 1 \times 2^2 = 1 \times 4 = 4 \\
+\ & 1 \times 2^1 = 1 \times 2 = 2 \\
+\ & 0 \times 2^0 = 0 \times 1 = 0 \\
& = 110 \text{ in base } 10
\end{aligned}
$$

- ### Binary to Octal

- Mark groups of *three* (from right)
- Convert each group

10101011	**10**	**101**	**011**
	2	**5**	**3**

10101011 is 253 in base 8

- **Use Binary to convert each group**
- **E.g. the first group is 10**
  - **$1 * 2^1 = 2$**
  - **$0 * 2^0 = 0$**
  - **Add = 2**

- **Binary to Hexadecimal**

• Mark groups of *four* (from right)
• Convert each group

10101011   **1010 1011**
                    A   B

10101011 is AB in base 16

  - **Use Binary to convert each group**
  - **E.g. the first group is 1010**
    - **1 * 2^3 = 8**
    - **0 * 2^2 = 0**
    - **1 * 2^1 = 2**
    - **0 * 2^0 = 0**
    - **Add = 10 ⇒ A**

- **Decimal to Other Bases**
  - **Algorithm for converting number in base 10 to other bases:**
  - **While the quotient is not zero:**
    - **Divide the decimal number by the new base**
    - **Make the remainder the next digit to the left in the answer**
    - **Replace the original decimal number with the quotient**

What is 1988 (base 10) in base 8?

```
    248       31       3      0
8 1988   8 248    8 31    8 3
  16        24      24      0
   38       08       7      3
   32        8
   68        0
   64
    4
```

Answer is : 3 7 0 4

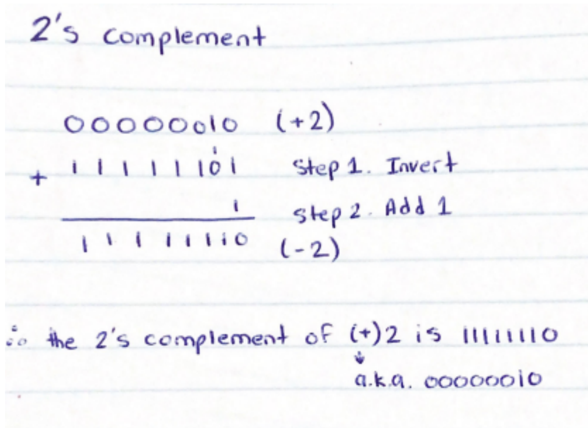
What is 3567 (base 10) in base 16?

```
     222         13        0
16 |3567   16 |222   16 |13
    32         16          0
     36         62        13
     32         48
      47        14
      32
      15
```

D E F

Data Representation

- **Representing Negative Values**
  - **Ten's complement representation we can use this formula to compute the representation of a negative number**
    - Negative(I) = $10^k - I$, where k is the number of digits
  - **For example, –3 is negative(3), so using two digits, its representation is**
    - **Negative(3) = 100 – 3 = 97**

- Two's Complement
  - **Converts a positive integer into a negative integer**
  - **Steps:**
    - **1. Invert (change all 1's to 0's and all 0's to 1's)**
    - **2. Add 1**
  - 

- **Representing Real Numbers**
  - **Floating Point**
    - **A real value in base 10 can be defined by the following formula where the mantissa is an integer:**
    - $\text{sign} * \text{mantissa} * 10^{\text{exp}}$
    - **This representation is called floating point because the radix point "floats"**
    - **E.g - 43. 254**
    - **= - * 4254 * 10^3**

  - **Scientific Notation**
    - **A form of floating-point representation in which the decimal point is kept to the right of the leftmost digit**
      - **E.g 12001.32708 is 1.200132708E+4 in scientific notation**
        - **(E+4 is how computers display x10^4)**

  - **Converting a Real Number to Binary**
    - **How to convert decimal fractions:**
    - **multiply by 2 and save the whole number part of the answer**
    - **Example 1: Convert the decimal number: 0.625 to binary**
      - **0.625 * 2 = 1.25 ⇒ Here we saved 1**
      - **Now disregard the whole number part of the previous result and multiply by 2 again. Continue this process until you get a zero in the decimal part:**

        - **0.25 * 2 = 0.50 ⇒ Here we saved 0**
        - **0.50 * 2= 1.00 ⇒ Here we saved 1 and the calculation stops here since the decimal part is zero**
    - **Example 2: Convert the decimal number: 5.425 to binary, keeping 4 decimal places**
        - **5 in Binary is: 101**
        - **To get the binary for 0.425 do the following:**
            - **0.425 * 2 = 0.85**
            - **0.85 * 2 = 1.70**
            - **0.70 * 2 = 1.4**
            - **0.4 * 2 = 0.8**
            - **So, 0.425 in Binary is .0110 (only need 4 decimal places)**
        - **So, 5.425 in Binary is: 101.0110**

- **Text Compression**
    - **Key Word Encoding**
        - Replace frequently used patterns of text with a single special character

          Example

WORD	SYMBOL
as	^
the	~
and	+
that	$
must	&
well	%
these	#

            - **Original:** that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the pursuit of Happiness.
            - **Compressed:** $ ~y are endowed by ~ir Creator with certain unalienable Rights, $ among # are Life, Liberty + ~ pursuit of Happiness.
            - **Compression ratio:** compressed # of characters / original # of characters ⇒ 117/136 = 0.86

    - **Run Length Encoding**
        - Replace a repeated sequence
            - with a flag
            - the repeated value
            - the number of repetitions
        - Example: nnnnn ⇒ *n5
            - * is the flag
            - n is the repeated value
            - 5 is the number of times n is repeated
        - **Rule** → only compress repeated values > 3
            - Example:
                - **Original:** aaabbhhhhhcd
                - **Compressed:** aaabb*h5cd
                - Do not compress a,b, c and d as they are not greater than 3

- **Compression Ratio =** compressed # of characters / original # of characters ⇒ 10/12 = 0.833

- **Huffman Encoding**
  - Huffman encoding is an example of prefix coding:
    - no character's bit string is the prefix of any other character's bit string
    - To decode:
      - Look for match left to right, bit by bit
      - Record letter when a match is found
      - Begin where you left off, going left to right
  - **Example**
    - ballboard = 1010001001001010110001111011
    - **To find Compression Ratio**
      - First make groups of 8 to find how many bytes the compressed form uses
        - 10100010
        - 01001010
        - 11000111
        - 1011xxxx
      - So, the compressed form of ballboard uses 4 bytes
      - **Using ASCII**
        - Each character represents 1 byte
        - Original form of ballboard uses 9 bytes
        - **Compression ratio:** 4/9 = 0.44
      - **Using Unicode**
        - Each Character represents 2 bytes
        - Orignal form of billboard uses 18 bytes
        - **Compression Ratio:** 4/18 = 0.22

Huffman Code	Character
00	A
01	E
100	L
110	O
111	R
1010	B
1011	D

Boolean Logic and Computing Fundamentals

- **Only outputs** → 0 = low voltage, 1 = high

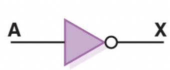
NOT Gate

A NOT gate accepts one input signal (0 or 1) and returns the complementary (opposite) signal as output

Boolean Expression

X = A'

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

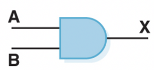
AND Gate

An AND gate accepts two input signals
If both are 1, the output is 1; otherwise,
the output is 0

Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

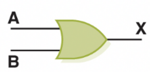
OR Gate

An OR gate accepts two input signals.
If both are 0, the output is 0; otherwise,
the output is 1

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

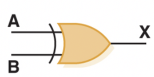
XOR Gate

An XOR gate accepts two input signals. If both are the same, the output is 0; otherwise, the output is 1

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Note the difference between the XOR gate and the OR gate; they differ only in one input situation

- When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR is called the exclusive OR because its output is 1 if (and only if):

- Either one input or the other is 1
- Excluding the case that they both are

NAND Gate

The NAND (“NOT of AND”) gate accepts two input signals
If both are 1, the output is 0; otherwise,
the output is 1

Boolean Expression

X = (A • B)'

Logic Diagram Symbol

Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

The NOR (“NOT of OR”) gate accepts two inputs.
If both are 0, the output is 1; otherwise,
the output is 0

Boolean Expression

X = (A + B)'

Logic Diagram Symbol

Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

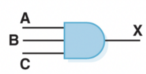
Gates with Multiple Inputs

Some gates can be generalized to accept three or more input values
A three-input AND gate, for example, produces an output of 1 only if all input values are 1

Boolean Expression

X = A • B • C

Logic Diagram Symbol



Truth Table

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Constructing Gates

- Transistor: device that acts either as a wire that conducts electricity or as a resistor that blocks the flow of electricity, depending on the voltage level of an input signal
- It is made of a semiconductor material, which is neither a particularly good conductor of electricity nor a particularly good insulator

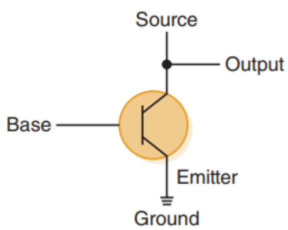


FIGURE 4.8 The connections of a transistor

    - made up of 3 terminals: a source, a base and an emitter

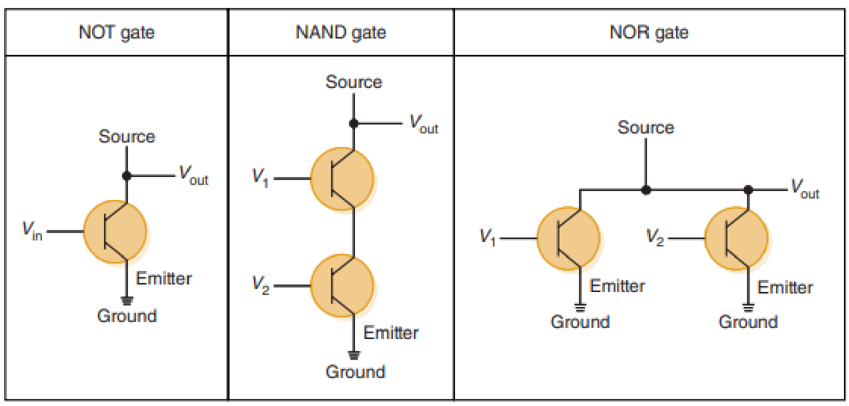


FIGURE 4.9 Constructing gates using transistors

    - **Not Gate** → one transistor
    - **Nand Gate** → two transistors
    - **Nor Gate** → two transistors
- **AND** gates are more complicated to construct than **NAND** Gates ⇒ three transistors
    - two for NAND and one for the NOT

Properties of Boolean Algebra

PROPERTY	AND	OR
Commutative	AB = BA	A + B = B + A
Associative	(AB)C = A(BC)	(A + B) + C = A + (B + C)
Distributive	A(B + C) = (AB) + (AC)	A + (BC) = (A + B) (A + C)
Identity	A1 = A	A + 0 = A
Complement	A(A') = 0	A + (A') = 1
De Morgan's law	(AB)' = A' OR B'	(A + B)' = A'B'