**Chapter 1**

Table of Contents

[Addition in Base 3](#_Toc44592291)

[Subtraction in Base 4](#_Toc44592292)

[Multiplication in Base 5](#_Toc44592293)

[Division in Base 6](#_Toc44592294)

['s Complement 7](#_Toc44592295)

[()'s Complement 8](#_Toc44592296)

[Binary Coded Decimal (BCD) 9](#_Toc44592297)

[8-4-2-1 Code 9](#_Toc44592298)

[XS 3 Code 9](#_Toc44592299)

[2421 Code 10](#_Toc44592300)

[Parity 11](#_Toc44592301)

[5043210 Code 12](#_Toc44592302)

[Gray Code 13](#_Toc44592303)

[Alphanumeric Code 14](#_Toc44592304)

[Binary Storage and Registers 15](#_Toc44592305)

[Registers in Different Components 16](#_Toc44592306)

[Binary Logic 17](#_Toc44592307)

[Switching Circuits and Binary Signals 18](#_Toc44592308)

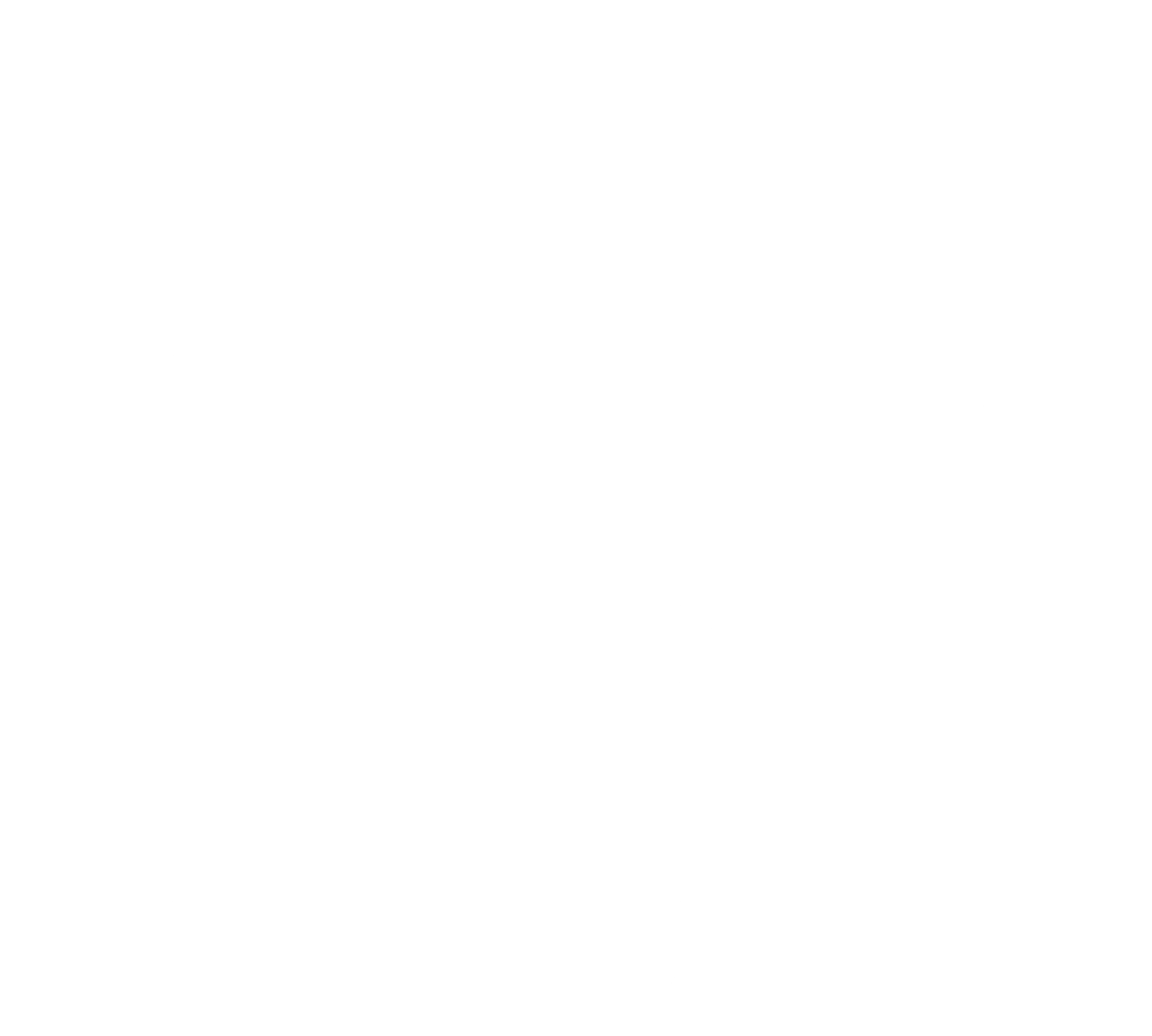
[Logic Gates 19](#_Toc44592309)

[Integrated Circuits 20](#_Toc44592310)

## Addition in Base

Addition is done almost normally, except that the values range from to . If an individual result becomes greater than , the result is written as and is carried.

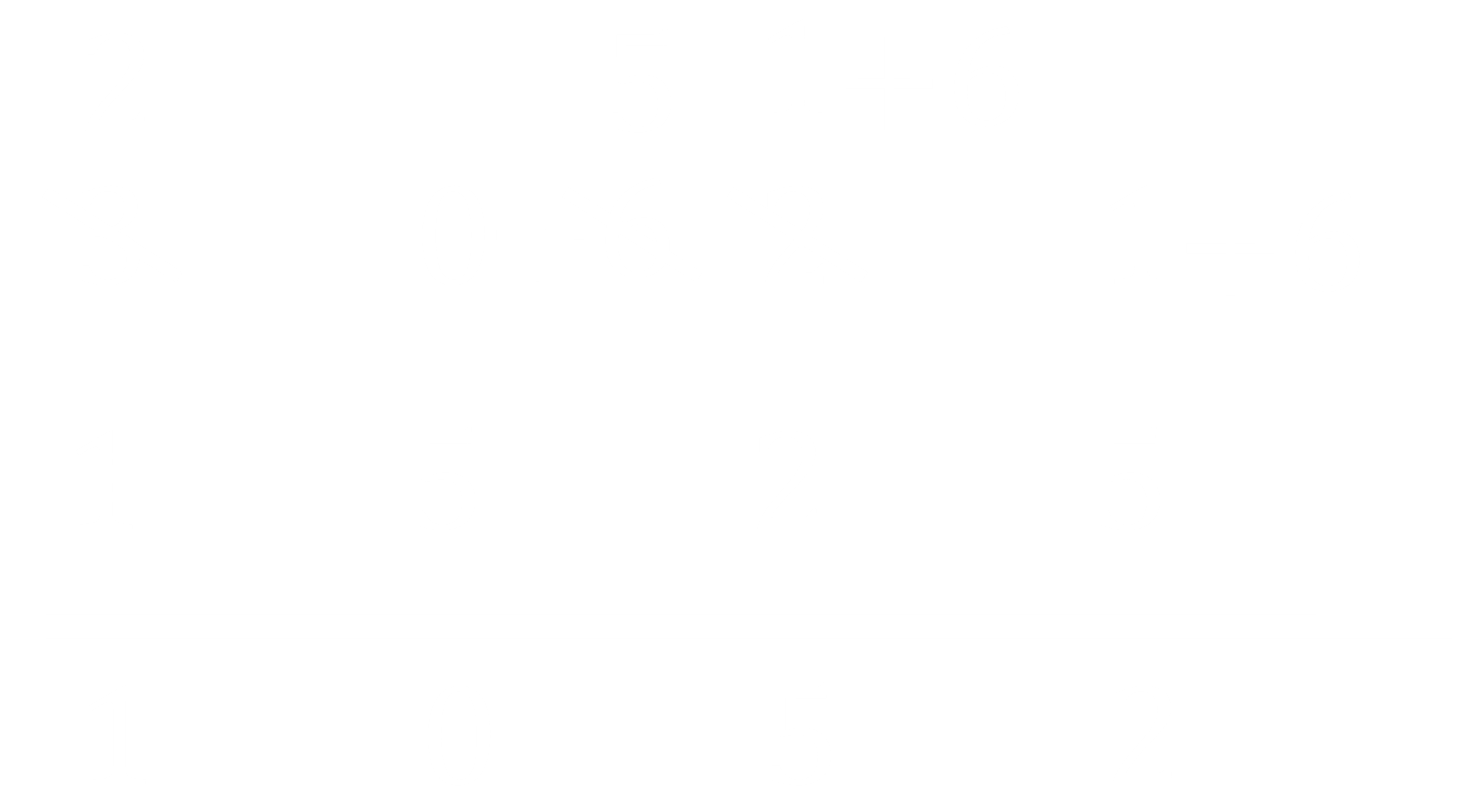
The following is an example of an addition done in base .



## Subtraction in Base

Subtraction is similar to addition. The only difference is that if a larger digit is subtracted from a smaller digit, must be borrowed. What this actually means is, the next digit will decrease by , and the current digit will increase by .

The following is an example of a subtraction done in base .



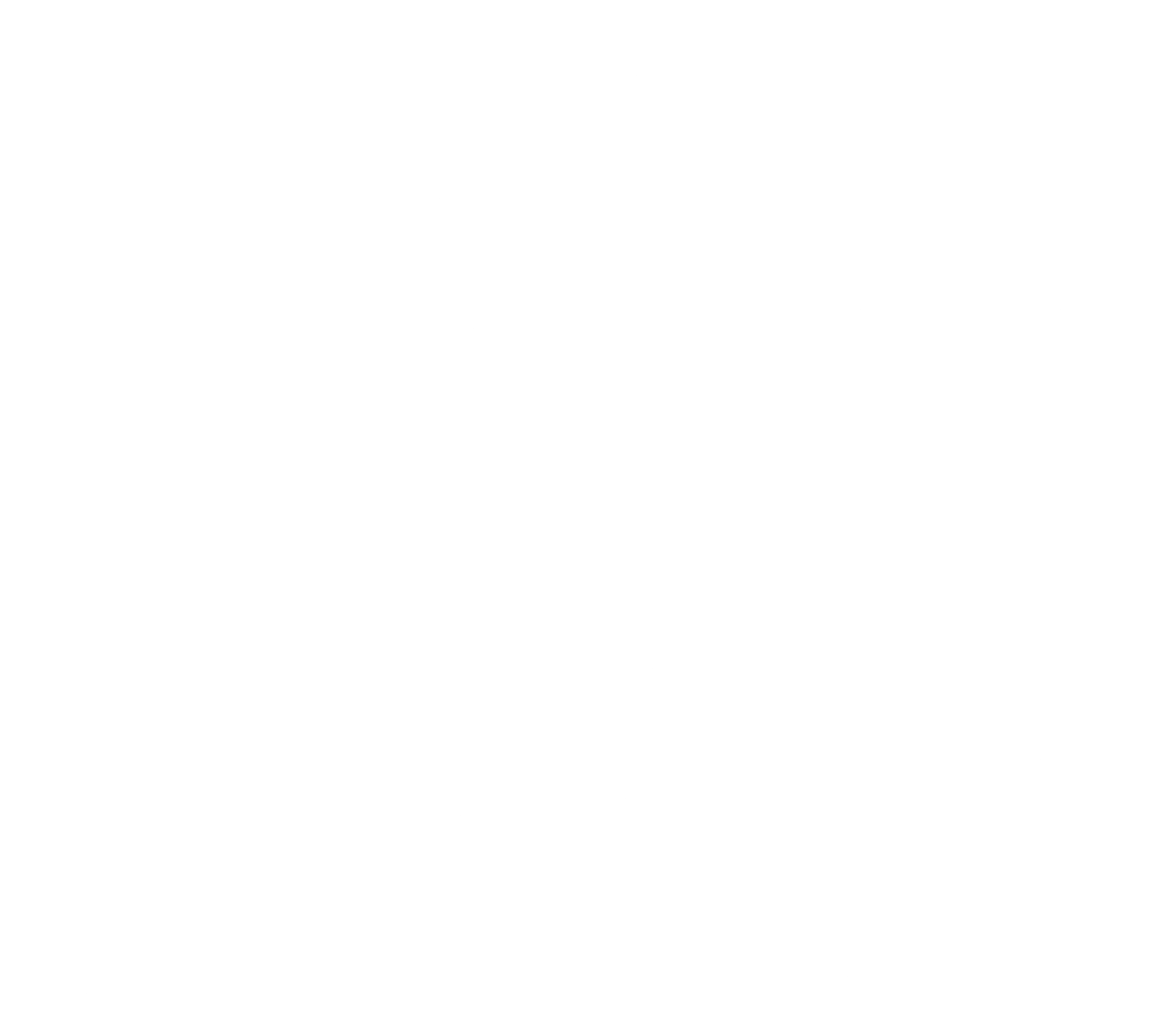
1. , so is borrowed from the next digit making it . The result is .

2. , so is borrowed. The next digit is is is borrowed from the digit after that. becomes , from which is then borrowed, making it .

## Multiplication in Base

Each digit is multiplied normally. If the result is greater than , then the largest possible multiple is subtracted from the result, and is taken as carry for the next digit. The carry is added after multiplying the next digit to find its result. The process is continued and addition in base is done later, as with usual multiplication.

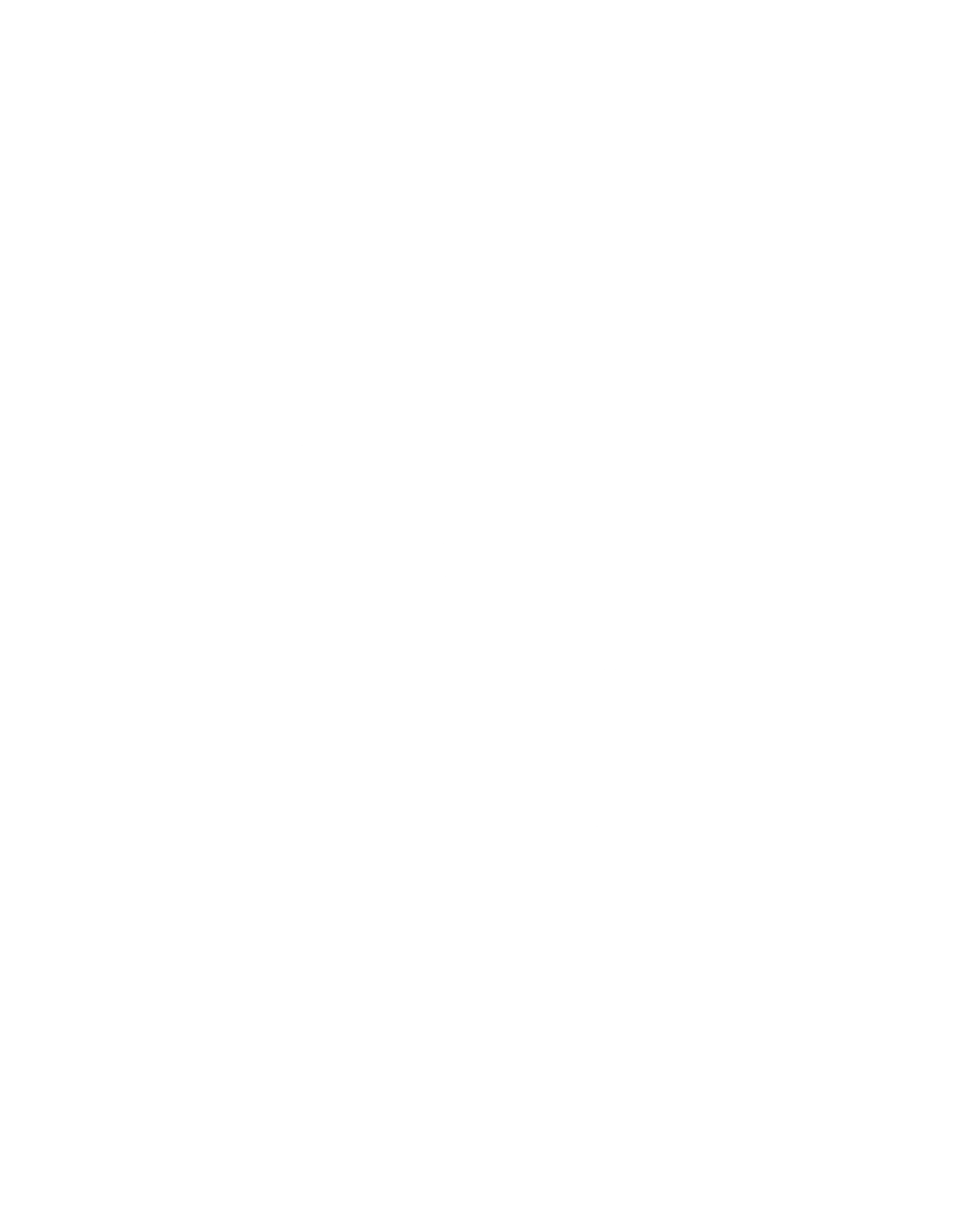
The following is an example of a multiplication done in base 6.



## Division in Base

The divisor is converted to decimal and the times table of the decimal result up to some arbitrary value is drawn up. Each value is converted to give the its corresponding value in base . The times table of the divisor in base has been found. This is used to perform long division, using subtraction in base as required.

Times Tables:



## 's Complement

* - base
* – number of digits
* – the number
* Discard end carry if it exists.
* If no end carry, 's complement answer and add negative sign.
* Shortcut - Leave least significant 0's. Subtract 1st non-zero number from right from base. Subtract rest from (base - 1).
* Shortcut - Addition of with the (r-1)’s complement

## ()'s Complement

* – digits after decimal
* No borrow
* If end carry, add 1 to least significant bit. Discard end carry.
* If no end carry, ()'s complement answer and add negative sign.
* Shortcut - subtract each digit from ()

## Binary Coded Decimal (BCD)

### 8-4-2-1 Code

* Weighted Code
* 9 onwards - called don't cares/invalids
* Less efficient than binary
* Not self-complementing

### XS 3 Code

* Unweighted
* 8421 Code +
* For numbers, add to each digit.
* Only unweighted code that is self-complementing.
* Adding two digits - add to each digit. Subtract from result.
* Adding two numbers (more than 1 digit in each number) - add to each digit of each number. add first digit of one number to first digit of other number. second digits. So on. So, groups of addition. If there is a carry on a group, add 0011 to result of that group. If there is no carry, subtract 0011.

## 2421 Code

* Each digit can be written in two forms, so two sets of digits.
* Primarily written using least significant numbers as far as possible. This set will not be self-complementing.
* Digits can be written so that they are self-complementing.

## Parity

* Can detect single bit error.
* A bit stream is sent along with an extra bit (called parity bit).
* The parity bit tells us about the total number of 1's in the signal.
* For even parity, if the original signal had odd number of 1's, then parity bit is 1. Otherwise 0.
* For odd parity, opposite.
* External noise can change bits. If odd number of 1's coming and even number of 1's expected, then signals error. However, can only detect error if error causes odd number of bits to be changed.
* Random errors - can't be fixed. Retransmit.
* Often errors - system error. Distorts meaning of message.

## 5043210 Code

* Decimal digit represented as where and
* Used in error detection

## Gray Code

* Reflected Binary Code (RBC)
* Two successive values differ by only one bit. Easier to switch values.
* Physical switches are not ideal. They have a brief period in between changes where they will read some inauthentic value.
* Unweighted
* Other names - Unit distance code/Minimum error code/Cyclic code/Single distance code (hamming distance = 1)
* Binary to Gray Code - Leave MSB as is. XOR each digit from MSB of original number with its immediate right digit.
* Gray Code to Binary - Leave MSB as is. XOR each digit from MSB of new number being formed with its immediate right digit.

## Alphanumeric Code

* Consists of 10 digits, 26 letters and some special symbols.
* More than 36 characters.
* Needs at least bits
* Internal code (6 bits) - ASCII from 32 to 95 only. ASCII minus 32 so 0 - 63
* ASCII code (7 bits)
* EBCDIC code (8 bits) - Extended Binary Coded Decimal Interchange Code. First 4 bits indicate zone. Second 4 bits indicate specific character of zone. Used on IBM mainframes.

## Binary Storage and Registers

* Binary cell (a binary storage element) can store 1 bit of information. Has two stable states.
* Punch card with holes, ferrite cores, flip-flops.
* A group of binary cells is called a register. An -cell register can store bits of discrete information.
* State of register – an -tuple number of 1s and 0s (-tuple just means ordered set of numbers)
* Content of a register – a function of the interpretation of stored information in it.
* Same bit configuration may be interpreted differently for different types of elements of information (types should be synched with the computer – e.g. string, integer, floating, image, audio, etc.)

## Registers in Different Components

* Processor Unit: store operands upon which operations are performed
* Control Unit: keep track of various computer sequences
* I/O devices: store information transferred to or from the device
* To process binary information, a computer must have:
* Devices which hold the data to be processed (Register)
* Circuit elements which manipulate data (Logic circuit, ALU)

## Binary Logic

* Deals only with logic () and not arithmetic ()
* AND, OR, NOT
* Truth tables

## Switching Circuits and Binary Signals

* Logic states represented as switches.
* Electronic digital circuit uses transistor as switches - known as switching circuits. Conducting - on, Non-conducting - off.
* Switch parallel to component is NOT.
* Voltage operand circuit - two separate voltage levels
* Current operand circuit - cut off or transition states

## Logic Gates

* Establish logical manipulation path carrying one bit of information
* Also called digital circuit, logic circuit or switching circuit
* Boolean algebra used for mathematical representation
* OR, AND, NOT, NAND, NOR, XOR

## Integrated Circuits

* Digital circuit constructed with integrated circuit
* small semi-conductor crystal named chip
* consists of transistors, diodes, resistors, capacitors and so on interconnected inside
* Chip mounted on metal or plastic package. Connections made on external pin.
* Flat and Dual in Line
* Small, cost effective, low power consumption, high reliability.
* Linear IC - Continuous data. Digital IC - Discrete Data
* Based on number of gates inside - SSI, MSI, LSI, VLSI IC (Small Scale Integration, Medium Scale, Large Scale, Very Large Scale)