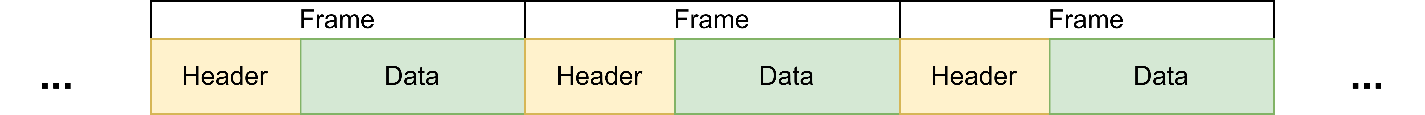
# Задание 1. Кадровая синхронизация

Дана битовая последовательность, имеющая структуру:



Frame – кадр данных, состоящий из Header(заголовок) и Data(случайных данных).

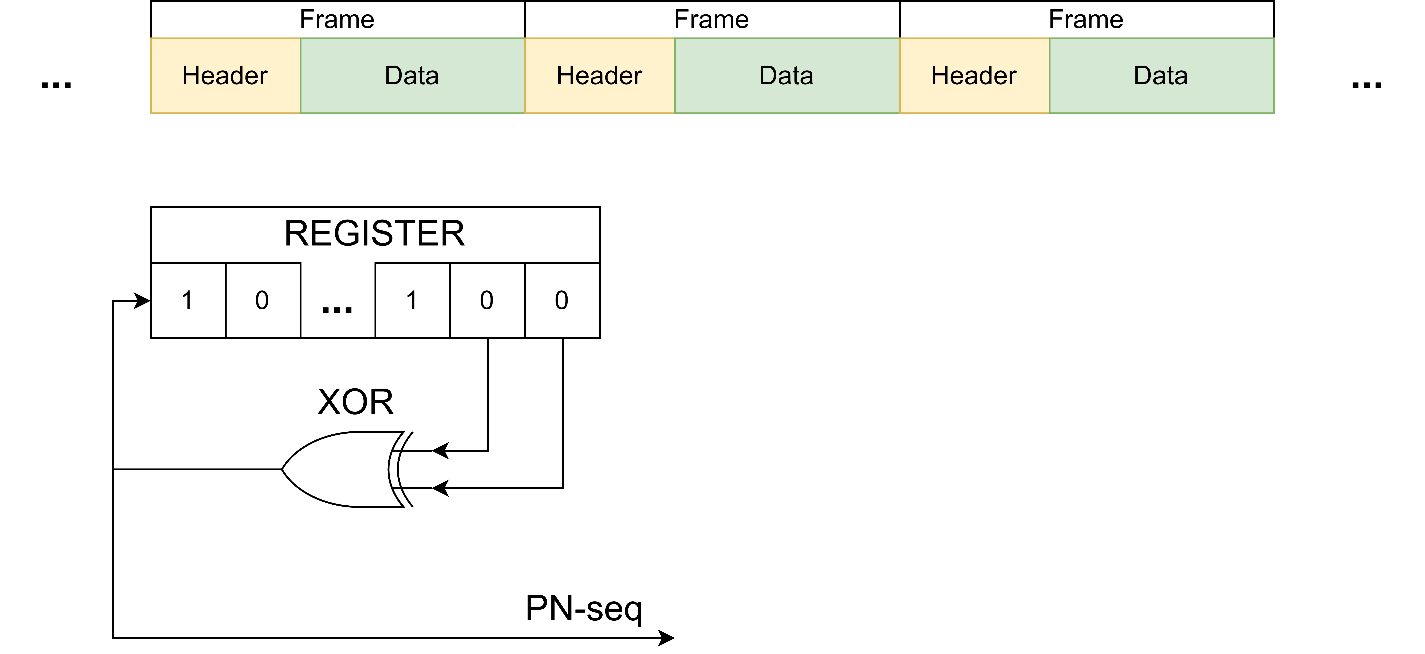
Header = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

Считая, что заголовок статичен и не меняется от кадра к кадру и зная саму последовательность Header необходимо:

1. Используя корреляционный анализ, найти позицию первого целого кадра данных. Результат сохранить в переменную Start\_Of\_Frame\_Position;
2. Определить количество кадров данных в предоставленной последовательности. Результат сохранить в переменную Number\_Of\_Frame.
3. Построить график корреляции от номера бита и сохранить его в файл Frame\_Corr.fig
4. Обе переменные записать в файл Frame\_search.mat

# Задание 2. Генератор псевдослучайной последовательности

1. Реализовать функцию генератора псевдослучайной последовательности на основе «Регистра с линейной обратной связью» (РСЛОС), как изображено на схеме ниже в виде функции Srambler(Register). Функция на вход принимает Register – начальное состояние регистров генератора.



1. С помощью циклической автокорреляционной функции оценить период повторения заданного генератора
2. Построить функцию автокорреляции от номера бита. Сохранить как ACF\_Srambler.fig.
3. Оценить период повторения псевдослучайной последовательности. Результат сохранить в переменную PN\_Period

Register = [1 0 0 1 0 1 0]

Полином:

# Задание 3. Цифровая обработка сигнала

## Предисловие

Это задания выполнить в одном скрипте за исключением функций. Все используемые в заданиях переменные и функции относятся и к другим упражнениям.

## Упражнение 1. Fourier transform

Дан сигнал **Signal** с известными гармониками :

Пусть размерность преобразования Фурье будет 128. С помощью обратного преобразования Фурье нужно вычислить этот сигнал **Signal**.

## Упражнение 2. Noise generation

Написать функцию **NoiseGenerator(SNR)**, генерирующую белый шум **Noise** с мощностью, заданной в виде SNR=23 dB, используя матлабовскую функцию **normrnd** в действительных и комплексных значениях. На вход функции подать значение SNR и исходный сигнал **Signal**. Сложить внутри данной функции сигналы **Signal** и **Noise**, полученный сигнал обозначить как **NoisedSignal** и подать его на выход функции.

## Упражнение 3. Powers of signals

Написать и встроить в скрипт функцию **PowerSignal**, рассчитывающую среднюю мощность сигнала. На вход данной функции подаётся исходный сигнал, а на выходе выдаётся его средняя мощность. В основном скрипте с помощью написанной функции рассчитать средние мощности сигнала, шума и зашумлённого сигнала, обозначить их **P\_Signal,** **P\_Noise,** **P\_NoisedSignal** соответственно.

## Упражнение 4. Parseval theorem

Проверить теорему Парсеваля. В дополнение к спектру сигнала **SignalSpec** вычислить спектры шума **NoiseSpec** и спектр результирующего сигнала **NoisedSignalSpec**. Вычислить с помощью написанной функции **PowerSignal** средние мощности спектров сигналов, сравнить со средними мощностями соответствующих им сигналов. Вывести **True**, если значения средних мощностей соответствующих сигналов и спектров совпадают с точностью до 0.1%, и **False**, если значения средних мощностей соответствующих сигналов и спектров не совпадают.

## Упражнение 5. Signal filtering

Написать функцию **FilterSignal**, реализующую фильтр LowPass, на вход которой подаётся исходный сигнал **NoisedSignal**. Фильтр пропускает частоты спектра до и с коэффициентом . Фильтрацию осуществлять путём перемножения спектров фильтра и спектра сигнала. Результатом функции будет сигнал **FilteredNoisedSignal**.

## Упражнение 6. SNR comparison

Вычислить и сравнить между собой SNR сигналов **NoisedSignal** и **FilteredNoisedSignal**. Дать ответ, какой сигнал лучше, и почему.