**Вопросы к экзамену по дисциплине «Информатика»**

**для студентов первого курса института «ИТ»**

[1 Что такое информация? Варианты определения данного понятия и их классификация. 1](#_Toc471567402)

[2 Свойства информации: объективность, точность, достоверность, полнота, актуальность, полезность, синтаксическая адекватность, связность, осмысленность, неисчерпаемость. 1](#_Toc471567403)

[3 Информатика как междисциплинарное научное направление: основные взгляды на содержание информатики. 2](#_Toc471567404)

[4 Информационное общество. Признаки перехода к информационному обществу. 2](#_Toc471567405)

[5 Системы счисления: определение, классификация, позиционные системы счисления и их основные понятия, сокращенная и полиномиальная запись чисел. 2](#_Toc471567406)

[6 Перевод целых чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму и методом подбора. 4](#_Toc471567407)

[7 Перевод вещественных чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму. 4](#_Toc471567408)

[8 Перевод вещественных чисел из системы с основанием N в десятичную. 4](#_Toc471567409)

[9 Определение количества разрядов, обеспечивающих достаточную точность, при переводе вещественного числа из десятичной системы в систему с основанием N. 5](#_Toc471567410)

[10 Перевод чисел из системы с основанием N в систему с основанием M, где M=NK и наоборот. 5](#_Toc471567411)

[11 Арифметические действия в позиционных системах счисления: сложение, вычитание, умножение, деление (на примере двоичной системы). 5](#_Toc471567412)

[12 Двоично-десятичная система: определение, достоинства и недостатки, правила выполнения сложения и вычитания. 6](#_Toc471567413)

[13 Прямой, обратный, обратный, дополнительный коды, модифицированные коды: определение, назначение, правила перевода, достоинства и недостатки. 7](#_Toc471567414)

[14 Понятие разрядной сетки. Понятие переполнения. Понятие машинного нуля. 11](#_Toc471567415)

[15 Сложение и вычитание в обратном и дополнительном коде. Переполнение и его устранение. 11](#_Toc471567416)

[16 Код со смещением: определение, назначение, правила выполнения сложения и вычитания. 11](#_Toc471567417)

[17 Представление чисел с фиксированной точкой: варианты фиксации точки для чисел со знаком и без, диапазоны представления чисел. Целочисленные типы данных. 12](#_Toc471567418)

[18 Представление чисел с плавающей точкой. Общая идея. Диапазон представления чисел. Понятие нормализованного и ненормализованного числа. 13](#_Toc471567419)

[19 Правила выполнения арифметических операций для чисел с плавающей точкой. Примеры. 13](#_Toc471567420)

[20 Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: общие правила представления мантиссы, общие правила представления порядка. 14](#_Toc471567421)

[21 Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат половинной точности. 15](#_Toc471567422)

[22 Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат одинарной точности. 16](#_Toc471567423)

[23 Алгоритмы перевода чисел из 10ой системы в форматы стандарта IEEE754 и наоборот. 16](#_Toc471567424)

[24 Базовые устройства схемотехники: понятие комбинационной схемы и цифрового автомата, классификация комбинационных схем и простых цифровых автоматов. 16](#_Toc471567425)

[25 Основы алгебры логики: логическая переменная и логическая функция, способы задания логической функции. 16](#_Toc471567426)

[26 Логические функции от двух переменных: названия, таблицы истинности, УГО. 17](#_Toc471567427)

[27 Основные понятия алгебры логики: конъюнкт, дизъюнкт, совершенный конъюнкт, совершенный дизъюнкт, минтерм, макстерм, дизъюнктивная форма, конъюнктивная форма. 17](#_Toc471567428)

[28 Совершенная дизъюнктивная нормальная форма, совершенная конъюнктивная нормальная форма. Определение. Методы построения. 17](#_Toc471567429)

[29 Основные логические законы и правила преобразования логических формул. 19](#_Toc471567430)

[30 Минимизация логических функций: цель минимизации, понятие МДНФ и МКНФ, минимизация методом эквивалентных логических преобразований. 20](#_Toc471567431)

[31 Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МДНФ функции от 3-х переменных, алгоритм минимизации. 20](#_Toc471567432)

[32 Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МДНФ функции от 4-х переменных, алгоритм минимизации. 20](#_Toc471567433)

[Построение МДНФ 21](#_Toc471567434)

[33 Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МКНФ функции от 3-х переменных, алгоритм минимизации. 24](#_Toc471567435)

[34 Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МКНФ функции от 4-х переменных, алгоритм минимизации. 24](#_Toc471567436)

[35 Минимизация частично определенных функций при помощи диаграмм Вейча. 24](#_Toc471567437)

[36 Приведение минимизированной логической функции к базису «ИЛИ-НЕ». 24](#_Toc471567438)

[37 Приведение минимизированной логической функции к базису «И-НЕ». 24](#_Toc471567439)

[38 Дешифраторы: определение, УГО, области применения, функциональная схема на примере дешифратора 2-4. 24](#_Toc471567440)

[39 Дешифраторы: определение, УГО, области применения, реализация логических функций на дешифраторах достаточной разрядности. 24](#_Toc471567441)

[40 Дешифраторы: определение, УГО, области применения, реализация логических функций на дешифраторах меньшей разрядности, чем количество переменных. 24](#_Toc471567442)

[41 Мультиплексоры: определение, УГО, области применения, функциональная схема мультиплексора на примере мультиплексора 4-1. 24](#_Toc471567443)

[42 Мультиплексоры: определение, УГО, области применения, реализация логических функций на мультиплексорах достаточной разрядности. 24](#_Toc471567444)

[43 Мультиплексоры: определение, УГО, области применения, реализация логических функций на мультиплексорах меньшей разрядности, чем количество переменных. 24](#_Toc471567445)

[44 Демультиплексоры: определение, УГО, области применения, функциональная схема демультиплексора на примере демультиплексора 1-4. 24](#_Toc471567446)

[45 Шифраторы: определение, УГО, области применения, таблица истинности и функциональная схема простого полного шифратора 4-2. 24](#_Toc471567447)

[46 Шифраторы: определение, УГО, области применения, таблица истинности и функциональная схема приоритетного шифратора 4-2. 24](#_Toc471567448)

[47 Сумматоры: определение, УГО, классификация, четвертьсумматор, полусумматор. 24](#_Toc471567449)

[48 Сумматоры: определение, УГО, полный одноразрядный сумматор, многоразрядный сумматор параллельного действия с последовательным переносом. 24](#_Toc471567450)

[49 Сумматоры: определение, УГО, сумматор последовательного действия – принцип работы, обобщенная структурная схема, достоинства и недостатки. 24](#_Toc471567451)

[50 Сумматоры: определение, УГО, сумматоры с параллельным переносом – рекуррентная формула для вычисления переносов, пример схемы для 2х разрядного сумматора, достоинства и недостатки. 24](#_Toc471567452)

[51 Компараторы: определение, таблица истинности, выражения для вычисления отношений «равно» и «больше» на примере компаратора двухразрядных чисел. 25](#_Toc471567453)

[52 Триггеры: определение, классификация, способы синхронизации, области применения. Сигнал синхронизации: назначение, структура. 25](#_Toc471567454)

[53 Триггеры: асинхронный RS-триггер на элементах «И-НЕ» и на элементах «ИЛИ-НЕ», его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567455)

[54 Триггеры: синхронный RS-триггер на элементах «И-НЕ» со статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567456)

[55 Триггеры: синхронный двухступенчатый RS-триггер с асинхронными входами и статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567457)

[56 Триггеры: синхронный RS-триггер с динамическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567458)

[57 Триггеры: синхронный D-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567459)

[58 Триггеры: синхронный T-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567460)

[59 Триггеры: синхронный двухступенчатый JK-триггер с асинхронными входами предустановки, его таблица истинности, функциональная схема, объяснение принципа работы, УГО. 25](#_Toc471567461)

[60 Регистры: определение, выполняемые функции, классификация, виды сдвига. 25](#_Toc471567462)

[61 Четырехразрядный параллельный регистр на D-триггерах: УГО, внутреннее устройство, выполняемые функции. 25](#_Toc471567463)

[62 Четырехразрядный регистр с последовательными приемом и выдачей на D-триггерах, с выбором направления сдвига: УГО, внутреннее устройство, варианты использования. 25](#_Toc471567464)

[63 Четырехразрядный регистр с параллельно-последовательным приемом и выдачей, реализующий сдвиг вправо: УГО, внутреннее устройство, варианты использования. 25](#_Toc471567465)

[64 Универсальный сдвиговый регистр: УГО, внутреннее устройство (на примере одного разряда), варианты использования. 25](#_Toc471567466)

[65 Счетчики: определение, основные параметры, классификация. 25](#_Toc471567467)

[66 Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с последовательным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки. 25](#_Toc471567468)

[67 Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с ускоренным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки. 25](#_Toc471567469)

[68 Синтез оптимальных счетчиков с требуемым модулем, шагом и направлением на D-триггерах. 25](#_Toc471567470)

[69 Быстрый синтез счетчиков с требуемым модулем, шагом и направлением на D-триггерах. 26](#_Toc471567471)

[70 Основы алгоритмизации. Понятие алгоритма, свойства алгоритмов. 26](#_Toc471567472)

[71 Основы алгоритмизации. Понятие алгоритма, правила построения блок-схем. 26](#_Toc471567473)

[72 Основы алгоритмизации. Алгоритм поиска максимума и минимума. 26](#_Toc471567474)

[73 Основы алгоритмизации. Принцип структурного программирования Дейкстры. 26](#_Toc471567475)

[74 Основы алгоритмизации. Алгоритм сортировки «Пузырек». 26](#_Toc471567476)

[75 Основы алгоритмизации. Алгоритм быстрой сортировки Хоара. 26](#_Toc471567477)

[76 Основы алгоритмизации. Рекурсия – назначение, виды, примеры организации. 26](#_Toc471567478)

[77 Основы алгоритмизации. Проверка вводимых данных – типичные ошибки и методы борьбы с ними. 26](#_Toc471567479)

# Что такое информация? Варианты определения данного понятия и их классификация.

Информация – модель объекта/явления. Informatio – понятие, суть, идея без материального носителя.

Любой смысл, получаемый из внешнего мира в процессе взаимодействия с ним.

Передача сигнала, в процессе которой устраняется неопределенность.

Совокупность сведений, которые циркулируют в природе и обществе.

Определения информации

без учета смысловой составляющей

с учетом смысловой составляющей

смысловая составляющая новая

смысловая составляющая не новая

С учетом смысловой составляющей:

**Информация** – это новые сведения о чем-либо, полученные при помощи некоторого метода интерпеитатации данных.

**Данные** – это сигналы, зафиксированные на материальном носителе, искуственным или естественным путем.

**Сигнал** – это событие, повлекшее изменение некоторой физической величины.

Без метода интерпертации данных данные бесполезны.

**Метод интерпретации** накладывает определенную структуру на данные и предписывает определенные действия для получения полезных сведений из данных.

# Свойства информации: объективность, точность, достоверность, полнота, актуальность, полезность, синтаксическая адекватность, связность, осмысленность, неисчерпаемость.

* объективность:

не зависит от методов ее фиксации, чьего-либо мнения.

* точность:

определяется степенью ее близости к реальному состоянию.

* достоверность:

общая точность и полнота информации. Объективная информация всегда достоверна, субъективная – не всегда.

* полнота:

информация может считаться полной, если достаточна для решения задачи.

* актуальность:

степень важности информации непосредственно в данных условиях/времени.

* полезность:

степень применимости пользователем.

* синтаксическая адекватность:

Характеризуется возможностью успешно получить информацию(интерпретировать данные):

* тип носителя
* скорость передачи
* способы кодировки
* точность предстваления
* связность:

привязанность к носителю.

* осмысленность:

смысл сохраняется независимо от формы предстваления информации.

* неисчерпаемость:

физически не стареет, тиражируется, копится, передается от поколения к поколению.

# Информатика как междисциплинарное научное направление: основные взгляды на содержание информатики.

Информатика – междисциплинарное научное направление, изучающее вопросы производства, хранения, накопления, передачи, обработки и использования информации.

1 концепция – информатика без вычислительной техники существовать не может.

2 концепция – информатика, как методология работы с информацие, существовала в том или ином виде всегда. Информатика – информационная культура личности человека.

# Информационное общество. Признаки перехода к информационному обществу.

1. Объектом и результатом труда большей части населения является информация.

2. Информация – това, определяющий основные экономические показатели.

3. Нет проблем с доступом к информации.

4. Существует необходимая техническая база для распространения и обработки информации.

5. Информация – оружие.

# Системы счисления: определение, классификация, позиционные системы счисления и их основные понятия, сокращенная и полиномиальная запись чисел.

Система счисления(СС) — правила наименования и записи чисел, а также выполнения арифметических операций. В каждой СС есть свой набор базовых символов.

СС

позиционные

не позиционные

традиционные

k-ичные

2-ичная

10-ичная

…

смешанные

k-q-ичные

2-10-чная

…

аддитивные

факторианая

фибоначчева

…

прочие

В позиционных СС значение цифры зависит от позиции (разряда).

# Перевод целых чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму и методом подбора.

Машинный алгоритм.

12510

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 125 | 2 |  |  |  |  |  |
| 124 | 62 | 2 |  |  |  |  |
| 1 | 62 | 31 | 2 |  |  |  |
|  | 0 | 30 | 15 | 2 |  |  |
|  |  | 1 | 14 | 7 | 2 |  |
|  |  |  | 1 | 6 | 3 | 2 |
|  |  |  |  | 1 | 2 | 1 |
|  |  |  |  |  | 1 |  |

12510=11111012

Метод подбора.

34210=256+(342-256)=28+86=28+64+(86-64)=28+26+22=28+26+16+4+2=28+26+24+22+21

34210=1\*28+0\*27+26+0\*25+24+0\*23+22+21+0\*20

34210=101010110

# Перевод вещественных чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму.

0.62510

|  |  |
| --- | --- |
| \* | 0.625 |
| 2 |
|  | 1.250 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | \* | 0.625 | | 2 | |  | 1.250 | | |  |  | | --- | --- | | \* | 0.250 | | 2 | |  | 0.500 | | |  |  | | --- | --- | | \* | 0.500 | | 2 | |  | 1.000 | |
| **1** | **0** | **1** |
|  | | |

0.62510=0.1012

# Перевод вещественных чисел из системы с основанием N в десятичную.

# Определение количества разрядов, обеспечивающих достаточную точность, при переводе вещественного числа из десятичной системы в систему с основанием N.

Точность считается достаточной, если погрешность составляет не более половины младшего разряда

p-ичная СС была.

В k-ичную переводим.

L – искомое кол-во разрядов в k-ичной СС.

m – число разрядов после запятой.

0.71410

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  | |

# Перевод чисел из системы с основанием N в систему с основанием M, где M=NK и наоборот.

11 011 110 010 001 1112=X8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | **0**11 | 011 | 110 | 010 | 001 | 111 |
| 8 | 3 | 3 | 6 | 2 | 1 | 7 |

11 011 110 010 001 1112=3362178

7AF916

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | 7 | A | F | 9 |
| 2 | 0111 | 1010 | 1111 | 1001 |

7AF916=111 1010 1111 10012

1642.2578=001 110 100 010.010 101 1112

# Арифметические действия в позиционных системах счисления: сложение, вычитание, умножение, деление (на примере двоичной системы).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1101  + 101  ------  10010 | 1110  - 101  ----  1001 | 1110  \* 11  ------  + 1110  1110  ------  101010 | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |  |  | |  | 1 | 1 |  |  |  | 1 | 1 | 1 | 0 | |  | 1 | 0 | 0 |  |  |  |  |  |  | |  |  | 1 | 1 |  |  |  |  |  |  | |  |  |  | 1 | 1 |  |  |  |  |  | |  |  |  | 1 | 1 |  |  |  |  |  | |  |  |  |  | 0 | 0 |  |  |  |  | |  |  |  |  | 0 | 0 |  |  |  |  | |  |  |  |  |  | 0 |  |  |  |  | |

# Двоично-десятичная система: определение, достоинства и недостатки, правила выполнения сложения и вычитания.

Каждый десятичный разряд числазаписывается в виде четырехбитного двоичного кода.

|  |  |
| --- | --- |
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | не используются | |
| … |
| 1111 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 6 | 9 | 1 |
| 0001 | 0110 | 1001 | 001 |

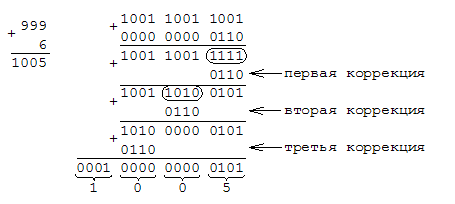
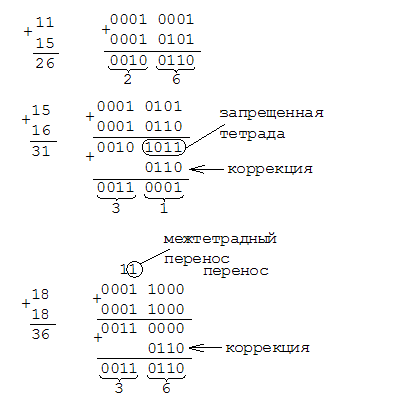
Достоинства: легче читать, проще вывести на семисегментный индикатор, проще ввести с клавиатуры, не теряется точность для вещественных чисел, упрощены умножение и деление на 10.

Недостатки: повышенный расход памяти, усложнены арифметические операции.

Применение: калькуляторы, часы, кодовые замки

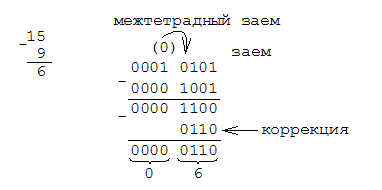
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 4 | 6 | 7 | 10 | = | + | 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 |  | 0 | 1 | 1 | 1 | 2-10 |  |  |  |  |  |  |  |  |  |  |  |
| 7 | 5 | 8 | 10 | = | 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 1 |  | 1 | 0 | 0 | 0 | 2-10 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | + | 1 | 0 | 1 | 1 | + | 1 | 0 | 1 | 1 | + | 1 | 1 | 1 | 1 |  | получены запрещенные тетрады  коррекция (15-9)=+6 | | | | | | | | | | |
|  |  |  |  |  |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |  |
|  |  | 0 | 0 | 0 | 1 |  | 0 | 0 | 1 | 0 |  | 0 | 0 | 1 | 0 |  | 0 | 1 | 0 | 1 |  |
|  |  | 1 | | | |  | 2 | | | |  | 2 | | | |  | 5 | | | |  |  |  |  |  |  |  |  |  |  |  |  |

Коррекция проводится, если получена запрещенная тетрада или был произведен перенос из младшей тетрады в старшую.



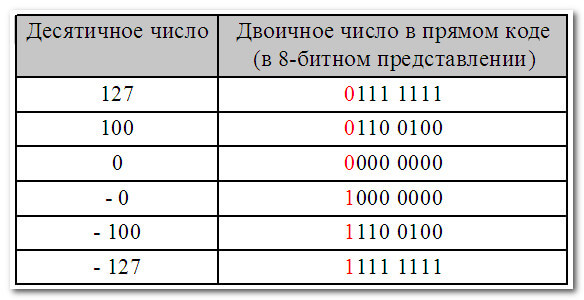
В случае вычитания коррекция делается в ту тетраду, к-я получает заем из старшей.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | 7 | 5 | 8 | 10 | = | - | 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 1 |  | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |
| 4 | 6 | 7 | 10 | = | 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 |  | 0 | 1 | 1 | 1 | 2-10 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 0 | 0 | 1 | 0 | - | 1 | 1 | 1 | 1 |  | 0 | 0 | 0 | 1 |  | получены запрещенные тетрады  коррекция -(15-9)=-6 | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  | 0 | 1 | 1 | 0 |  |  |  |  |  |
|  |  |  |  |  |  |  | 0 | 0 | 1 | 0 |  | 1 | 0 | 0 | 1 |  | 0 | 0 | 0 | 1 |  |
|  |  |  | | | |  | 2 | | | |  | 9 | | | |  | 1 | | | |  |  |  |  |  |  |  |  |  |  |  |



# Прямой, обратный, обратный, дополнительный коды, модифицированные коды: определение, назначение, правила перевода, достоинства и недостатки.

Прямой код предназначен для отображения целых и дробных чисел со знаком.



Обратный код. Положительные такие же, как в прямом коде, а отрицательные инвертируются.

Обратный код позволяет вычесть одно число из другого, используя операцию сложения.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 10 | + | 8 | 10 | = | 2 | 2 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 4 | 10 | = | 0 | . | 1 | 1 | 1 | 0 | пр | = | 0 | . | 1 | 1 | 1 | 0 | об |  |  |  |  |  |  |  |
|  | 8 | 10 | = | 0 | . | 1 | 0 | 0 | 0 | пр | = | 0 | . | 1 | 0 | 0 | 0 | об |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | + | 0 | . | 1 | 1 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 0 | . | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 1 | . | 0 | 1 | 1 | 0 | - переполнение (1. – отрицательное) | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | + | 0 | . | 0 | 1 | 1 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 0 | . | 0 | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 0 | . | 1 | 0 | 1 | 1 | 0 | = | 2 | 2 | 10 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 4 | 10 | - | 8 | 10 | = | 2 | 2 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 4 | 10 | = | 0 | . | 1 | 1 | 1 | 0 | пр | = | 0 | . | 1 | 1 | 1 | 0 | об |  |  |  |  |  |  |  |
| - | 8 | 10 | = | 1 | . | 1 | 0 | 0 | 0 | пр | = | 1 | . | 0 | 1 | 1 | 1 | об |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | + | 0 | . | 1 | 1 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1 | . | 0 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | + | **1** | 0 | . | 0 | 1 | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 0 | . | 0 | 1 | 1 | 0 | = | 6 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| - | 1 | 4 | 10 | = | 1 | . | 1 | 1 | 1 | 0 | пр | = | 1 | . | 0 | 0 | 0 | 1 | об |  |  |  |  |  |  |
|  | - | 8 | 10 | = | 1 | . | 1 | 0 | 0 | 0 | пр | = | 1 | . | 0 | 1 | 1 | 1 | об |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | + | 1 | . | 0 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1 | . | 0 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | + | **1** | 0 | . | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 0 | . | 1 | 0 | 0 | 1 | - переполнение | | | | | | | | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | + | 1 | . | 1 | 0 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1 | . | 1 | 0 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | + | **1** | 1 | . | 0 | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | **1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 1 | . | 0 | 1 | 0 | 0 | 1 | = | 1 | . | 1 | 0 | 1 | 1 | 0 | пр | = | - | 2 | 2 | 10 |  |

Дополнительный код.

Положительные такие же как и в прямом и обратном кодах.

Отрицательные:

1 способ: к младшему разряду обратного кода прибавляется единица.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | 1 | 4 | 10 | = | 1 | . | 1 | 1 | 1 | 0 | пр | = | 1 | . | 0 | 0 | 0 | 1 | об | = | 1 | . | 0 | 0 | 1 | 0 | дп |
|  | - | 8 | 10 | = | 1 | . | 1 | 0 | 0 | 0 | пр | = | 1 | . | 0 | 1 | 1 | 1 | об | = | 1 | . | 1 | 0 | 0 | 0 | дп |

2 способ: переписываем прямой код справа налево до первой единицы включительно, остальные инвертируем.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | 1 | 4 | 10 | = | 1 | . | 1 | 1 | 1 | 0 | пр | = | 1 | . | 0 | 0 | 1 | 0 | дп |
|  | - | 8 | 10 | = | 1 | . | 1 | 0 | 0 | 0 | пр | = | 1 | . | 1 | 0 | 0 | 0 | дп |

3 способ: вычитаем из нуля двоичный код числа.



Сложение:

Вычеркиваем все символы справа от первого после точки.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 6 | 10 | = | 0 | . | 1 | 1 | 0 | пр | = | 0 | . | 1 | 1 | 0 | дп |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| - | 6 | 10 | = | 1 | . | 1 | 1 | 0 | пр | = | 1 | . | 0 | 1 | 0 | дп |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | + | 0 | . | 1 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | . | 0 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 0 | . | 0 | 0 | 0 | = | 0 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | - | 8 | 10 | = | 1 | . | 1 | 0 | 0 | 0 | пр | = | 1 | . | 1 | 0 | 0 | 0 | дп |  |  |  |  |  |  |  |  |  |  |  |
| - | 1 | 4 | 10 | = | 1 | . | 1 | 1 | 1 | 0 | пр | = | 1 | . | 0 | 0 | 1 | 0 | дп |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | + | 1 | . | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | . | 0 | 0 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | 0 | . | 1 | 0 | 1 | 0 | - переполнение (получили положительное знач.) | | | | | | | | | | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | + | 1 | . | 1 | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | . | 1 | 0 | 0 | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1 | 1 | . | 0 | 1 | 0 | 1 | 0 | = | 1 | . | 0 | 1 | 0 | 0 | 1 | = | 1 | . | 1 | 0 | 1 | 1 | 0 | пр | = | 2 | 2 | 10 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Модифицированный код.

1.1011 – отрицательное число или положительное с переполнением?

Модификация – два разряда для знака.

00. – положительное.

11. – отрицательное.

01. – положительное с переполнением.

10. – отрицательное с переполнением.

В дополнительном коде сложение отрицательных и положительных чисел происходит по одному правилу. + меньше памяти занимает.

# Понятие разрядной сетки. Понятие переполнения. Понятие машинного нуля.

Разрядная сетка – это кол-во двоичных разрядов, выделяемых в памяти для отображения чисел. Она напрямую зависит от разрядности процессора. Если число вышло за пределы разрядной сетки слева, то говорят о переполнении разрядной сетки. Если справа – то о возникновении машинного нуля.

# Сложение и вычитание в обратном и дополнительном коде. Переполнение и его устранение.

см. 13.

# Код со смещением: определение, назначение, правила выполнения сложения и вычитания.

смещение +3

|  |  |  |  |
| --- | --- | --- | --- |
|  | прямой код | смещенные числа | код со смещением |
| 4 | 0.100 | 7 | 111 |
| 3 | 0.11 | 6 | 110 |
| 2 | 0.10 | 5 | 101 |
| 1 | 0.01 | 4 | 100 |
| 0 | 0.00 | 3 | 011 |
| -1 | 1.01 | 2 | 010 |
| -2 | 1.10 | 1 | 001 |
| -3 | 1.11 | 0 | 000 |

Код со смещением позволяет сдвинуть числовую шкалу, содержащию как положительные, так и отрицательные числа, полностью в область положительных чисел.

–максимальное число в смещенном коде.

1- – минимальное число в смещенном коде.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | A | + | смещение |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | B | + | смещение |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | - | A+B | + | 2\* смещение |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | смещение |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | A+B | + | смещение |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

При вычитании аналогично прибавлять смещение.

# Представление чисел с фиксированной точкой: варианты фиксации точки для чисел со знаком и без, диапазоны представления чисел. Целочисленные типы данных.

Форма прдставления числа

С фиксированной точкой

С плавающей точкой

справа от младшего разряда

слева от старшего разряда

Где-то в середине сетки

справа от младшего разряда:

Целые числа. Точка формально находится справа от младшего разряда, т.е. вне разрядной сетки.

без знака:

unsigned int, unsigned short int, unsigned long int, unsigned char.

со знаком:

short int, int, long int.

слева от старшего разряда:

Аналогично, только все числа меньше единицы, т.е. дробные. Используется в специальных процессорах.

без знака:

со знаком:

С точкой, фиксированной где-то в середине сетки.

k – число целых разрядов.

m – число дробных разрядов.

без знака:

|  |  |
| --- | --- |
| целая часть (k) | дробная часть(m) |
| n=k+m | |

со знаком:

|  |  |  |
| --- | --- | --- |
| знак | целая часть (k) | дробная часть(m) |
| n=k1+m1+1 | | |

# Представление чисел с плавающей точкой. Общая идея. Диапазон представления чисел. Понятие нормализованного и ненормализованного числа.

|  |  |  |  |
| --- | --- | --- | --- |
| знак порядка | знак мантиссы | порядок | мантисса |
| 1 | 1 | k | m |
| n=k+m+2 | | | |

Число считается нормализованным, если в старшем разряде после запятой стоит значащая цифра (в двоичной системе это 1).

# Правила выполнения арифметических операций для чисел с плавающей точкой. Примеры.

Сложение/вычитание:

1. Выравниваем порядки.

2. Складываем/вычитаем.

3. Нормализуем.

Умножение:

1. Складываем порядки.

2. Перемножаем мантиссы.

3. Нормализуем.

Деление:

1. Вычитаем порядок делителя из порядка делимого.

2. Делим.

3. Нормализуем.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | . | 1 | \* | 2 | ^ | 5 |  | и |  | 0 | . | 1 | \* | 2 | ^ | 3 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | + | 0 | . | 1 | 0 | 0 | \* | 2 | ^ | 5 |  | + | 0 | . | 1 | 0 | 0 | \* | 2 | ^ | 5 |  |  |  |  |  |  |  |  |  |
|  | 0 | . | 0 | 0 | 1 | \* | 2 | ^ | 5 |  | 0 | . | 0 | 0 | 1 | \* | 2 | ^ | 5 |  |  |  |  |  |  |  |  |  |
|  |  | 0 | . | 1 | 0 | 1 | \* | 2 | ^ | 5 |  |  | 0 | . | 0 | 1 | 1 | \* | 2 | ^ | 5 | = | 0 | . | 1 | 1 | \* | 2 | ^ | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | \* | 0 | . | 1 | \* | 2 | ^ | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | . | 1 | \* | 2 | ^ | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | . | 0 | 1 | \* | 2 | ^ | 8 | = | 0 | . | 1 | \* | 2 | ^ | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | . | 1 | \* | 2 | ^ | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | . | 1 | \* | 2 | ^ | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 1 | \* | 2 | ^ | 2 | = | 0 | . | 1 | \* | 2 | ^ | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: общие правила представления мантиссы, общие правила представления порядка.

Представление мантиссы:

В записи числа используется нормализованная мантисса.

Но реализация нормализации отличается от общей идеи. Дело в том, что в «традиционном» нормализованном числе 1 в старшем бите мантиссы есть всегда. Следовательно, ее можно не сохранять, а просто считать, что она там есть.

Поэтому стандарт определяет мантиссу следующим образом: она состоит из неявного бита, который всегда равен 1, двоичной точки и остальных разрядов. Получается, что мантисса охватывает диапазон чисел [1,2).

Порядок числа записывается в смещенном коде, т.е. к нему прибавляется фиксированное число, чтобы порядок был всегда неотрицательным. Это упрощает выполнения операций над порядками, избавляет от знакового разряда порядка.

Истинный порядок может быть как положительным, так и отрицательным. Все доступные разряды порядка разделяются поровну между его положительными и отрицательными значениями. При выполнение арифметических операций процессор учитывает сдвиг.

Одна комбинация резервируется для спец нужд.

# Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат половинной точности.

Всего 16 разрядов(бит)

Сдвиг порядка = 15

Ограничения точности для целых чисел:

От 0 до 2048 передаются как есть.

2049 – 4096 округляются к ближайшему четному числу.

4097 – 8192 округляются к ближайшему, делящемуся нацело на 4.

8193 – 16384 округляются к ближайшему, делящемуся нацело на 8.

16385 – 32768 округляются к ближайшему, делящемуся нацело на 16.

32769 – 65535 округляются к ближайшему, делящемуся нацело на 32.

Почему, например, числа от 4097 до 8192 округляются до ближайшего целого, делящегося нацело на 4?



Число вышло за РС справа на два разряда, значит оно уменьшилось в 22=4 раза !

Смещение равно , где n – число бит порядка.

# Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат одинарной точности.

Всего 32 разряда (float)

порядок – 8 бит.

мантисса – 23+1 бит.

знак – 1 бит.

2-ой точности (double):

Всего 64 разряда

порядок – 11 бит.

мантисса – 52+1 бит.

знак – 1 бит.

# Алгоритмы перевода чисел из 10ой системы в форматы стандарта IEEE754 и наоборот.

1. Получить число в прямом коде.

2. Нормализовать.

3. Сместить порядок.

4. Разместить в сетке знак порядка и мантиссу.

# Базовые устройства схемотехники: понятие комбинационной схемы и цифрового автомата, классификация комбинационных схем и простых цифровых автоматов.

Комбинационные схемы – состояние на выходе в данный момент времени однозначно определяется состояниями на входах в тот же момент времени.

элементы «И», «ИЛИ», «НЕ», «И-НЕ», «ИЛИ-НЕ» (исключение - «ИЛИ»)

* мультипликаторы и демультипликаторы
* шифраторы и дешифраторы
* компараторы
* комбинационные сумматоры
* составные из вышеназванных

Простые цифровые автоматы – состояние на выходе определяется не только состояниями на входах в данный момент времени, но и предыдущим состоянием системы.

* триггеры
* регистры
* счетчики
* составные из вышеназванных

# Основы алгебры логики: логическая переменная и логическая функция, способы задания логической функции.

Логическая переменная – булева переменная (TRUE/FALSE, 1/0)

Логическая функция – функция от логических переменных/функций. Тоже может быть T/F.

Благодаря тому, что каждая логическая переменная имеет только два значения, множество различных комбинаций входных значений переменных дискретно, конечно и перечеслимо.

Для k логических переменных существует комбинаций и уникальных логических функций.

Способы задания:

* Словесный:

описывается фразой на естественном языке.

* Табличный: таблица истинности.
* Аналитический: формулой
* Векторный: функция задается только перечислением значений на различных наборах. Сами наборы и их кол-во однозначно востанавливаются по кол-ву значений функции.
* Графический: временная диаграмма, осцилограмма, фиклограмма.
* Схемотехнический: комбинационная схема, к-я реализует функцию.

# Логические функции от двух переменных: названия, таблицы истинности, УГО.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ФУНКЦИЯ | X1 | 0 | 0 | 1 | 1 |
| X2 | 0 | 1 | 0 | 1 |
| константа – 0 | F0 | 0 | 0 | 0 | 0 |
| конъюнкция | F1 | 0 | 0 | 0 | 1 |
|  | F2 | 0 | 0 | 1 | 0 |
| повторение X1 | F3 | 0 | 0 | 1 | 1 |
|  | F4 | 0 | 1 | 0 | 0 |
| повторение X2 | F5 | 0 | 1 | 0 | 1 |
| сложение по модулю 2 (исключающее «ИЛИ») (либо …, либо…) | F6 | 0 | 1 | 1 | 0 |
| дизъюнкция | F7 | 0 | 1 | 1 | 1 |
| элемент Вебба (стрелка Пирса) «ИЛИ-НЕ» | F8 | 1 | 0 | 0 | 0 |
|  | F9 | 1 | 0 | 0 | 1 |
|  | F10 | 1 | 0 | 1 | 0 |
| обратная импликация () | F11 | 1 | 0 | 1 | 1 |
|  | F12 | 1 | 1 | 0 | 0 |
| импликация (следование) () | F13 | 1 | 1 | 0 | 1 |
| функция Шеффера (штрих Шеффера) «И-НЕ» | F14 | 1 | 1 | 1 | 0 |
| константа – 1 | F15 | 1 | 1 | 1 | 1 |

# Основные понятия алгебры логики: конъюнкт, дизъюнкт, совершенный конъюнкт, совершенный дизъюнкт, минтерм, макстерм, дизъюнктивная форма, конъюнктивная форма.

**Конъюнкт** – конъюнкция некоторого кол-ва переменных или их отрицаний.

**Дизъюнкт** – дизъюнкция некоторого кол-ва переменных или их отрицаний.

Если конъюнкт или дизъюнкт состоит из всех переменных функции, где каждая переменная участвуюет единожды, то он **совершенный**.

**Минтерм** (совершенный конъюнкт, конституента единицы) – конъюнкция всех переменных, которые входят в прямом виде, если значение данной переменной в точке определения равно **1**, либо в инверсном виде, если значение переменной равно **0**. Принимает значение **1** лишь на одном наборе значений переменных (например: ). В форму записи минтерма аргумены равные нулю входят с отрицанием. Всего для функции от переменных можно сформировать минтермов.

**Макстерм** (совершенный дизъюнкт, конституента нуля) – это дизъюнкция всех переменных, которые входят в прямом виде, если значение данной переменной в точке области определения равно **0**, либо в инверсном виде, если значение переменной равно **1**. Принимает значение **0** лишь на одном наборе своих аргументов.

**ДНФ** – дизъюнкция конечного числа конъюнктов.

**КНФ** – конъюнкция конечного числа дизъюнктов.

# Совершенная дизъюнктивная нормальная форма, совершенная конъюнктивная нормальная форма. Определение. Методы построения.

**СДНФ** – дизъюнкция совершенных конъюнктов.

**СКНФ** – конъюнкция совершенных дизъюнктов.

Любая логическая функция, не являющаяся нулем имеет только одну СДНФ.

Любая логическая функция, не являющаяся единицей имеет только одну СКНФ.

Любая логическая функция, не являющаяся конституентой нуля или конституентой единицы называется выполнимой. Функции принимающие значение единицы на всех наборах аргументов называется тождественно истинными. Нуля – тождественно ложными.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Постороение СДНФ.

Выписать совершенные конъюнкции и связать их дизъюнкциями.

Построение СКНФ.

Выписать совершенные дизъюнкции и связать их конъюнкцией

# Основные логические законы и правила преобразования логических формул.

Коммутация:

ассоциативный:

Дистрибутивный:

Закон идемпотентости:

Закон противоречия

Закон исключения третьего:

Закон исключения констант:

Закон исключения (правило склеивания):

Закон общей инверсии (законы де Моргана:

Преобразования:

Коммутация:

Коммутация:

# Минимизация логических функций: цель минимизации, понятие МДНФ и МКНФ, минимизация методом эквивалентных логических преобразований.

**МДНФ** – минимальная *ДНФ.*

**МКНФ** – минимальная *КНФ.*

**Минимизация** – нахождение нормальной формы минимальной сложности.

**Минимальная нормальная форма** – функция с минимальным кол-во переменных.

**Метод эквивалентных логических преобразований** – см. пункт 29.

# Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МДНФ функции от 3-х переменных, алгоритм минимизации.

См. 32.

# Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МДНФ функции от 4-х переменных, алгоритм минимизации.

Решение:

F=6EB9;

F2=0110111010111001

Таблица истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F(A,B,C,D)** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Построение МДНФ:





МДНФ в базисе «И-НЕ»:

МДНФ в базисе «ИЛИ-НЕ»:

**Построение МКНФ:**



берем дизъюнкты с отрицанием каждого члена отдельно:

МКНФ в базисе «И-НЕ»:

МКНФ в базисе «ИЛИ-НЕ»:

# Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МКНФ функции от 3-х переменных, алгоритм минимизации.

См. 32.

# Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МКНФ функции от 4-х переменных, алгоритм минимизации.

См. 32.

# Минимизация частично определенных функций при помощи диаграмм Вейча.

См. 32.

# Приведение минимизированной логической функции к базису «ИЛИ-НЕ».

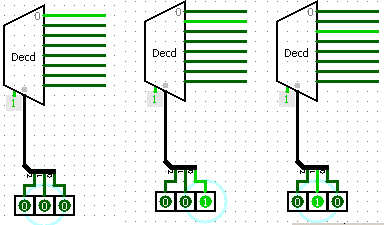
См. 32.

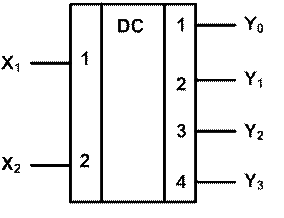
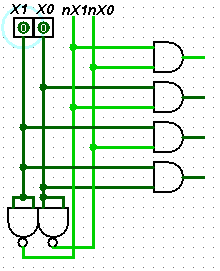
# Приведение минимизированной логической функции к базису «И-НЕ».

См. 32.

# Дешифраторы: определение, УГО, области применения, функциональная схема на примере дешифратора 2-4.

Дешифратор – комбинационная схема, обладающая N адресными входами, одним разрешающим входом E и выходами. На адресные входы подается двоичное число, которое в своем десятичном представлении задает номер выхода. На выходе – унитарный код.

**

**

Области применения:

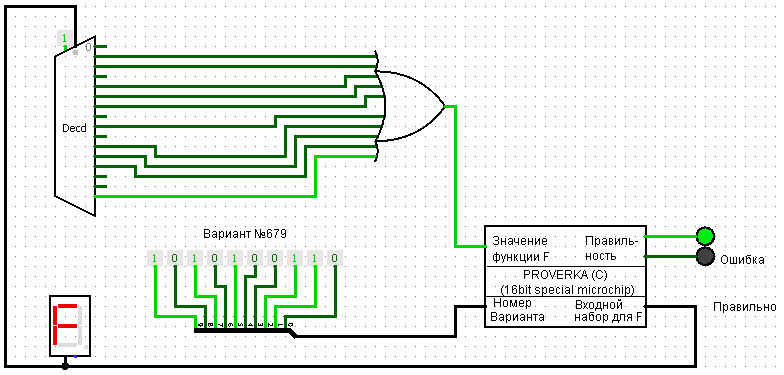
* в составе схем управления другими устройствами для последовательной подачи разрешающих сигналов
* в составе схем преобразователей кодов
* для реализации логических функций

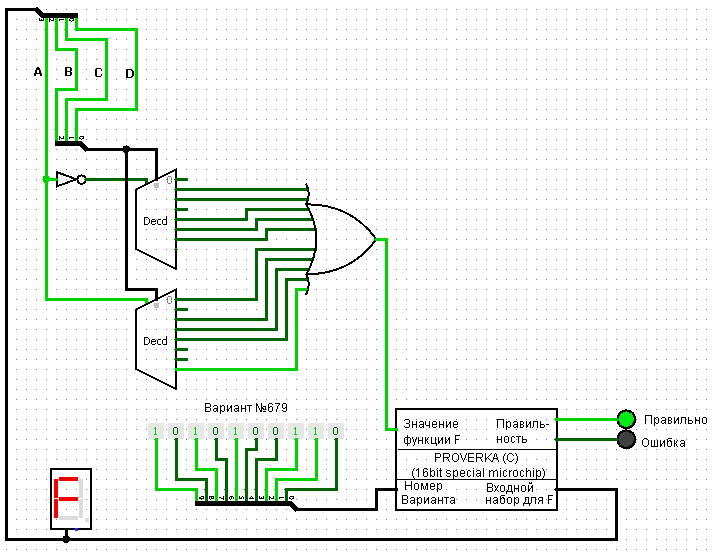
# Дешифраторы: определение, УГО, области применения, реализация логических функций на дешифраторах достаточной разрядности.

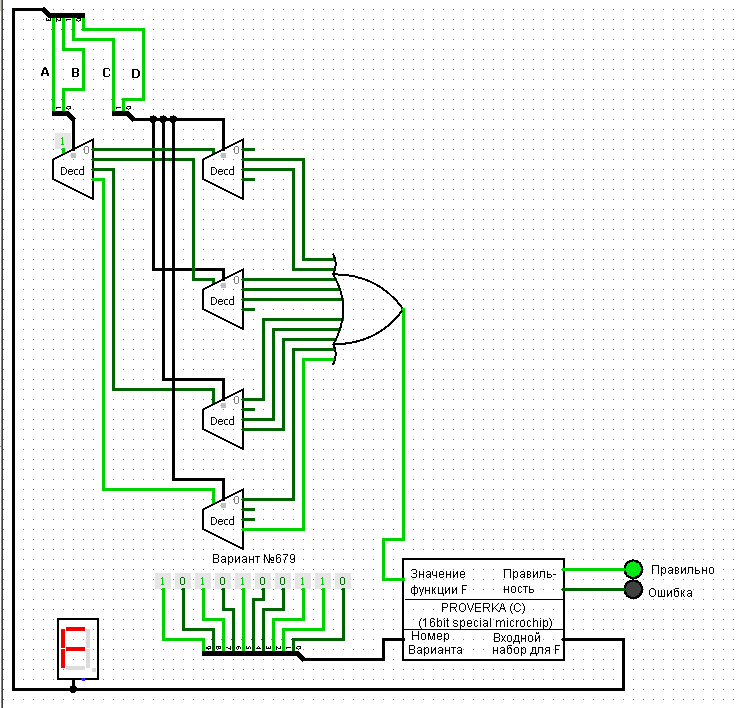
см. 38.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F(A,B,C,D)** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Реализация функции при помощи дешифратора 4-16.**



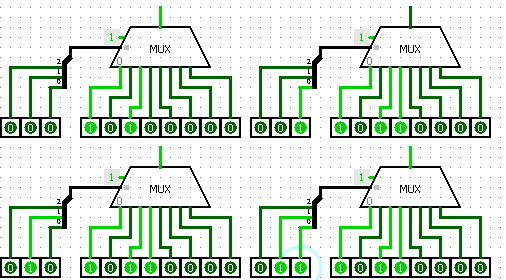
**Реализация функции при помощи дешифраторов 3-8.**

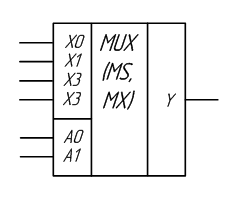
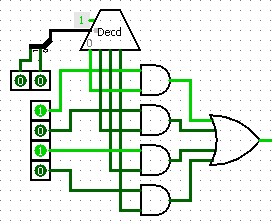
**Реализация функции при помощи дешифраторов 2-4.**

# Дешифраторы: определение, УГО, области применения, реализация логических функций на дешифраторах меньшей разрядности, чем количество переменных.

см. 40.

# Мультиплексоры: определение, УГО, области применения, функциональная схема мультиплексора на примере мультиплексора 4-1.

Мультиплексор – комбинационная схема, имеющая информационных входов, N адресных входов, (разрешающий вход – опционально) и одним выходом.

**

Области применения:

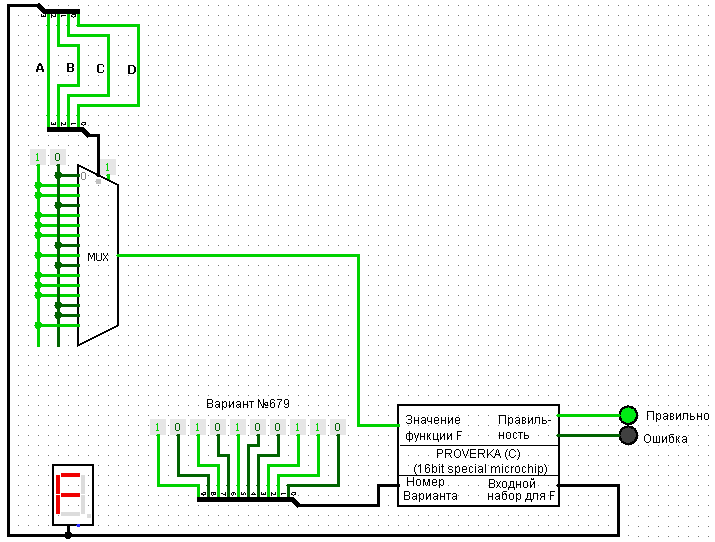
* “Ленивая” реализация логических функций, когда минимизацией можно пренебречь.
* В качестве коммутатора N к 1:
* Для преобразования параллельного кода в последовательный.
* Для поочердного подключения многих источников информации к одному потребителю.

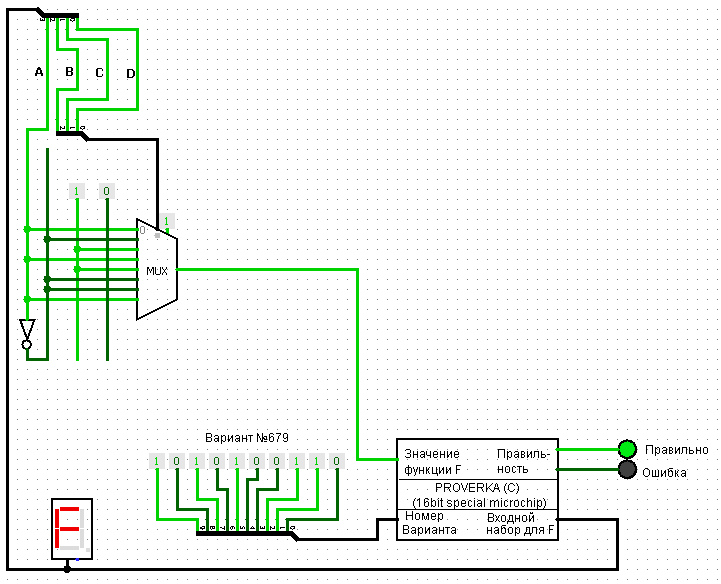
# Мультиплексоры: определение, УГО, области применения, реализация логических функций на мультиплексорах достаточной разрядности.

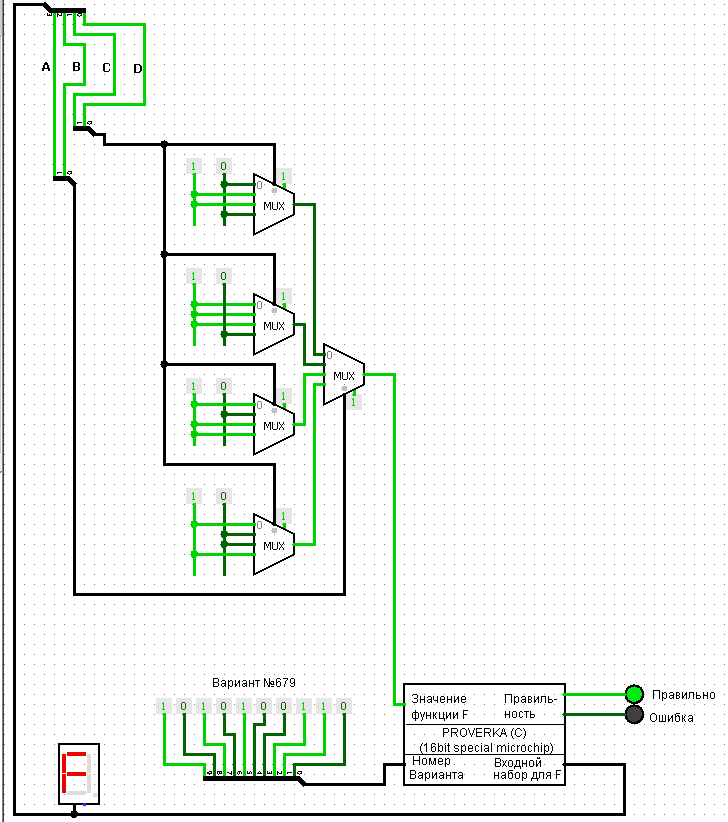
см. 41

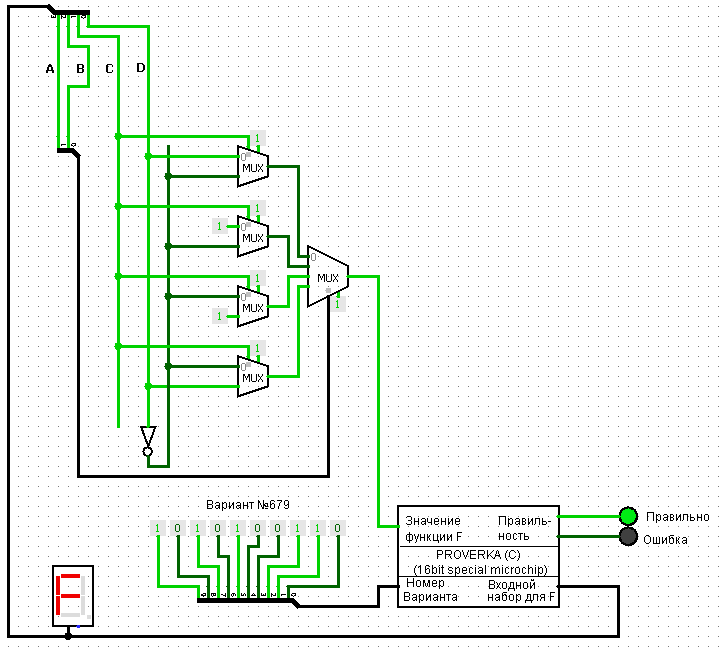
**Таблица истинности.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F(A,B,C,D)** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Реализация функции при помощи мультиплексора 16-1.**

**Реализация функции при помощи мультиплексора 8-1.**

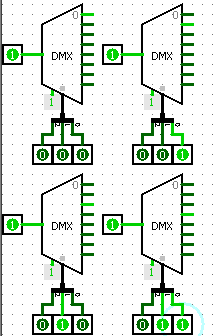
**Реализация функции при помощи мультиплексоров 4-1.**

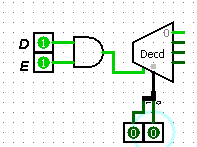
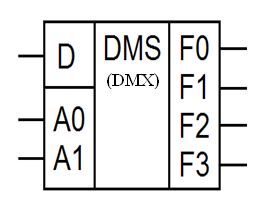
**Реализация функции при помощи мультиплексоров 2-1 и одного мультиплексора 4-1.**

# Мультиплексоры: определение, УГО, области применения, реализация логических функций на мультиплексорах меньшей разрядности, чем количество переменных.

см. 43.

# Демультиплексоры: определение, УГО, области применения, функциональная схема демультиплексора на примере демультиплексора 1-4.

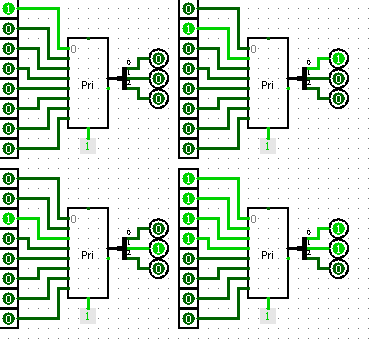
Схема обратная мультиплексору. Один информационный вход, N адресных входов, (разрешающий вход – опционально) и выходов. Реализуются через дешифратор.



Области применения:

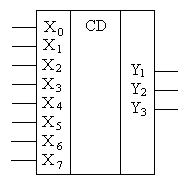
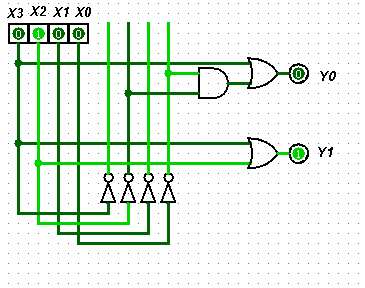
* коммутатор 1 к N
* реализация логических функций (т.е. реализация на дешифраторе)

# Шифраторы: определение, УГО, области применения, таблица истинности и функциональная схема простого полного шифратора 4-2.

Выполняет функцию обратную дешифратору. Имеет информационных входов, N выходов. На входе – унитарный код. 

Виды:

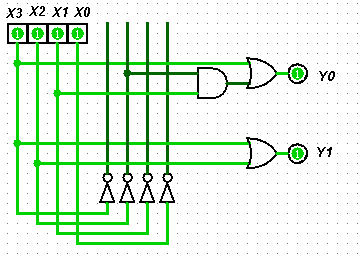
* Полные. Реализуются все выходные комбинации.
* Неполные. Реализуется часть выходных комбинаций.
* Простые. Допустимо иметь только один сигнал на всем множестве информационных входов.
* Приоритетные. Из множества информационных входов выбирается старший (по номеру).

Области применения:

* определение кодов нажатых клавиш
* в составе преобразователей кодов

# Шифраторы: определение, УГО, области применения, таблица истинности и функциональная схема приоритетного шифратора 4-2.

см. 45. 

# Сумматоры: определение, УГО, классификация, четвертьсумматор, полусумматор.

Сумматор – это цифровое устройство, выполняющее арифметическое сложение двух чисел в том или ином коде. Если используются специальные коды, то на сумматоре можно выполнять вычитание.

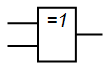
Классификация:

1. В зависимости от системы счисления:
   * Двоичные;
   * Двоично-десятичные;
   * десятичные;
   * и др.
2. По количеству одновременно обрабатываемых разрядов складваемых чисел:
   * одноразрядные;
   * многоразрядные.
3. По числу входов и выходов одноразрядные двоичные сумматоры делятся на:
   * четвертьсумматоры;
   * полусумматоры;
   * полные одноразрядные двоичные сумматоры.
4. По способу представления и обработки складываемых чисел многоразрядные сумматоры подразделяются на:
   * последовательные;
   * параллельные.
5. По возможности сохранения результата выделяют два типа сумматора:
   * комбинационный сумматор;
   * накапливающий сумматор.
6. По способу организации межразрядных связей.
   * сумматоры с последовательным переносом;
   * сумматоры с параллельным переносом;
   * сумматоры с условным переносом;
   * сумматоры со сквозным переносом.

Четверть сумматор:

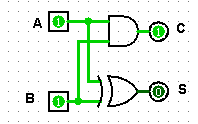
Простейший двоичный сумматор, имеет два входа для двух одноразрядных чисел и один выход, на котором формируется их сумма.

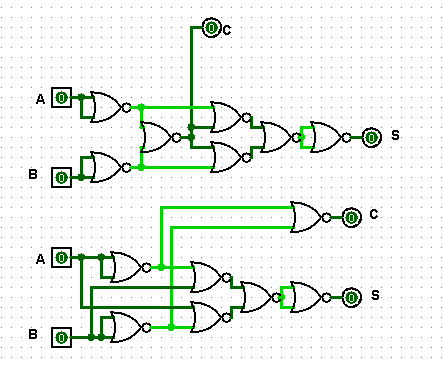
Реализуется логическим элементом «ИСКЛЮЧАЮЩЕЕ ИЛИ».

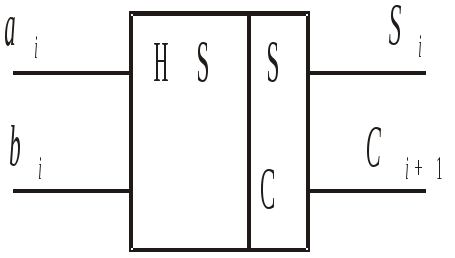


Полусумматоры – это комбинационна схема, которая вырабатывает сигналы суммы(S) и переноса(C) при сложении двух двочиных одноразрядных чисел A и B.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |







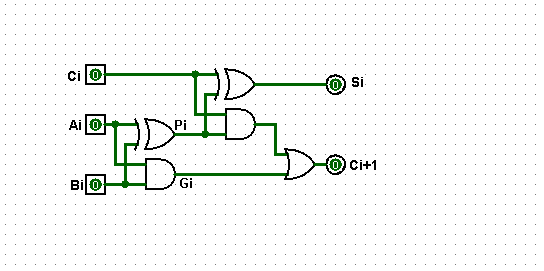
# Сумматоры: определение, УГО, полный одноразрядный сумматор, многоразрядный сумматор параллельного действия с последовательным переносом.

См. 47.

Полный сумматор в отличии от полусумматора должен воспринимать три входных сигнала: 2 одоразрядных слагаемых и сигнал переноса от предыдущего разряда.

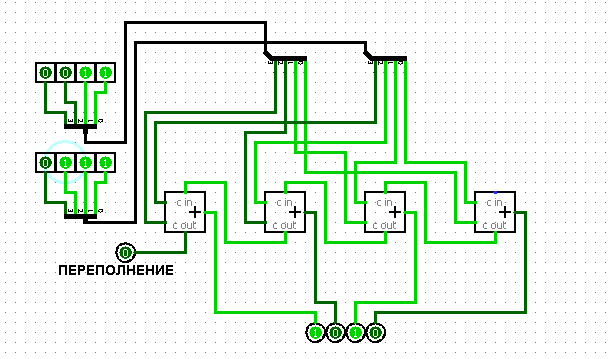
В качестве результат сумматор выдает сумму и перенос в следующий разряд.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ai | Bi | Ci | Si | Ci+1 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |



# Многоразрядный сумматор параллельного действия (аргументы подаются одновременно по всем разрядам) с последовательным переносом.

Для сложения двух многоразрядных чисел на каждый разряд необходим один полный сумматор.



# Сумматоры: определение, УГО, сумматор последовательного действия – принцип работы, обобщенная структурная схема, достоинства и недостатки. http://literaturki.net/images/csht-2/image331.gif

Сумматор последовательного действия состоит из состоит из одноразрядного сумматора, выход переноса которого соединен с его же входом переноса через элемент задержки (D-триггер). Опереция суммирования во всех разрядах слагаемых осуществляется с помощью одного и того же одноразрядного сумматора, но последовательно во времени, начиная с младших разрядов. Сумма накапливается постепенно. Обычно подобный сумматор используется в комбинации со сдвиговыми регистрами.

Достоинство – минимум оборудования.

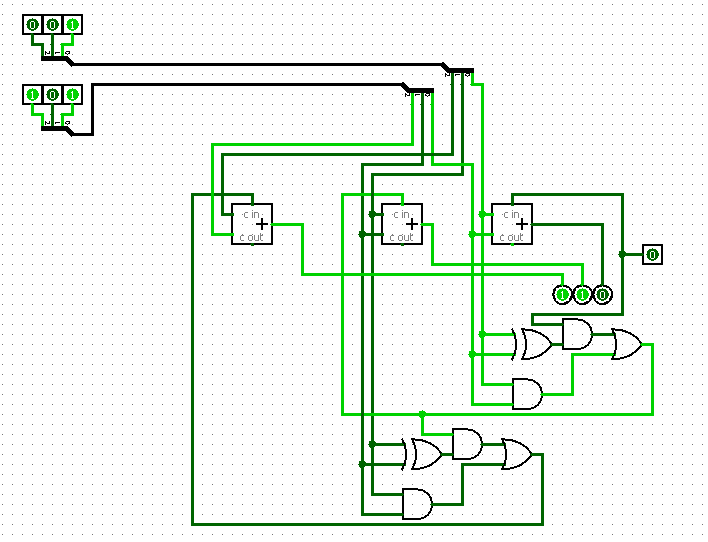
Недостаток – большое время работы.

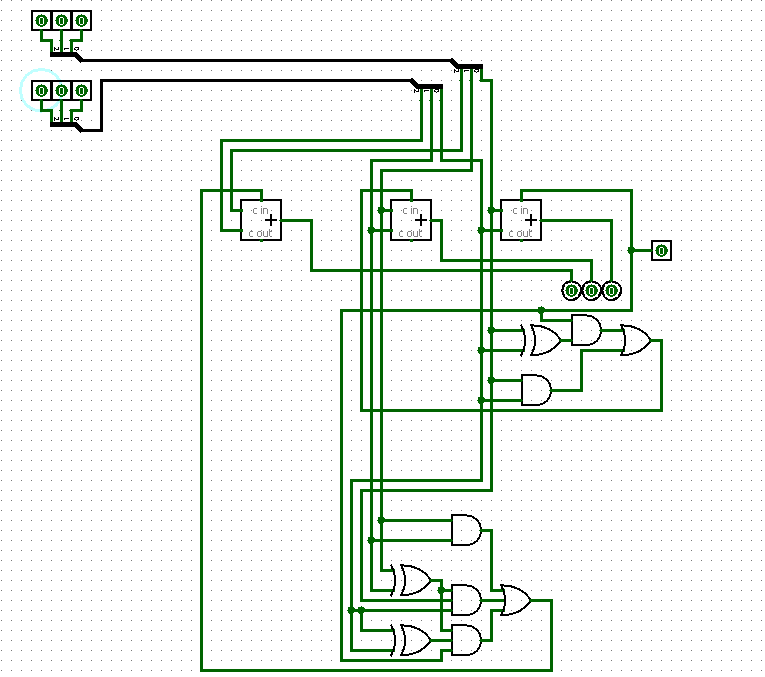
# Сумматоры: определение, УГО, сумматоры с параллельным переносом – рекуррентная формула для вычисления переносов, пример схемы для 2х разрядного сумматора, достоинства и недостатки.

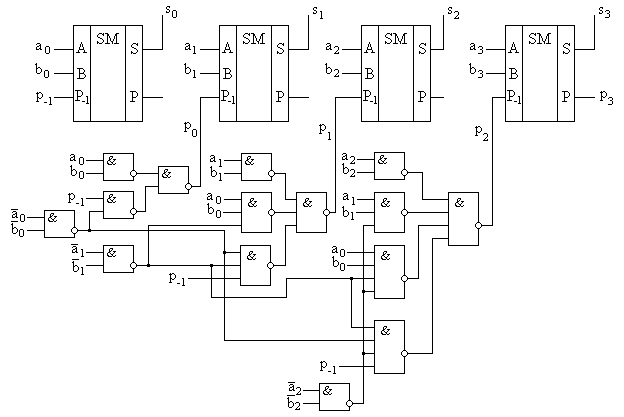
см. 49,50.

Чтобы уменьшить время операции сложения многоразрядных чисел, можно использовать схемы параллельного переноса. При этом все сигналы переноса вычисляются непосредственно по значениям входных переменных.

где

**

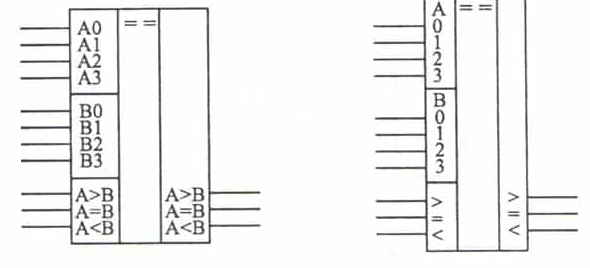




# Компараторы: определение, таблица истинности, выражения для вычисления отношений «равно» и «больше» на примере компаратора двухразрядных чисел.

**Компараторы** выполняют операцию определения отношения между числами. Основными отношениями можно считать «равно» и «больше». Все прочие отношения при желании можно выразить как комбинацию основных. Например, «меньше» это отрицание «больше» и отрицание «равно».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | F1(A>B) | F2(A=B) | F3(A<B) |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |



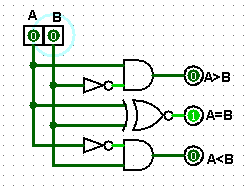


Таблица истинности одноразрядного компаратора с учетом входящих результатов сравнения старших разрядов старших разрядов.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ai+1=Bi+1 | Ai+1>Bi+1 | Ai+1<Bi+1 | Ai | Bi | A>B | A=B | A<B |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | \* | \* | 1 | 0 | 0 |
| 0 | 0 | 1 | \* | \* | 0 | 0 | 1 |

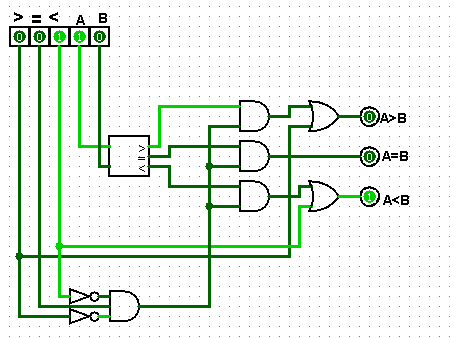
Признак равенства для одноразрядных чисел:

Признак равенства для многоразрядных чисел:

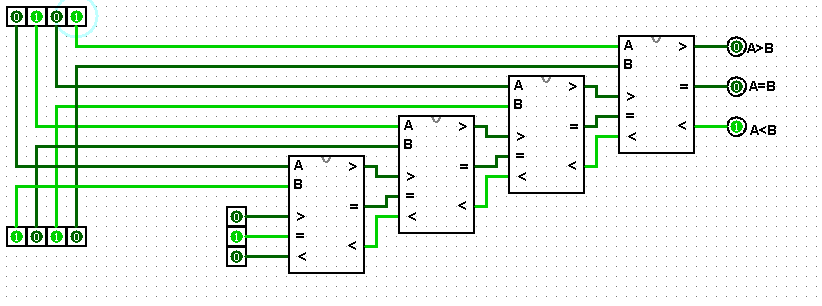
Признак «больше» для двухразрядных чисел:

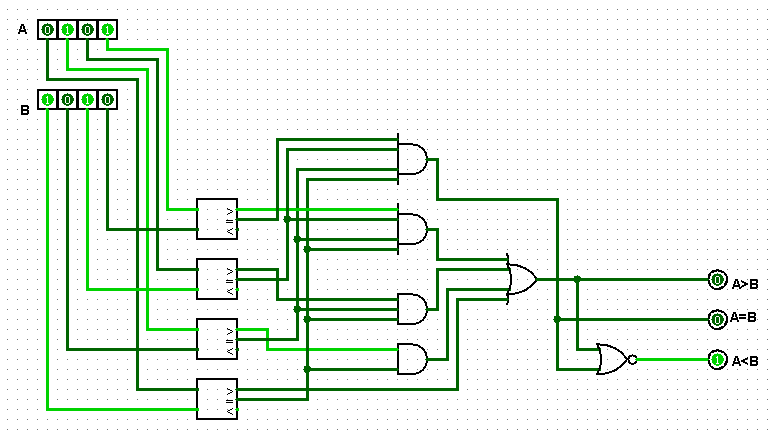
Признак больше для многоразрядных чисел:

Одноразрядный компаратор с входящими переносами от старшего разряда



Последовательное наращивание одноразрядных компараторов.



Параллельный четырехразрядный компаратор.

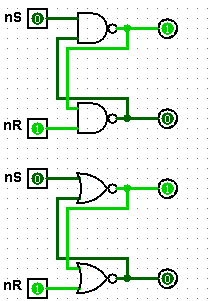
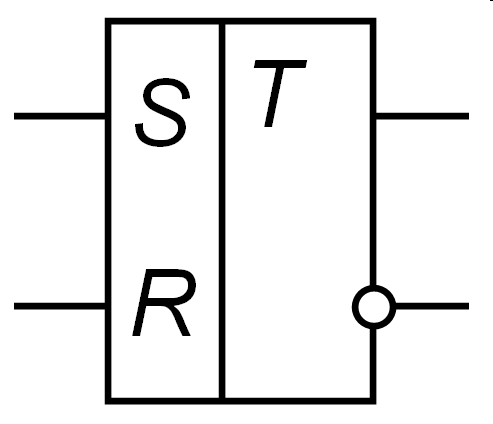
# Триггеры: определение, классификация, способы синхронизации, области применения. Сигнал синхронизации: назначение, структура.

Триггер – простейший цифровой автомат, которые представляет собой элементарную ячейку памяти и может хранить один бит информации. Тригер может находится в одном из двух состояний – ноль или единица. Эффект запоминания возникает благодаря наличию обратных связей между элементами. Триггеры могут быть использованы как самостоятельно, так и образовывать регистры и счетчики.

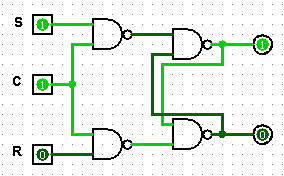
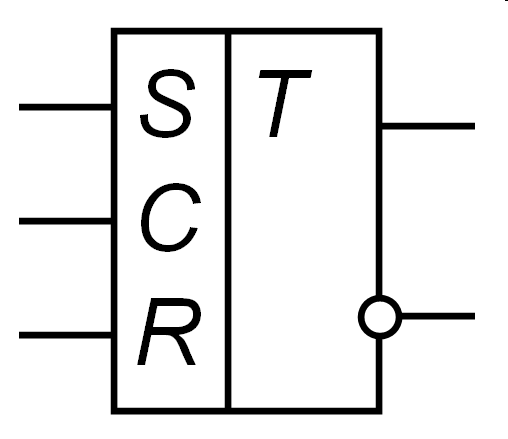
Классификация:

1. По логике функционирования: RS, JK, T, D.
2. По способу записи информации:
   1. асинхронные
   2. синхронные
      1. со статическим управлением (стробируемые)
      2. с динамическим управлением (тактируемые)
3. По кол-ву ступеней:
   1. одноступенчатые
   2. двуступенчатые

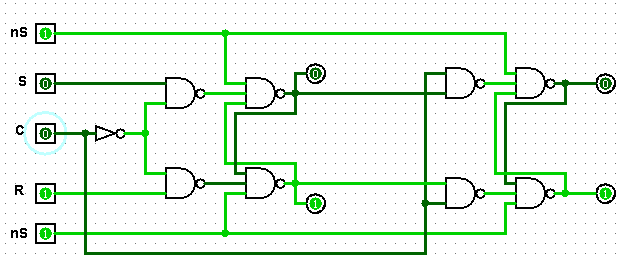
# Триггеры: асинхронный RS-триггер на элементах «И-НЕ» и на элементах «ИЛИ-НЕ», его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

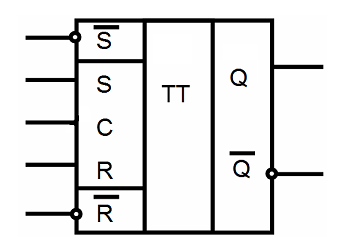
 

# Триггеры: синхронный RS-триггер на элементах «И-НЕ» со статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

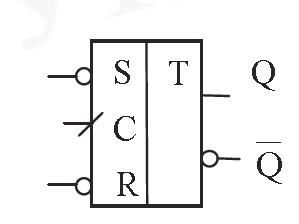
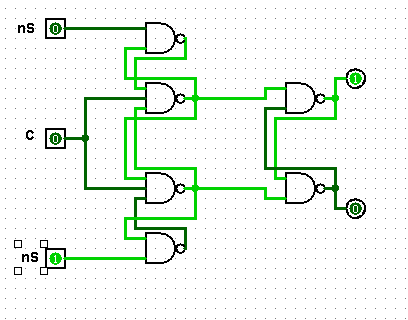
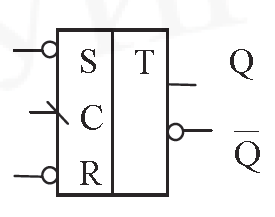
 

# Триггеры: синхронный двухступенчатый RS-триггер с асинхронными входами и статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

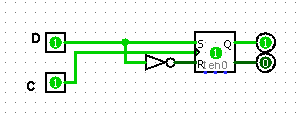
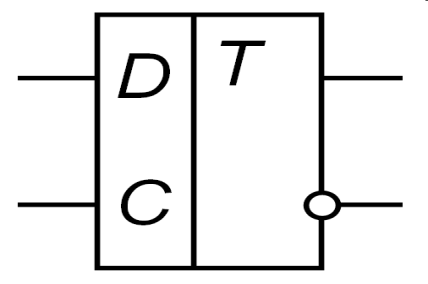




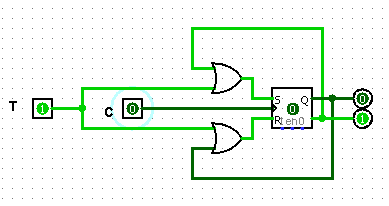
# Триггеры: синхронный RS-триггер с динамическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

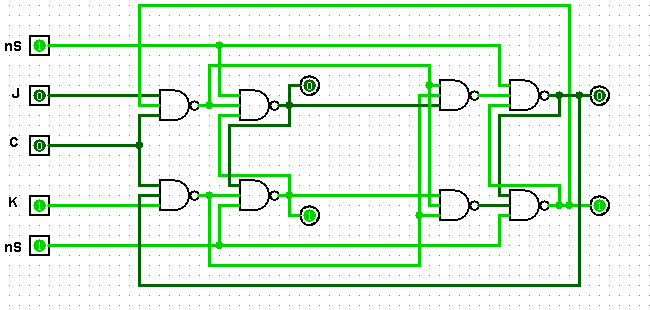
# Триггеры: синхронный D-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

# Триггеры: синхронный T-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.



# Триггеры: синхронный двухступенчатый JK-триггер с асинхронными входами предустановки, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.



# Регистры: определение, выполняемые функции, классификация, виды сдвига.

Регистр – цифровой автомат, основным назначение которого являются

* запись (прием).
* временное хранение
* выдача

группы двоичных разрядов (байта, слова)

Этот минимум операций выполняет простейший регистр – регист хранения.

При этом запись и выдача информации осуществляется на все разряды одновременно (т.е. в параллельном коде.

Усовершенствованный регистр (сдвиговый) может выполнять операцию сдвига информации вправо и влево, что эквивалентно делению и умножению двоичного числа на 2.

Регистры строятся на основе триггеров.

Благодаря операции сдвига, становятся возможными разные способы приема и выдачи информации:

* параллельный прием;
* параллельная выдача;
* последовательный прием;
* последовательная выдача.

Получается, что сдвиговый регистр это устройство, которое может:

* преобразовывать последовательный код в параллельный и наоборот;
* быстро выполнять операции умножения и деления на 2.

Классификация:

* по количеству разрядов
* по виду триггеров, на которых реализованы
* способу приема и выдачи данных
  + параллельные (простейшие регистры хранения, на синхронных триггерах)
  + последовательные (сдвиговые) (последовательный прием, последовательная выдача). запись и считывание информации происходит в первый триггер, а та информация, которая была в этом триггере, перезаписывается в следующий — то же самое происходит и с остальными триггерами.
  + параллельно-последовательные (универсальный сдвиговый регистр).

Сдвиговые регистры.

Триггеры сдвигового регистра связаны между собой цепями переноса, что позволяет одновременно переносить содержимое отдельных триггеров регистра в соседние триггеры, осуществляя операцию сдвига.

Сдвиг характеризуется:

* направлением
* типом:
  + логический:

Освобождающиеся биты заполняются нулями как при сдвиге вправо, так и при сдвиге влево.

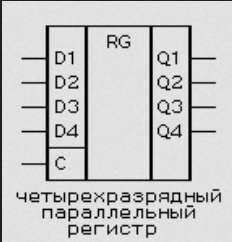
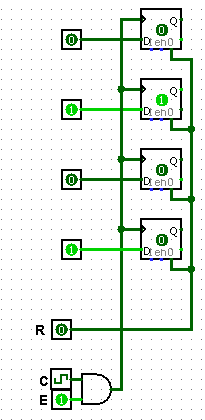
* + арифметический:

Содержимое регистра понимается как число в дополнительном коде. При сдвиге влево справа появится 0, при сдвиге вправо слева дублируется предыдущее значение.

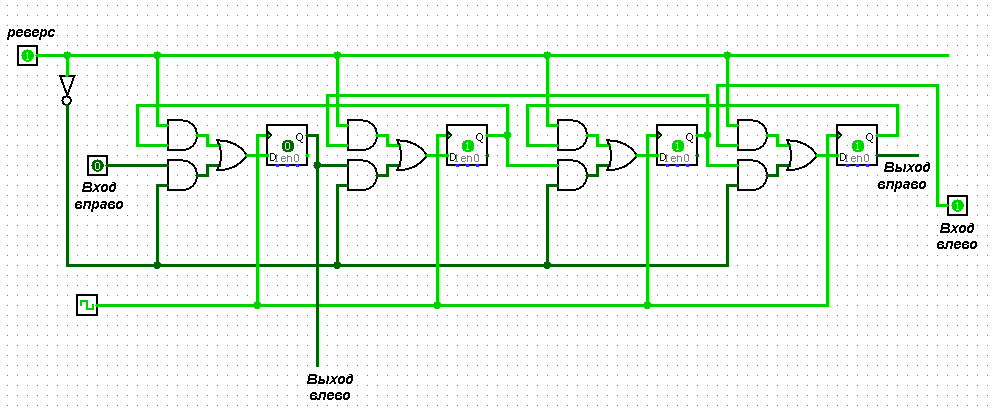
* + циклический:

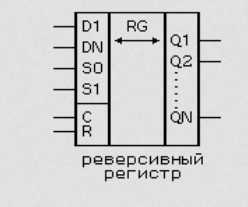
Вытесняемое значение заносится в свободный триггер на другом конце.

# Четырехразрядный параллельный регистр на D-триггерах: УГО, внутреннее устройство, выполняемые функции.

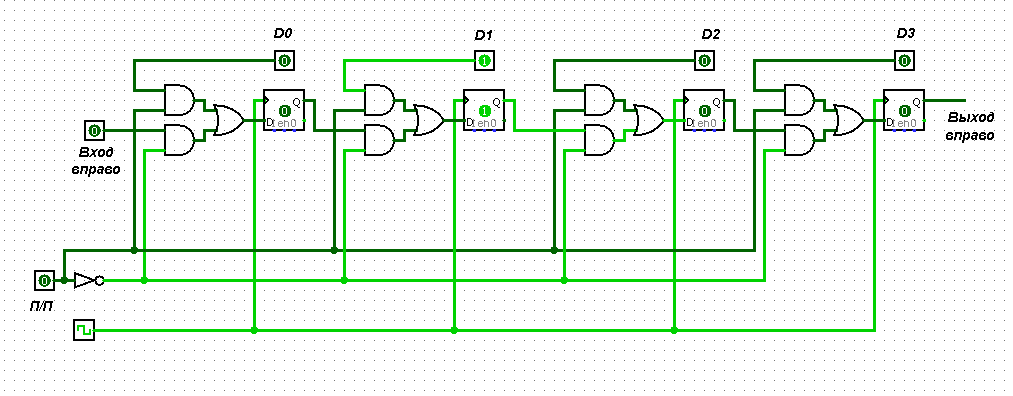
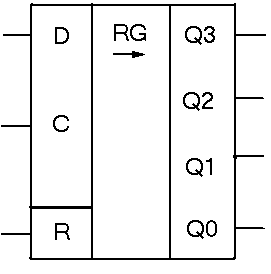


# Четырехразрядный регистр с последовательными приемом и выдачей на D-триггерах, с выбором направления сдвига: УГО, внутреннее устройство, варианты использования.

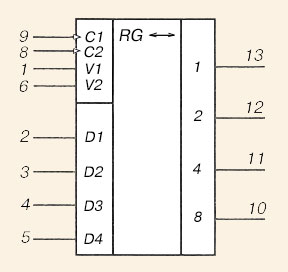




# Четырехразрядный регистр с параллельно-последовательным приемом и выдачей, реализующий сдвиг вправо: УГО, внутреннее устройство, варианты использования.

# Универсальный сдвиговый регистр: УГО, внутреннее устройство (на примере одного разряда), варианты использования.



# Счетчики: определение, основные параметры, классификация.

Счетчиком называется цифровой автомат, предназначенный для счета входных импульсов, хранения их кол-ва в виде двоичного числа.

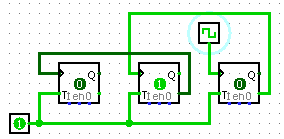
Параметры счетчика

* Модуль счета M ( – максимальное значение счетчика)
* Шаг счета
* Направление счета

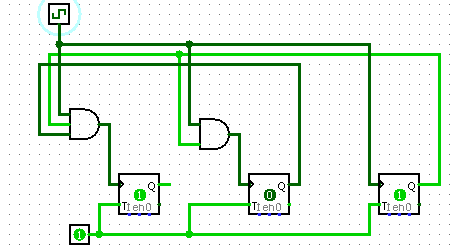
Классификация:

* По модулю счета:
  + Двоичные
  + Двоично-десятичные
  + С произвольным постоянным модулем счета
  + С переменным модулем счета
* По шагу счета:
  + 1
  + 2
  + …
  + K : K<M
* По направлению счета:
  + Суммирующие
  + Вычитающие
  + Реверсивные
* По способу организации межразрядных связей:
  + Счетчики с последовательным переносом (асинхронные), в которых переключение триггеров разрядных схем осуществляется последовательно один за другим.
  + Счетчики с параллельным переносом (синхронные), в ктороыз переключение всех триггеров разрядных схем осуществляется одновременно по сигналу синхронизации.
  + Счетчики с комбинированным последовательно-параллельным переносом.

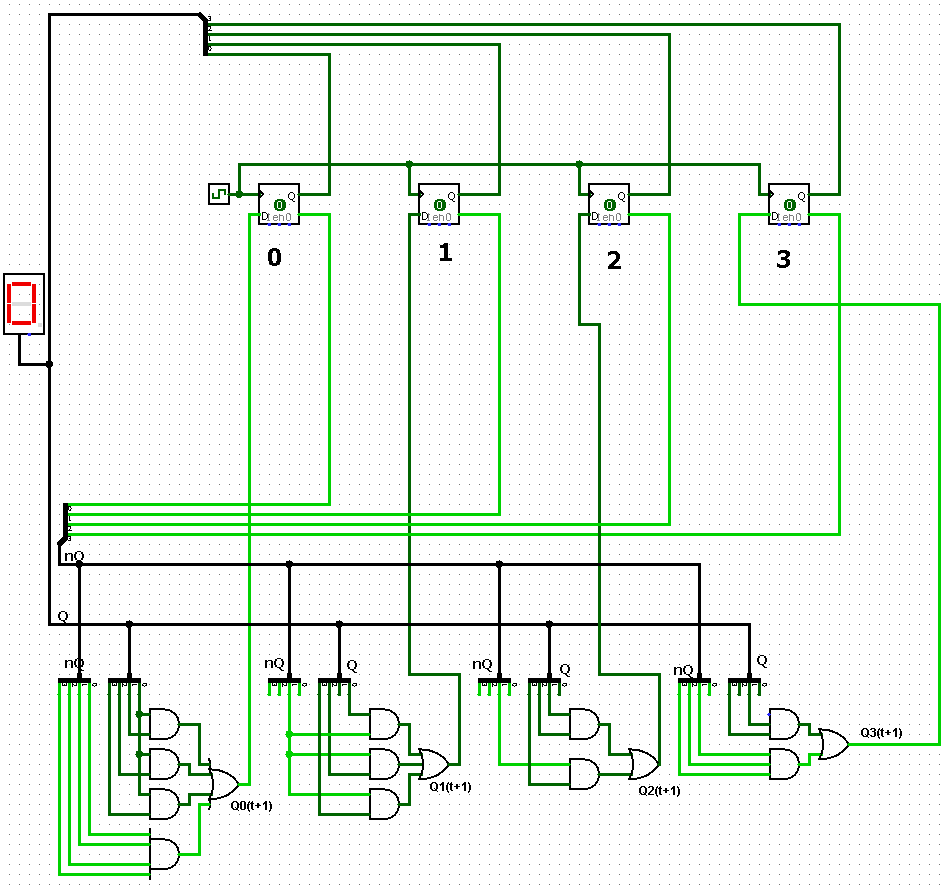
# Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с последовательным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки.



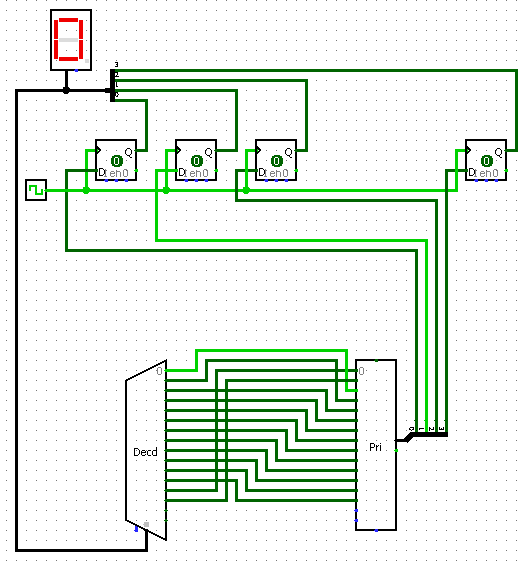
# Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с ускоренным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки.



# Синтез оптимальных счетчиков с требуемым модулем, шагом и направлением на D-триггерах.



# Быстрый синтез счетчиков с требуемым модулем, шагом и направлением на D-триггерах.



# Основы алгоритмизации. Понятие алгоритма, свойства алгоритмов.

Алгоритм — это последовательность действий, которые необходимо выполнить для решения определенной задачи или получения результата.

Алгоритм обладает следующими свойствами:

* Дискретность (прерывность, раздельность) – процесс решения задачи представляется как последовательное выполнение простых (или ранее определенных) шагов. Каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего.
* Определенность – каждое действие алгоритма должно быть четким, однозначным и не оставлять места для множества возможных смыслов.
* Результативность (конечность) – алгоритм должен приводить к решению задачи за конечное число шагов.
* Массовость – алгоритм решения задачи разрабатывается в общем виде, то есть, он должен быть применим для некоторого класса задач, различающихся только исходными данными.

Классификация алгоритмов по структуре

* Линейный алгоритм – набор команд (указаний), выполняемых последовательно во времени друг за другом, безальтернативно, без повторений уже реализованных ранее команд.
* Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ выполняет переход на одну из нескольких возможных команд.
* Циклический алгоритм – алгоритм, предусматривающий многократное повторение одного и того же фрагмента (одних и тех же операций) над новыми исходными данными.

Цикл программы – последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

# Основы алгоритмизации. Понятие алгоритма, правила построения блок-схем.

см. 70.

**Изображение алгоритмов с помощью блок-схем**

| Наименование | Обозначение | Функция |
| --- | --- | --- |
| Начало-конец  (пуск-остановка) |  | Обозначает начало и конец программы или подпрограммы |
| Действие |  | Выполнение одной или нескольких операций |
| Ввод/вывод |  | Операции ввода или вывода даных |
| Предопределенный процесс |  | Вызов подпрограммы |
| Условие |  | Проверка значение выражение и разветвление алгоритма |
| Множественные условия | Или | Проверка значение выражение и разветвление алгоритма на более, чем три ветви. |
| Границы цикла с  предусловием или  постусловием |  | Обозначает начало и конец цикла, где указываются условия цикла. |
| Цикл со счетчиком |  | Описывает начальное значение переменной, ее приращение и конечное значение |
| Соединитель |  | Нужен для соединения линий алгоритма при их разрыве или переходе на другую страницу |
| Комментарий |  | Применятся для описания набора блоков |

# Основы алгоритмизации. Алгоритм поиска максимума и минимума.

Обобщенный алгоритм:

ЕСЛИ (n>0) ТО

min=A[0]

i=1

ПОКА (i<n)

ЕСЛИ (min>A[i]) ТО

min=A[i]

i=i+1

# Основы алгоритмизации. Принцип структурного программирования Дейкстры.

Язык структурного программирования разработан в 1970 годах Эдсгером Вибе Дейкстрой.

Основная идея – отказ от оператора перехода (Goto) в языках программирования высокого уровня, так как они могут создавать проблемы с чтением программы и с её выполнением (не выполняется очистка памяти в точке «ухода» и инициализация переменных в точке «прихода»).

Язык включает 3 основные **операции**:

1. Действие

2. Условие — ЕСЛИ (условие) ТО

ИНАЧЕ

3. Цикл — ПОКА (условие)

перечень действий

При записи алгоритма содержимое условий и циклов должно записываться с отступом, образуя структурированный уровневый вид:

действия

ЕСЛИ (условие) ТО

действия

ПОКА (условие)

действия

действия

ИНАЧЕ

действия

**Принципы структурного программирования:**

1. Следует отказаться от использования оператора безусловного перехода.

2. Любая программа строится на основе трех базовых конструкций: действие, условие, цикл.

3. Базовые конструкции могут быть вложены друг друга в произвольной форме и неограниченном количестве.

4. Повторяющиеся фрагменты желательно оформлять в виде подпрограмм.

5. Логически законченные группы инструкций желательно объединять в блоки.

6. Все перечисленные конструкции должны иметь один вход и один выход.

7. Разработка программы ведется пошагово, используя метод «сверху-вниз».

**Метод «сверху-вниз».**

При разработке алгоритма или программы, первоначально реализуется основной управляющий алгоритм без детализации функциональных элементов, которые заменяются функциями-заглушками, возвращающими постоянное значение.

По мере разработки каждая функция-заглушка реализуется своим алгоритмом, вызывающим, при необходимости, другие функции-заглушки.

Разработка заканчивается когда все функции-заглушки реализованы в виде набора базовых конструкций.

**Метод «снизу-вверх».**

Предполагает обратный путь разработки: Сначала реализуются элементарные функции из которых собираются более сложные конструкции. В настоящее время активно используется, так как существует большое количество библиотек, содержащих готовые функции.

# Основы алгоритмизации. Алгоритм сортировки «Пузырек».

i=0

ПОКА (i<n)

j=i+1

ПОКА (j<n)

ЕСЛИ (А[i]>A[j]) ТО //сортируем по возрастанию

врем=А[i]

А[i]=A[j]

A[j]=врем

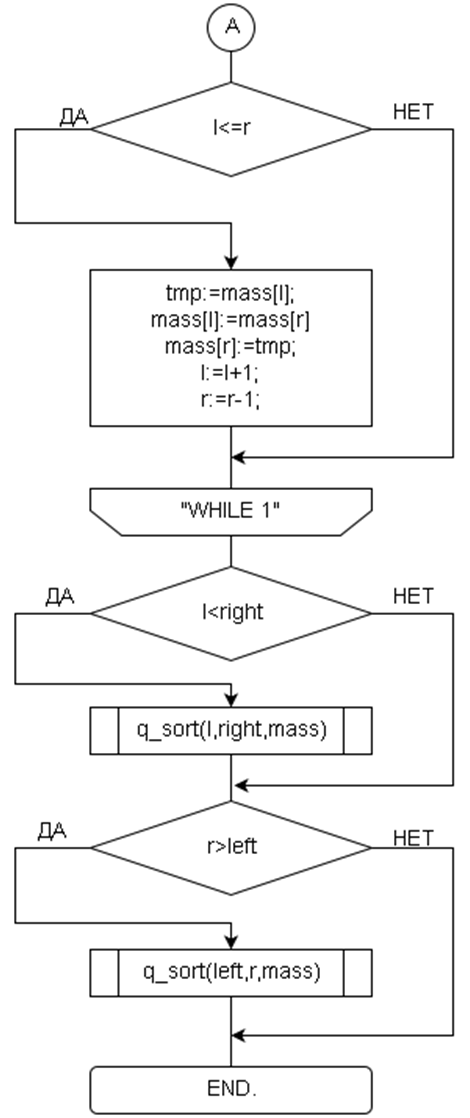
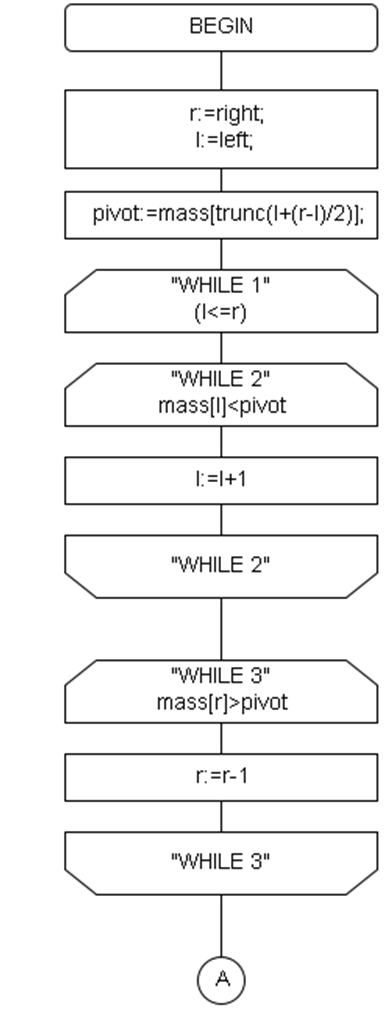
j=j+1

i=i+1

# Основы алгоритмизации. Алгоритм быстрой сортировки Хоара.

procedure q\_sort (left, right: integer; var mass: array of integer);

Быстрая сортировка



# Основы алгоритмизации. Рекурсия – назначение, виды, примеры организации.

Рекурсией называется вызов функцией самой себя с некоторым изменением входных параметров.

Пример рекурсии – вычисление факториала:

6!=6\*5!

5!=5\*4!

.....

1!=1

Главным условием для использования рекурсии в функциях является наличие базы рекурсии – то есть значения, которое зависит только от входного параметра и позволяет выйти из рекурсии.

В примере база – это факториал 1.

Алгоритмы в духе «У попа была собака...» использовать нельзя!

Еще одним ограничивающим фактором применения простой рекурсии является объем специальной области памяти, где хранится контекст и адреса возврата при вызове функций. При большой глубине рекурсии эта память может быть переполнена, что приведет к ошибке выполнения программы.

Иными словами 1000000000! теоретически вычислимая рекурсия, так как имеет базу, но глубина рекурсии может превысить допустимые ограничения, что приведет к невозможности получения результата.

Тем не менее, рекурсия может быть бесконечной. Это так называемая **хвостовая рекурсия.** Она поддерживается в некоторых языках программирования. Для ее работы нужно чтобы рекурсивный вызов был всегда последним в теле рекурсивной функции, и чтобы компилятор или интерпретатор языка умели обнаруживать подобный вызов. В таком случае нет нужды запоминать адрес возврата и память не переполняется.

**Хвостовая рекурсия** любимый прием функциональных языков программирования (Haskell, Erlang и др.), в основе которых лежит понимание любой программы как математической функции.

# Основы алгоритмизации. Проверка вводимых данных – типичные ошибки и методы борьбы с ними.

1. Операции с данными.

Выражение В=А+Б потенциально не содержит проблем, а выражение В=А/Б может привести к появлению ошибки.

А будет ли возможна ошибка если написать:

ЕСЛИ (Б!=0) ТО

В=А/Б

Кроме того, что А может быть 0 и второе выражение станет вычислить невозможно, ошибка может возникнуть и при А=1 и Б=3, так как для целого В значение 0,(3) является «машинным нулем».

Аналогичные ограничения есть у четных корней, прямых и обратных тригонометрических функций.

**1. Операции с данными.**

Все операнды функций, имеющих ограничения на область допустимых значений (ОДЗ), должны в обязательном порядке проверяться на соответствие ОДЗ.

**2. Операции с массивами и памятью.**

Память данных и команд в современных ЭВМ в большинстве случаем не разделена и слабо защищена от попадания в чужую область памяти.

Так, объявив массив на 5 элементов можно попытаться обратиться к 6, 10, -5 элементам попав в чужую область памяти и исказив как данные, так и программные коды.

Аналогичные ошибки бывают при копировании данных:

Запрос пользователю: Введите имя

Скопировать в строку введенное имя

При этом размер копируемой области памяти в большинстве случаев определяется длинной введенной строки, без учета размера строки получателя, что приведет к переполнению, если пользователь ввел очень большое имя.

При любых операциях с массивами и памятью необходимо контролировать выход индексных переменных за границы.

При копировании блоков памяти необходимо проверять как размер копируемого блока, так и размер области получателя выбирая минимальное из них или отказываясь от копирования при превышении допустимого объема.

**3. Знаковые и беззнаковые переменные.**

Как уже упоминалось знаковые и беззнаковые переменные это разная интерпретация одних и тех же значений разрядной сетки числа.

Например:

ЕСЛИ n<10

i=0

ПОКА (i<n)

.....

i=i+1

Этот безобидный код ограничивает количество выполняемых циклов 10.

Но если мы введем n=-1 и при этом n будет беззнаковым или беззнаковым будет i, то при однобайтовом n станет 255, при двухбайтовом – 65535, при четырехбайтовом > 4 млрд.

Нельзя использовать в одном выражении знаковые и беззнаковые переменные, так как это может привести к непредсказуемой их интерпретации и ошибкам, как следствие того, что знаковое число воспринимается как большое беззнаковое и наоборот.

Перед вычислениями знаковые и беззнаковые числа должны быть принудительно приведены к одной форме.