МОДУЛЬ 3

РАЗДЕЛ 4. Логические основы информатики.

ТЕМА 7. Логические основы компьютера.

Виды электронных вычислительных машин

Электронные вычислительные машины (ЭВМ) представляют собой устройство, предназначенное для выполнения вычислительных операции по заданной программе. Современная электронная вычислительная машина – это сложнейший комплекс устройств, восхищающий своим технологическим совершенством и разнообразием физических принципов работы.

Вычислительные машины в зависимости от способа представления информации подразделяются на две большие группы: вычислительные машины непрерывного действия, или аналоговые вычислительные машины (АВМ), и вычислительные машины дискретного действия, или цифровые вычислительные машины (ЦВМ).

В АВМ входные, выходные и промежуточные величины представляются в виде токов или напряжений, значения которых в определенном масштабе соответствуют числом. Математические действия над числами заменяются в АВМ различными преобразованиями электрических токов или напряжений.

Подлинный прогресс науки, называемой математической логикой, был достигнут в середине XIX в. Прежде всего благодаря труду английского логика Джорджа Буля «Математический анализ логики». Он перенес на логику законы и правила алгебраических действий, ввел логические операции, предложил способ записи высказываний в символической форме.

Современная математизированная формальная логика представляет собой обширную научную область и находит широкое применение как внутри математики (исследование оснований математики), так и вне ее (анализ и синтез автоматических устройств, теоретическая кибернетика, в частности, искусственный интеллект).

Формы мышления.

Первые учения о формах и способах рассуждений возникли в странах Древнего Востока (Китай, Индия), – это были философские рассуждения. Но в основе современной логики лежат учения, созданные древнегреческими мыслителями. Основы формальной логики заложил Аристотель, который впервые отделил логические формы мышления (речи) от его содержания.

Законы логики отражают в сознании человека свойства, связи и отношения объектов окружающего мира. Логика позволяет строить формальные модели окружающего мира, отвлекаясь от содержательной

стороны. Мышление всегда осуществляется в каких-то формах. Основными формами мышления являются *понятие, высказывание* и *умозаключение.*

Понятие выделяет существенные признаки объекта, которые отличают его от других объектов. Объекты, объединенные понятием, образуют некоторое множество. Например, понятие «компьютер» объединяет множество электронных устройств, которые предназначены для обработки информации и обладают монитором и клавиатурой. Даже по этому короткому описанию компьютер трудно спутать с другими объектами, например с механизмами, служащими для перемещения по дорогам и хранящимися в гаражах, которые объединяются понятием «автомобиль».

Понятие *– это форма мышления, фиксирующая основные, существенные признаки объекта.* Понятие имеет две стороны: содержание и объем. Содержания понятия составляет совокупность существенных признаков объекта. Чтобы раскрыть содержание понятия, следует найти признаки, необходимые и достаточные для выделения данного объекта из множества других объектов. Свое понимание окружающего мира человек формулирует в форме высказываний (суждений, утверждений).

Высказывание строится на основе понятий и по форме является повествовательным предложением. Высказывание может быть ложным или истинным. *Истинным* будет высказывание, в котором связь понятий правильно отражает свойства и отношение реальных вещей. *Ложным* высказывание будет в том случае, когда оно не соответствует реальной действительности.

Высказывание *– это форма мышления, в которой что-либо утверждается или отрицается о свойствах реальных предметов и отношениях между ними. Высказывание может быть либо ложно, либо истинно.*

Умозаключение. Умозаключения позволяют на основе известных фактов, выраженных в форме суждений (высказываний), получать заключение, то есть новое знание. Примером могут быть геометрические доказательства.

Умозаключение *– это форма мышления, с помощью которой из одного или нескольких суждений (посылок) может быть получено новое суждение (заключение).*

Основные логические операции и правила их выполнения.

Среди задач, для решения которых привлекается ЭВМ, немало таких, которые по традиции принято называть логическими. Кто не знает шуточной задачи о перевозке волка, козы и капусты с одного берега на другой. В логических задачах исходными данными являются не только числа, но и неожиданные, подчас весьма запутанные суждения.

Появлению и развитию математической логики способствовало стремление найти строгие правила обоснования и доказательства новых положений в науке. По мнению некоторых ее создателей, математическая

логика должна была стать математическим аппаратом той части обычной логики, которую принято называть формальной.

Как и математика, формальная логика следует строгим правилам и не вникает в сущность анализируемых суждений. Одна из ее задач – установление формальных правил получения новых суждений из исходных, истинность которых не подвергается сомнению.

Алгебра логики – это раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними.

Логическое высказывание – это любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Алгебра логики рассматривает любое высказывание только с одной точки зрения – является ли оно истинным или ложным. Заметим, что зачастую трудно установить истинность высказывания. Так, например, высказывание "площадь поверхности Индийского океана равна 75 млн кв. км" в одной ситуации можно посчитать ложным, а в другой – истинным. Ложным – так как указанное значение неточное и вообще не является постоянным. Истинным – если рассматривать его как некоторое приближение, приемлемое на практике.

Употребляемые в обычной речи слова и словосочетания "**не**", "**и**", "**или**", "**если... , то**", "**тогда и только тогда**" и другие позволяют из уже заданных высказываний строить новые высказывания. Такие слова и словосочетания называются логическими операциями.

Высказывания, образованные из других высказываний с помощью логических связок, называются *составными*. Высказывания, не являющиеся составными, называются *простыми* (элементарными).

Так, например, из элементарных высказываний "Петров – врач", "Петров – шахматист" при помощи логической операции "и" можно получить составное высказывание, "Петров – врач и шахматист", понимаемое как "Петров – врач, хорошо играющий в шахматы".

При помощи логической операции "или" из этих же высказываний можно получить составное высказывание "Петров – врач или шахматист", понимаемое в алгебре логики как "Петров или врач, или шахматист, или и врач и шахматист одновременно".

Истинность или ложность получаемых таким образом составных высказываний зависит от истинности или ложности элементарных высказываний.

Чтобы обращаться к логическим высказываниям, им назначают имена. Пусть через **A** обозначено высказывание "Тимур поедет летом на море", а через **B** – высказывание "Тимур летом отправится в горы". Тогда составное высказывание "Тимур летом побывает и на море, и в горах" можно кратко записать как **A** и **B**. Здесь "и" – логическая операция, **A**, **B** – логические переменные, которые могут принимать только одно из двух значений – "истина" или "ложь", обозначаемые, соответственно, "1" или "0".

Каждая логическая связка рассматривается как операция над логическими высказываниями и имеет свое название и обозначение:

В алгебре высказываний над простыми высказываниями определены следующие логические операции:

- "**или**" – логическое сложение (дизъюнкция);
- "**и**" – логическое умножение (конъюнкция);
- "**не**" – логическое отрицание.
- "**если... , то**" – логическое следование (импликация);
- "**тогда и только тогда**" – эквивалентность;

Рассмотрим эти логические операции.

Логическое сложение. Это сложное суждение, которое может быть составлено из нескольких простых высказываний. Своим названием обязано латинскому *disjunctio* – разоблачение, различие. Значение этого сложного высказывания будет истина тогда и только тогда, когда истинно хотя бы одно из входящих в него суждений. Пусть число входящих в него высказываний будет минимальным и равно двум. Обозначим эти высказывания A и B. Тогда все возможные наборы их значений и итогового сложного высказывания Y можно изобразить с помощью таблицы, которая называется таблицей истинности.

Таблица истинности

A	B	Y = A + B
истинно	истинно	истинно
истинно	ложно	истинно
ложно	истинно	истинно
ложно	ложно	ложно

или

A	B	Y = A + B
1	1	1
1	0	1
0	1	1
0	0	0

Для обозначения дизъюнкции используется знак "+" или "V" и записывается

Y=A+B или Y = A V B

Это логическое сложение, поэтому:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

Логическое умножение. Это также сложное высказывание, которое может быть составлено из нескольких простых высказываний. Своим названием обязано латинскому *conjunctio* – соединяю. По определению *конъюнктивным* называется сложное высказывание, которое истинно тогда и

только тогда, когда истинны все входящие в него высказывания. Как и в предыдущем случае рассмотрим пример с двумя высказываниями A и B. Тогда все возможные наборы их значений и итогового сложного высказывания Y можно изобразить с помощью таблицы, которая называется таблицей истинности.

Таблица истинности

A	B	Y = A * B
истинно	истинно	истинно
истинно	ложно	ложно
ложно	истинно	ложно
ложно	ложно	ложно

или

A	B	Y = A * B
1	1	1
1	0	0
0	1	0
0	0	0

Для обозначения конъюнкции используется знак "*" и записывается

Y=A*B

Это логическое умножение, поэтому:

1 * 1 = 1
1 * 0 = 0
0 * 1 = 0
0 * 0 = 0

Логическое отрицание. Присоединение частицы "НЕ" к сказуемому данного простого высказывания A называется операцией логического отрицания. Операция логическое отрицание над высказыванием A записывается $\overline{A}$. Возможные наборы значений и итогового высказывания Y можно изобразить с помощью таблицы, которая называется таблицей истинности.

Таблица истинности

A	$Y = \overline{A}$
истинно	ложно
ложно	истинно

или

A	$Y = \overline{A}$
1	0
0	1

Логическое следование. Соединение двух высказываний в одно с использованием оборота речи "Если …, то…" называется операцией логического следования или импликацией. Операция импликации над высказываниями A и B записывается A → B (читается "A имплицирует B" или "B следует из A"). По определению *импликацией* называется сложное высказывание, которое ложно тогда и только тогда, когда A – истинно, а B – ложно. Рассмотрим пример с двумя высказываниями A и B. Тогда все возможные наборы их значений и итогового сложного высказывания Y можно изобразить с помощью таблицы, которая называется таблицей истинности.

Таблица истинности

A	B	Y=A → B
истинно	истинно	истинно
истинно	ложно	ложно
ложно	истинно	истинно
ложно	ложно	истинно

или

A	B	Y=A → B
1	1	1
1	0	0
0	1	1
0	0	1

Эквивалентность. Соединение двух простых высказываний A и B в одно с использованием оборота речи, или, как принято говорить, логической операции "…тогда и только тогда…", называется операцией эквивалентности. Многоточием помечены высказывания, над которыми проводится операция эквивалентности. Операция эквивалентности над высказываниями A и B записывается A~B (читается "A эквивалентно B"). *Эквивалентностью* называется сложное высказывание, которое истинно тогда и только тогда, когда A и B – одновременно истинны или ложны. Рассмотрим пример с двумя высказываниями A и B. Тогда все возможные наборы их значений и итогового сложного высказывания Y можно изобразить с помощью таблицы, которая называется таблицей истинности.

Таблица истинности

A	B	Y=A ~ B
истинно	истинно	истинно
истинно	ложно	ложно
ложно	истинно	ложно
ложно	ложно	истинно

или

A	B	Y=A ~ B
1	1	1
1	0	0
0	1	0
0	0	1

Три логические операции: дизъюнкция, конъюнкция и отрицание составляют "полную систему" – это значит, что любое логическое выражение можно составить с использованием только этих трех операций. Например,

операцию импликации A→B можно заменить на 1-A+A*B, а операцию эквивалентность A~B заменить на $1-(A-B)^2$.

ОСНОВНЫЕ ЗАКОНЫ АЛГЕБРЫ ЛОГИКИ

Закон	Для ИЛИ	Для И
Переместительный	$x \vee y = y \vee x$	$x \cdot y = y \cdot x$
Сочетательный	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Распределительный	$x \cdot (y \vee z) = x \cdot y \vee x \cdot z$	$x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z)$
Правила де Моргана	$\overline{x \vee y} = \bar{y} \cdot \bar{x}$	$\overline{x \cdot y} = \bar{y} \vee \bar{x}$
Идемпотенции	$x \vee x = x$	$x \cdot x = x$
Поглощения	$x \vee (x \cdot y) = x$	$x \cdot (x \vee y) = x$
Склеивания	$(x \cdot y) \vee (\bar{x} \cdot y) = y$	$(x \vee y) \cdot (\bar{x} \vee y) = y$
Операция переменной с ее инверсией	$x \vee \bar{x} = 1$	$x \cdot \bar{x} = 0$
Операция с константами	$x \vee 0 = x;\ x \vee 1 = 1$	$x \cdot 1 = x;\ x \cdot 0 = 0$
Двойного отрицания	$\bar{\bar{x}} = x$	

Логические элементы. Таблица истинности. Структурная формула. Функциональная схема.

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: “1” и “0”.

Из этого следует два вывода:

1. одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных;
2. на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, из десятков тысяч которых состоят основные узлы компьютера.

Логический элемент – простейшая структурная единица ЭВМ, выполняющая определенную логическую операцию над двоичными переменными. Реализуется обычно на электронных приборах: диодах, транзисторах, микросхемах. Имеет несколько входов для приема сигналов и выход, для выдачи результирующего сигнала. **Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И–НЕ, ИЛИ–НЕ** и другие, а также **триггер.**

Чтобы представить два логических состояния – “1” и “0” в логических элементах, соответствующие им входные и выходные сигналы, имеют один из двух установленных уровней напряжения: +5 вольт и 0 вольт.

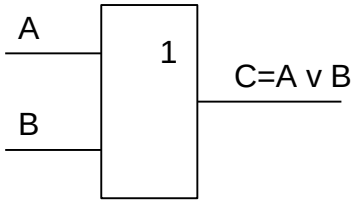
Высокий уровень обычно соответствует значению “истина” (“1”), а низкий – значению “ложь” (“0”).

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Это упрощает запись и понимание сложных логических схем.

Рассмотрим некоторые логические элементы.

Логический элемент ИЛИ, выполняет логическое сложение (дизъюнкция), имеет не менее 2-х входов и один выход.

Схема элемента и таблица истинности.



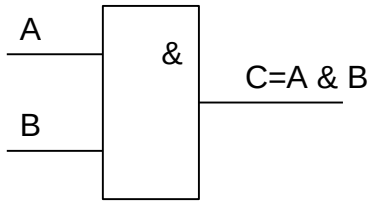
A	B	C=A v B
0	0	0
0	1	1
1	0	1
1	1	1

Работа логического элемента описывается таблицей истинности. *Таблица истинности* – это табличное представление логической схемы (операции), в котором перечислены все возможные сочетания значений истинности входных сигналов (операндов) вместе со значением истинности выходного сигнала (результата операции) для каждого из этих сочетаний. Она содержит n+1 столбец (где n – количество входов (аргументов) и один выход (функция)) и 2^n строк. Чтобы ее построить, мы должны перечислить все возможные комбинации сигналов, которые можно подать на входы. Их будет 2^n. Если 2 входа, то комбинаций 4, если 3 входа, то комбинаций 8.

Чтобы не ошибиться в переборе всех комбинаций, поступают следующим образом. В первую строку записывают значения всех переменных равными нулю и рассматривают полученное n-значное число как число, записанное в двоичной системе счисления. Чтобы получить следующее число к предыдущему добавляют 1 в двоичной системе счисления. И так до тех пор, пока не получим двоичное число, состоящее из единиц. Это число будет последней строкой таблицы истинности.

Логический элемент И, выполняет логическое умножение (конъюнкция), имеет не менее 2-х входов и один выход.

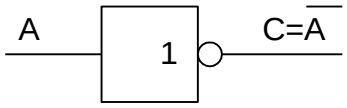
Схема элемента и таблица истинности.



A	B	C=A & B
0	0	0
0	1	0
1	0	0
1	1	1

На выходе логического элемента И будет единица тогда и только тогда, когда на всех входах будут 1

Логический э**лемент НЕ**, выполняет логическое отрицание, имеет один вход и один выход. Схема элемента и таблица истинности.



A	C=$\overline{A}$
0	1
1	0

Эти три логических элемента составляют **полную систему**. Это значит, что с их помощью можно построить любое, сколь угодно сложное устройство. Т.е. если бы мы разрисовали схему компьютера, то она бы состояла из этих 3-х логических элементов.

Логические элементы **И–НЕ, ИЛИ–НЕ** получим из элементов **И** и **ИЛИ**, сделав отрицание выходного сигнала.

Приведенные логические схемы как отдельные устройства в современной вычислительной технике вы не встретите. Элементной базой ЭВМ стали интегральные схемы, которые в одном корпусе содержат набор устройств.

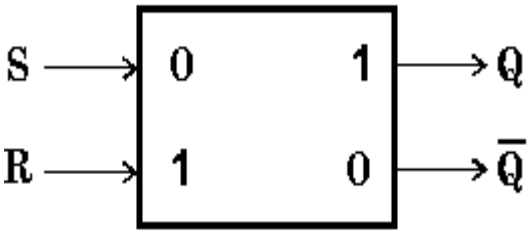
Схема, состоящая из логических элементов, называется *функциональной*.

Формула, описывающая работу функциональной схемы, называется *структурной* или *булевой функцией*.

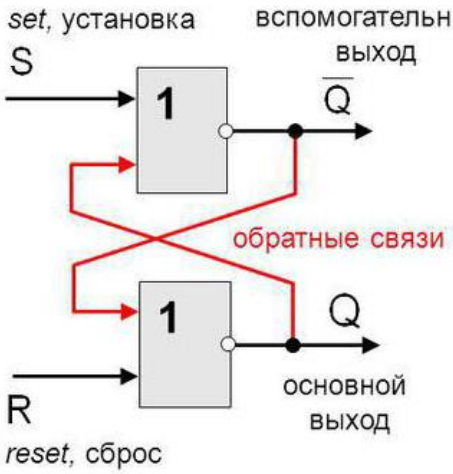
Триггер **–** простейший цифровой автомат с памятью (изобретен в 1918 году Бонч-Бруевичем, руководителем Нижегородской лаборатории связи). Эта электронная схема, широко применяемая в регистрах компьютера для надёжного запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое– двоичному нулю.

Термин **триггер** происходит от английского слова **trigger** – защёлка, спусковой крючок. Для обозначения этой схемы в английском языке чаще употребляется термин **flip-flop**, что в переводе означает "хлопанье". Это звукоподражательное название электронной схемы указывает на её способность почти мгновенно переходить ("перебрасываться") из одного электрического состояния в другое и наоборот.

Самый распространённый тип триггера – так называемый RS-триггер (имеет два входа S и R, соответственно, от английских *set* – установка, и *reset* – сброс и два выхода прямой - Q и инверсный - $\bar{Q}$). Он сохраняет своё предыдущее состояние при нулевых входах, и меняет своё выходное состояние при подаче на один из его входов единицы. При подаче единицы на вход **S** (от английского *Set* - установить) выходное состояние становится равным логической единице. А при подаче единицы на вход **R** (от английского *Reset* - сбросить) выходное состояние становится равным логическому нулю. Обозначение триггера.

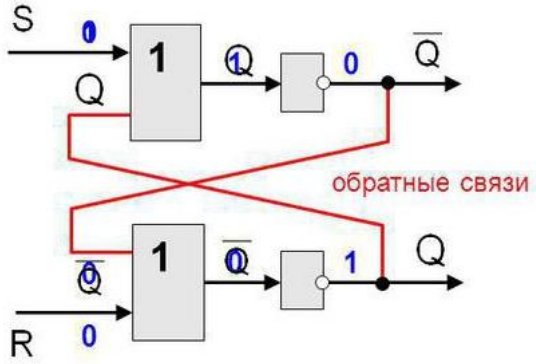


Реализацию триггера с помощью логических элементов ИЛИ–НЕ.



S	R	Q	$\overline{Q}$	режим
0	0	Q	$\overline{Q}$	хранение
0	1	0	1	сброс
1	0	1	0	установка 1
1	1	0	0	запрещен

Реализацию триггера с помощью логических элементов И–НЕ.



S	R	Q	$\overline{Q}$	режим
0	0	Q	$\overline{Q}$	хранение
0	1	0	1	сброс
1	0	1	0	установка 1
1	1	0	0	запрещен

Выход одного элемента подключен ко входу второго, это обеспечивает триггеру два устойчивых состояния и, соответственно, возможность хранения информации в виде 1 или 0. Вход R называют входом установки триггера в нулевое состояние, а вход S – в единичное. Выход Q – прямой, а $\overline{Q}$ – инверсный, т.к. на $\overline{Q}$ потенциал всегда соответствует состоянию, которое противоположно состоянию выхода Q.

Триггер помнит только один двоичный разряд. Для хранения многоразрядных чисел триггеры объединяют в устройство, называемое **регистром**.

Регистр – это устройство, представляющее собой отдельные ячейки внутренней быстродействующей памяти микропроцессора. Они используются для временного хранения единицы информации при обработке данных процессором. Количество триггеров в регистре определяет разрядность компьютера: 8-ми, 16-ти, 32-х, 64-х разрядный компьютер.

Рассмотрим некоторые примеры.

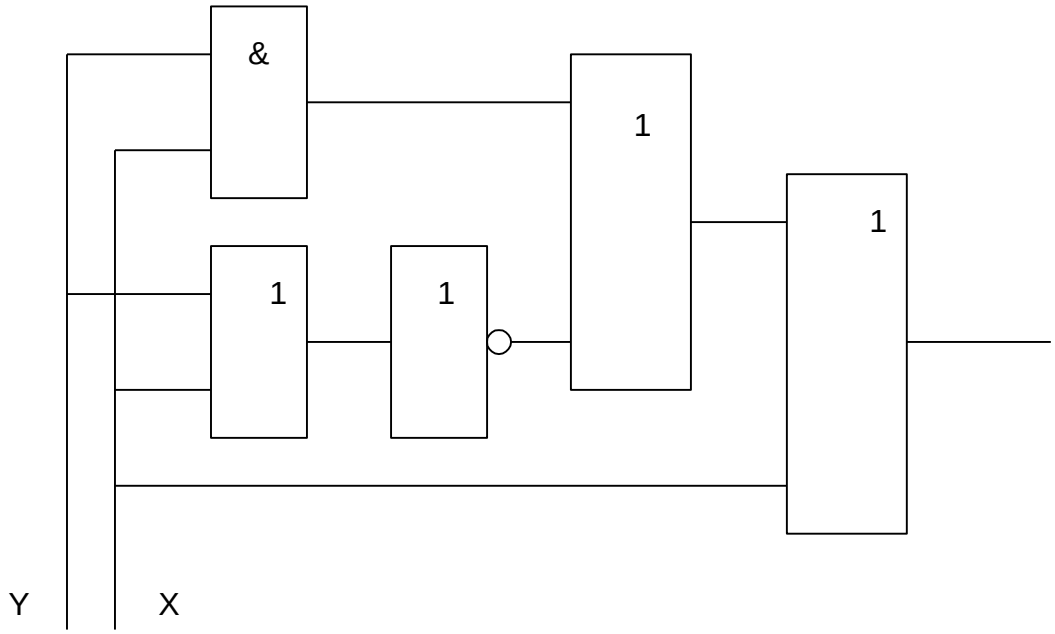
Удобной формой записи при нахождении значений формулы является таблица, содержащая кроме значений переменных и значений формулы также и значения промежуточных формул.

1. **Составим таблицу истинности и функциональную схему для структурной формулы $x \cdot y \vee \overline{x \vee y} \vee x$,** которая содержит две переменные x и y. В первых двух столбцах таблицы запишем четыре возможных пары значений этих переменных, в последующих столбцах – значения промежуточных формул и в последнем столбце – значение формулы. В результате получим таблицу:

Переменные		Промежуточные логические формулы					Формула
x	y	$\overline{x}$	$\overline{x} \cdot y$	$x \vee y$	$\overline{x \vee y}$	$x \cdot y \vee \overline{x \vee y}$	$x \cdot y \vee \overline{x \vee y} \vee x$
0	0	1	0	0	1	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	1	0	0	1	0	0	1

Из таблицы видно, что **при всех наборах значений переменных x и y формула $x \cdot y \vee \overline{x \vee y} \vee x$ принимает значение 1**, то есть является ***тождественно истинной*.**

Функциональная схема.



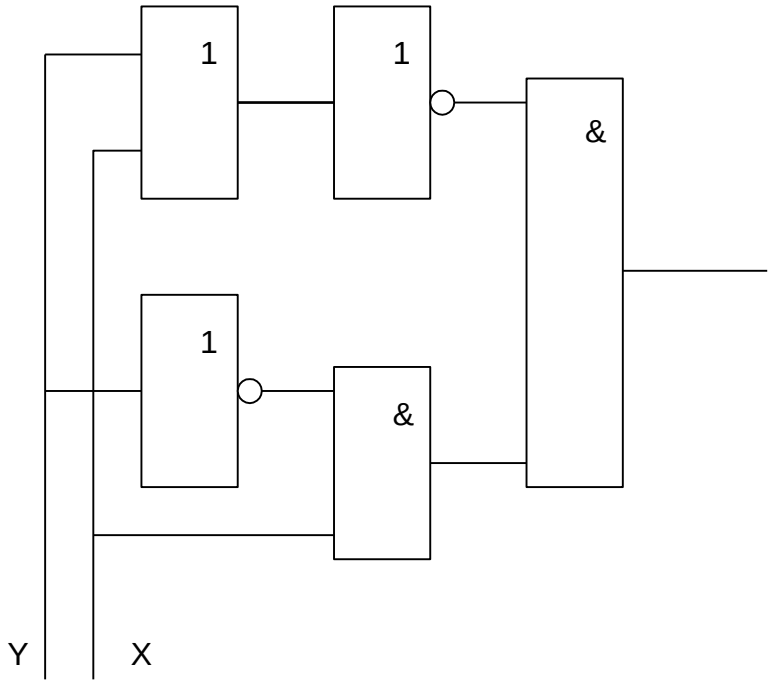
2. **Составим таблицу истинности и функциональную схему для структурной формулы** $\overline{x \vee y} \cdot (x \cdot \bar{y})$**.**

Таблица истинности.

Переменные		Промежуточные логические формулы				Формула
x	y	$x \vee y$	$\overline{x \vee y}$	$\bar{y}$	$x \cdot \bar{y}$	$\overline{x \vee y} \cdot (x \cdot \bar{y})$
0	0	0	1	1	0	0
0	1	1	0	0	0	0
1	0	1	0	1	1	0
1	1	1	0	0	0	0

Из таблицы видно, что **при всех наборах значений переменных x и y формула** $\overline{x \vee y} \cdot (x \cdot \bar{y})$**принимает значение 0**, то есть является ***тождественно ложной***.

Функциональная схема.

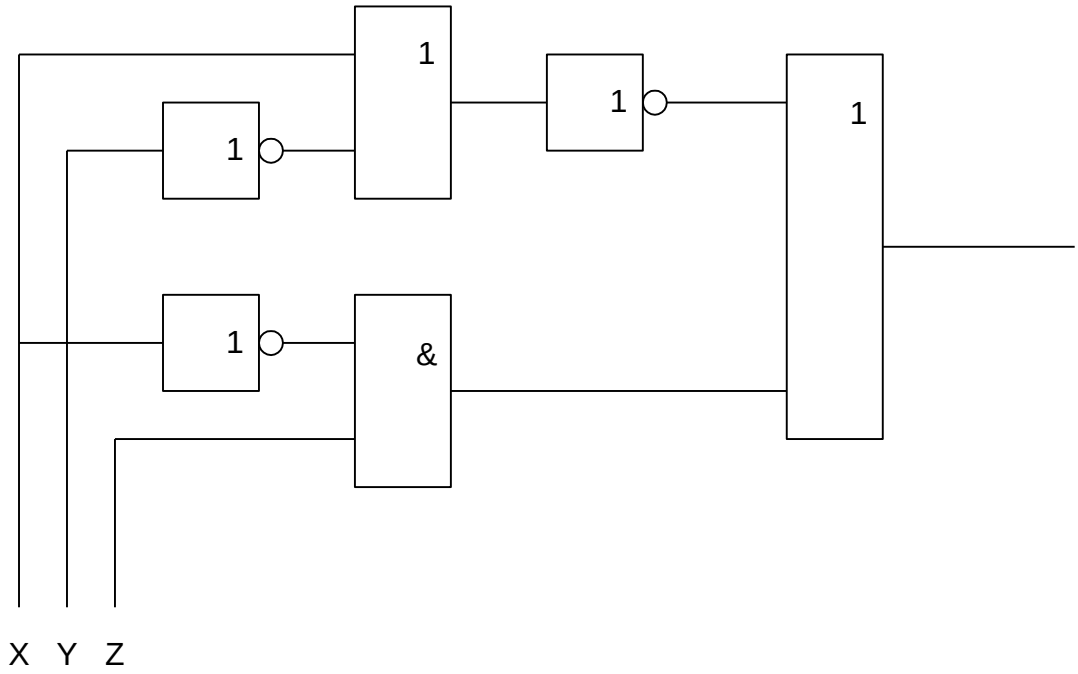


3. **Таблица истинности для структурной формулы $\overline{x \vee \bar{y}} \vee \bar{x} \cdot z$:**

Переменные			Промежуточные логические формулы					Формула
x	y	z	$\bar{y}$	$x \vee \bar{y}$	$\overline{x \vee \bar{y}}$	$\bar{x}$	$\bar{x} \cdot z$	$\overline{x \vee \bar{y}} \vee \bar{x} \cdot z$
0	0	0	1	1	0	1	0	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	1	0	1
0	1	1	0	0	1	1	1	1
1	0	0	1	1	0	0	0	0
1	0	1	1	1	0	0	0	0
1	1	0	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0

Из таблицы видно, что **формула $\overline{x \vee \bar{y}} \vee \bar{x} \cdot z$ в некоторых случаях принимает значение 1, а в некоторых — 0**, то есть является **выполнимой**.

Функциональная схема:



Упрощение формул.

Равносильные преобразования логических формул имеют то же назначение, что и преобразования формул в обычной алгебре. Они служат для упрощения формул или приведения их к определённому виду путем использования основных законов алгебры логики.

Под *упрощением формулы*, не содержащей операций импликации и эквиваленции, понимают равносильное преобразование, приводящее к формуле, которая либо содержит по сравнению с исходной меньшее число операций конъюнкции и дизъюнкции и не содержит отрицаний неэлементарных формул, либо содержит меньшее число вхождений переменных.

Преобразование (минимизация) логических формул (булевых функций).

Использование булевой алгебры позволяет не только более удобно оперировать с булевыми выражениями (представляющими те или иные электронные узлы), чем над схемами или логическими диаграммами, но и на формальном уровне путем эквивалентных преобразований и базовых теорем упрощать их, давая возможность создавать экономически и технически более совершенные электронные устройства любого назначения. То есть минимизация ЛФ необходима для построения комбинационных схем цифровых автоматов минимальной сложности.

Пример. Соревнования по подъему штанги судят три судьи, один из них главный. Надпись «Вес взят» зажигается, если «За» проголосовали все судьи или главный судья и один из помощников.

Решение.

Пусть судья x_1 – главный, а судьи x_2 и x_3 – помощники. Тогда таблица истинности будет иметь вид:

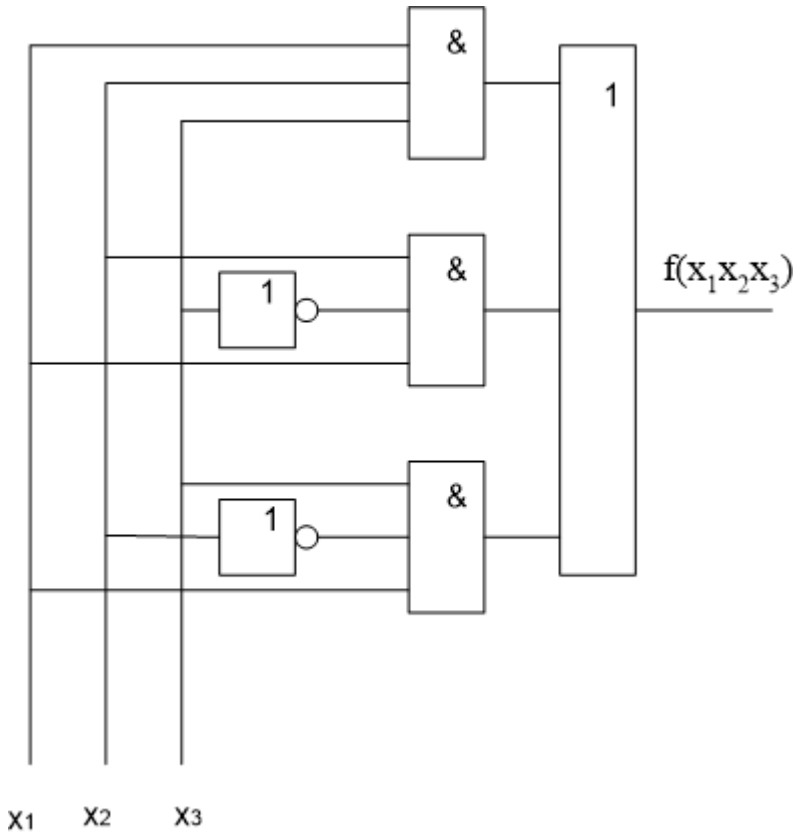
Таблица истинности

x_1	x_2	x_3	$f(x_1x_2x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Структурная формула (булева функция), описывающая работу электронного устройства для голосования будет следующей:

$$f(x_1,x_2,x_3) = x_1*\overline{x_2}*x_3 + x_1*x_2*\overline{x_3} + x_1*x_2*x_3$$

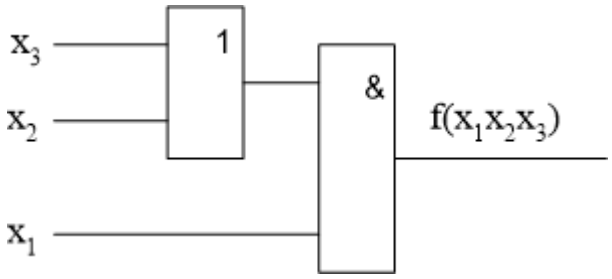
Функциональная схема электронного устройства:



Упростим булеву функцию.

$$f(x_1x_2x_3) = x_1*\overline{x_2}*x_3 + x_1*x_2*x_3 + x_1*x_2*\overline{x_3} + x_1*x_2*x_3 = x_1*x_3*(\overline{x_2} + x_2) + x_1*x_2*(\overline{x_3} + x_3) = x_1*x_3 + x_1*x_2 = x_1*(x_3 + x_2)$$

Построим функциональную схему, соответствующую упрощенной булевой функции (электронное устройство)



Вторая схема содержит меньше электронных элементов, следовательно, будет дешевле в производстве и надежнее в работе.

Некоторые преобразования логических формул похожи на преобразования формул в обычной алгебре (вынесение общего множителя за скобки, использование переместительного и сочетательного законов и т.п.), тогда как **другие преобразования основаны на свойствах, которыми не обладают операции обычной алгебры** (использование распределительного закона для конъюнкции, законов поглощения, склеивания, де Моргана и др.).

Покажем на примерах некоторые **приемы и способы, применяемые при упрощении логических формул:**

1) $$\overline{x \vee y} \cdot (x \cdot \bar{y}) = \bar{x} \cdot \bar{y} \cdot (x \cdot \bar{y}) = \bar{x} \cdot x \cdot \bar{y} \cdot \bar{y} = 0 \cdot \bar{y} \cdot \bar{y} = 0 \cdot \bar{y} = 0$$

(законы алгебры логики применяются в следующей последовательности: правило де Моргана, сочетательный закон, правило операций переменной с её инверсией и правило операций с константами);

2) $$\bar{x} \cdot y \vee \overline{x \vee y} \vee x = \bar{x} \cdot y \vee \bar{x} \cdot \bar{y} \vee x = \bar{x} \cdot (y \vee \bar{y}) \vee x = \bar{x} \vee x = 1$$

(применяется правило де Моргана, выносится за скобки общий множитель, используется правило операций переменной с её инверсией);

3) $$(x \vee y) \cdot (\bar{x} \vee y) \cdot (\bar{x} \vee \bar{y}) = (x \vee y) \cdot (\bar{x} \vee y) \;\; (\bar{x}vy) \cdot (\bar{x} \vee \bar{y}) = y \cdot \bar{x}$$

(**повторяется** второй **сомножитель**, что разрешено законом идемпотенции; затем комбинируются два первых и два последних сомножителя и используется закон склеивания);

4)
$$x \cdot \bar{y} \vee \bar{x} \cdot y \cdot z \vee x \cdot z = x \cdot \bar{y} \vee \bar{x} \cdot y \cdot z \vee x \cdot z \cdot (y \vee \bar{y}) =$$
$$= x \cdot \bar{y} \vee \bar{x} \cdot y \cdot z \vee x \cdot y \cdot z \vee x \cdot \bar{y} \cdot z = (x \cdot \bar{y} \vee x \cdot \bar{y} \cdot z) \vee (\bar{x} \cdot y \cdot z \vee x \cdot y \cdot z) =$$
$$= x \cdot \bar{y} \vee y \cdot z$$

(**вводится вспомогательный логический сомножитель** $(y \vee \bar{y})$; затем комбинируются два крайних и два средних логических слагаемых и используется закон поглощения);

5) $$\overline{x \cdot y \vee \bar{z}} = \overline{x \cdot y} \cdot \bar{\bar{z}} = (\bar{x} \vee \bar{y}) \cdot z$$

(сначала добиваемся, чтобы знак отрицания стоял только перед отдельными переменными, а не перед их комбинациями, для этого дважды применяем правило де Моргана; затем используем закон двойного отрицания);

6) $$x \cdot y \vee x \cdot y \cdot z \vee x \cdot z \cdot p = x \cdot (y \cdot (1 \vee z) \vee z \cdot p) = x \cdot (y \vee z \cdot p)$$

(выносятся за скобки общие множители; применяется правило операций с константами);

7)
$$x \vee \overline{y \cdot \bar{z}} \vee \overline{\bar{x} \vee y \vee \bar{z}} = x \vee \bar{y} \vee \bar{\bar{z}} \vee \bar{\bar{x}} \cdot \bar{y} \cdot \bar{\bar{z}} = x \vee \bar{y} \vee z \vee x \cdot \bar{y} \cdot z =$$
$$= x \vee z \vee (\bar{y} \vee x \cdot \bar{y} \cdot z) = x \vee z \vee \bar{y}$$

(к отрицаниям неэлементарных формул применяется правило де Моргана; используются законы двойного отрицания и склеивания);

8)

$$x\cdot\overline{y}\vee x\cdot y\cdot z\vee x\cdot\overline{y}\cdot z\vee x\cdot\overline{y\cdot z}=x\cdot(\overline{y}\vee y\cdot z\vee\overline{y}\cdot z\vee\overline{y\cdot z})=$$
$$=x\cdot((\overline{y}\vee\overline{y}\cdot z)\vee(y\cdot z\vee\overline{y\cdot z}))=x\cdot(\overline{y}\vee\overline{y}\cdot z\vee 1)=x\cdot 1=x$$

(общий множитель x выносится за скобки, комбинируются слагаемые в скобках — первое с третьим и второе с четвертым, к дизъюнкции $y\cdot z\vee\overline{y\cdot z}$ применяется правило операции переменной с её инверсией);

9)

$$(x\cdot\overline{y}\vee z)\cdot(\overline{x}\vee y)\vee\overline{z}=x\cdot\overline{y}\cdot\overline{x}\vee x\cdot\overline{y}\cdot y\vee z\cdot\overline{x}\vee z\cdot y\vee\overline{z}=0\vee 0\vee$$
$$\vee z\cdot\overline{x}\vee z\cdot y\vee\overline{z}=z\cdot\overline{x}\vee(z\vee\overline{z})\cdot(y\vee\overline{z})=z\cdot\overline{x}\vee 1\cdot(y\vee\overline{z})=z\cdot\overline{x}\vee y\vee$$
$$\vee\overline{z}=(z\cdot\overline{x}\vee\overline{z})\vee y=(z\vee\overline{z})\cdot(\overline{x}\vee\overline{z})\vee y=1\cdot(\overline{x}\vee\overline{z})\vee y=\overline{x}\vee\overline{z}\vee y$$

(используются распределительный закон для дизъюнкции, правило операции переменной с ее инверсией, правило операций с константами, переместительный закон и распределительный закон для конъюнкции);

10)

$$x\cdot y\cdot(\overline{x}\cdot z\vee\overline{\overline{x\cdot y}\cdot z}\vee z\cdot t)=x\cdot y\cdot(\overline{x}\cdot z\vee\overline{\overline{x\cdot y}}\vee\overline{z}\vee z\cdot t)=$$
$$=x\cdot y\cdot(\overline{x}\cdot z\vee x\cdot y\vee\overline{z}\vee z\cdot t)=x\cdot y\vee x\cdot y\cdot\overline{z}\vee x\cdot y\cdot z\cdot t=x\cdot y$$

(используются правило де Моргана, закон двойного отрицания и закон поглощения).

Из этих примеров видно, что при упрощении логических формул не всегда очевидно, какой из законов алгебры логики следует применить на том или ином шаге. Навыки приходят с опытом.

ТЕМА 8. Принцип программного управления компьютером.

Карта Карно – это фактически таблица истинности, представленная в специальном виде. Для функции двух переменных (в таблице истинности 4 строки) карта Карно – это квадрат 2 х 2 клетки. Для функции трех переменных (в таблице истинности 8 строк) – это прямоугольник 2 х 4 клетки. Для функции четырех переменных (в таблице истинности 16 строк) – это квадрат 4 х 4 клетки. Метод минимизации логических функций с помощью карт Карно является наиболее удобным при не большем количестве переменных (не более пяти).

Для успешной работы с картами Карно необходимо соблюдать правила разметки осей карты:

1. Вертикальная и горизонтальная оси размечается независимо от каждой.
2. Начинать разметку можно с любого сочетания переменных.
3. Все сочетания переменных должны быть перечислены.
4. Переменные, обозначающие клетки карты Карно, должно отличаться не более чем одним знаком, причем соседними являются крайние клетки строки (столбца).

Примеры карт Карно.

Таблица истинности и карта Карно для функции от двух переменных f(a,b). Переменные a и b – это логические переменные. Значение 1 – истина, 0 – ложь.

a	b	f(a,b)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)

a \ b	0	1
0	f(0,0)	f(0,1)
1	f(1,0)	f(1,1)

Таблица истинности и карта Карно для функции от трех переменных f(a,b,c)

a	b	c	f(a,b,c)
0	0	0	f(0,0.0)
0	0	1	f(0,0,1)
0	1	0	f(0,1,0)
0	1	1	f(0,1,1)
1	0	0	f(1,0,0)
1	0	1	f(1,0,1)
1	1	0	f(1,1,0)
1	1	1	f(1,1,1)

a \ b c	0 0	0 1	1 1	1 0
0	f(0,0,0)	f(0,0,1)	f(0,1,1)	f(0,1,0)
1	f(1,0,0)	f(1,0,1)	f(1,1,1)	f(1,1,0)

…

Таблица истинности и карта Карно для функции от четырех переменных f(a,b,c,d)

a	b	c	d	f(a,b,c,d)
0	0	0	0	f(0,0.0,0)
0	0	0	1	f(0,0,0,1)
0	0	1	0	f(0,0,1,0)
0	0	1	1	f(0,0,1,1)
0	1	0	0	f(0,1,0,0)
0	1	0	1	f(0,1,0,1)
0	1	1	0	f(0,1,1,0)
0	1	1	1	f(0,1,1,1)
1	0	0	0	f(1,0,0,0)
1	0	0	1	f(1,0,0,1)
1	0	1	0	f(1,0,1,0)
1	0	1	1	f(1,0,1,1)
1	1	0	0	f(1,1,0,0)
1	1	0	1	f(1,1,0,1)
1	1	1	0	f(1,1,1,0)
1	1	1	1	f(1,1,1,1)

a b \ c d	0 0	0 1	1 1	1 0
0 0	f(0,0,0,0)	f(0,0,,0,1)	f(0,0,1,1)	f(0,0,1,0)
0 1	f(0,1,0,0)	f(0,1,0,1)	f(0,1,1,1)	f(0,1,1,0)
1 1	f(1,1,0,0)	f(1,1,0,1)	f(1,1,1,1)	f(1,1,1,0)
1 0	f(1,0,0,0)	f(1,0,0,1)	f(1,0,1,1)	f(1,0,1,0)

Рассмотрим смысл четвертого правила разметки карты Карно.

В первом столбце значения **a** и **b** равны 0, 0. Поэтому во втором столбце значения **a** и **b** не могут быть 1, 1, так как отличаются двумя знаками. И в последнем столбце не могут быть 1, 1, так как тоже отличаются двумя знаками. Могут быть 0, 1 или 1, 0, так как отличаются только одним знаком.

Во втором столбце значения равны соответственно 0, 1. Поэтому ни в первом столбце, ни в третьем столбце не должны быть значения 1, 0, так как не будет отличия в знаках. И так для всех столбцов и строк. То есть при разметке осей нужно четко следить за тем, чтобы соседними не оказались сочетания 00 и 11, либо 01 и 10.

Пример заполнения карты Карно на основании таблицы истинности.

x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
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	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

x_1x_2 \ x_3x_4	0 0	0 1	1 1	1 0
0 0	1	0	0	0
0 1	1	1	0	1
1 1	1	1	1	0
1 0	1	1	0	1

Процесс минимизации логической функции с использованием карты Карно:

1. В карте Карно обводим группы единиц контурами. Контур может содержать одну, две или четыре клетки, т.е. количество клеток внутри контура должно быть равно степени двойки. Клетки должны составлять прямоугольник или квадрат.

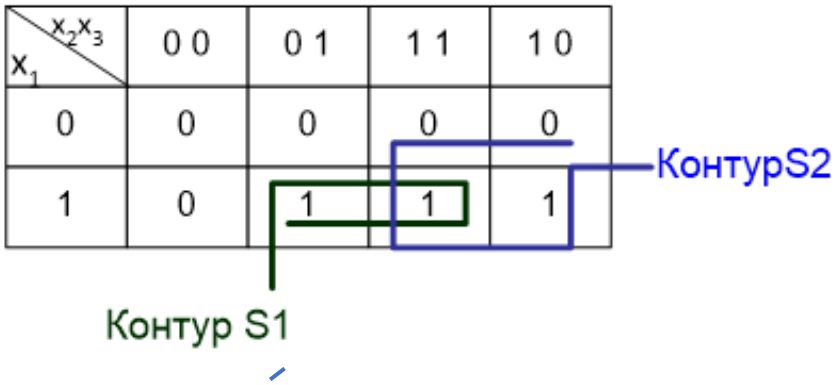
2. При проведении контуров крайние строки карты (верхние и нижние, левые и правые), а также угловые клетки считаются соседними.
3. В контур можно включать только соседние клетки, содержащие только единицы или только нули.
4. Каждый контур должен включать максимально возможное количество клеток. Контур с единицами не должен содержать клеток с нулями. Одноименные контуры могут накладываться один на другой, то есть одна и та же единица (ноль) может входить в несколько единичных (нулевых) контуров.
5. После обведения контуров нужно записать *минимальное выражение как логическую сумму логических произведений.* Каждому логическому произведению соответствует один контур карты Карно. *В произведение входят только те переменные, которые остаются в данном контуре неизменными.* При этом переменная входит в произведение с инверсией, если ее значение в данном контуре равно 0, и без инверсии, если ее значение равно 1.

Примеры.

Пример 1. Минимизируем с помощью карты Карно задачу по подъему штанги, рассмотренную выше. Логическая функция, таблица истинности и карта Карно:

$f(x_1x_2x_3) = x_1*\overline{x_2}*x_3 + x_1*x_2*\overline{x_3} + x_1*x_2*x_3$

x_1	x_2	x_3	$f(x_1x_2x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



В карте три единицы. Следовательно, будет два контура. S1 и S2.

Внутри **контура S1** переменная x_1 не меняет значение (ее значение единица), следовательно, она будет в произведении. Переменная x_2 меняет значение с 0 на 1, следовательно, ее не будет в произведении. Переменная x_3 не меняет значение, следовательно, она будет в произведении. В результате произведение переменных в контуре S1 будет такое x_1*x_3.

Внутри **контура S2** переменная x_1 не меняет значение, следовательно, она будет в произведении. Переменная x_2 не меняет значение, следовательно, она будет в произведении. Переменная x_3 меняет значение с 1 на 0, следовательно, ее не будет в произведении. В результате произведение переменных в контуре S2 будет такое x_1*x_2.

Таким образом минимальная функция будет

$f(x_1x_2x_3) = x_{1*}x_3 + x_{1*}x_2 = x_{1*}(x_3 + x_2)$,

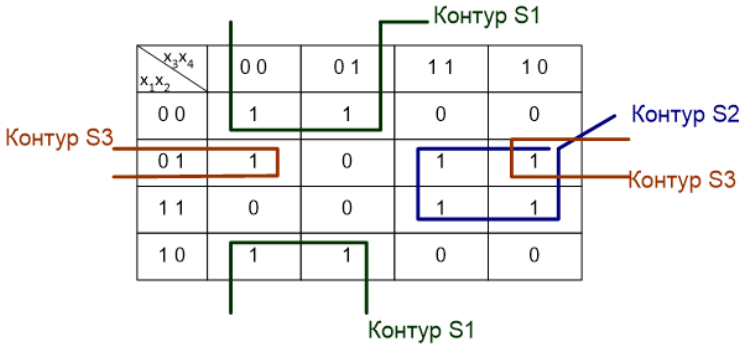
то есть получили тот же результат, ту же самую минимальную функцию.

Пример 2. Минимизируем функцию от четырех переменных

$f(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3\overline{x}_4 + \overline{x}_1x_2x_3x_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3x_4 + x_1x_2x_3\overline{x}_4 + x_1x_2x_3x_4$.

Таблица истинности и карта Карно:

x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



В результате объединения единиц получили три контура. Контур S1 содержит 4 единицы (клетки): две в верхней строке и две в нижней. Контур S2 содержит четыре единицы (клетки), расположенные рядом. Контур S3 содержит две единицы (клетки): одна в крайнем левом столбце, вторая в крайнем правом столбце.

Запишем произведение переменных в каждом контуре. Внутри **контура S1** переменная $\mathbf{x_1}$ меняет значение с 0 на 1, следовательно, не будет входить в произведение и переменная $\mathbf{x_4}$, меняет значение с 0 на 1 – тоже не будет

входить в произведение. Так как значения переменных x_2 и x_3 равны 0, то в произведение будут входить отрицания этих переменных, то есть x_2, то есть $\overline{x}_2 * \overline{x}_3$.

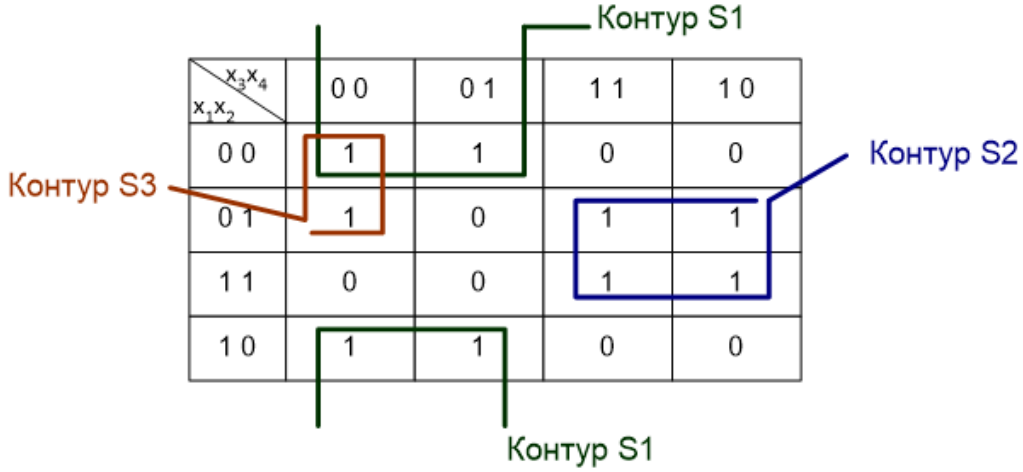
Внутри **контура S2** переменная x_1 меняет значение с 0 на 1, следовательно, не будет входить в произведение и переменная x_4, меняет значение с 1 на 0 – тоже не будет входить в произведение. В результате произведение переменных в контуре S1 будет такое $x_2 * x_3$.

Внутри **контура S3** значение меняет только переменная x_3, следовательно, не будет входить в произведение. Так как значения переменных x_1 и x_4 равны нулю, то в произведение будут входить отрицания этих переменных, то есть $\overline{x}_1 * \overline{x}_4$. В результате произведение переменных в контуре S1 будет такое $\overline{x}_1 * x_2 * \overline{x}_4$.

Следовательно, минимальная логическая функция будет иметь вид

$f(x_1,x_2,x_3,x_4) = \overline{x}_2\overline{x}_3 + x_2x_3 + \overline{x}_1x_2\overline{x}_4$

Объединим в таблице Карно клетки по-другому (контур S3 будет иметь другой вид):



Тогда поменяется только произведение в **контуре S3**. Так как в контуре меняет значение только переменная x_2, то она не будет входить в произведение. В произведении будут только три переменные x_1, x_3, x_4 и с отрицаниями, так как их значения равны нулю. В этом случае минимальная функция будет

$f(x_1,x_2,x_3,x_4) = \overline{x}_2\overline{x}_3 + x_2x_3 + \overline{x}_1\overline{x}_3\overline{x}_4$

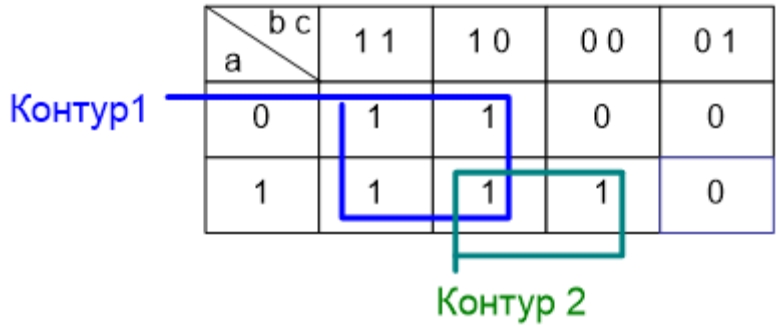
Предпочтение следует отдать первой минимальной функции, так как в ней на одно отрицание меньше.

Пример 3. Минимизировать с помощью карты Карно логическую функцию от трех переменных, представленную таблицей истинности и нарисовать функциональную схему.

a	*b*	*c*	*f*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Строим карту Карно.

При таком варианте разметки осей первый контур, состоящий из четырех единиц, получается разорванным. Сделаем другую разметку осей. Получим следующую карту Карно.



При такой разметке осей контур будет иметь нормальные очертания, а выражение, ему соответствующее, останется без изменений. Учитывая, что при данном горизонтальном начертании карты Карно крайние столбцы являются соседними, ее можно представить себе, как цилиндр, развернутый на плоскости.

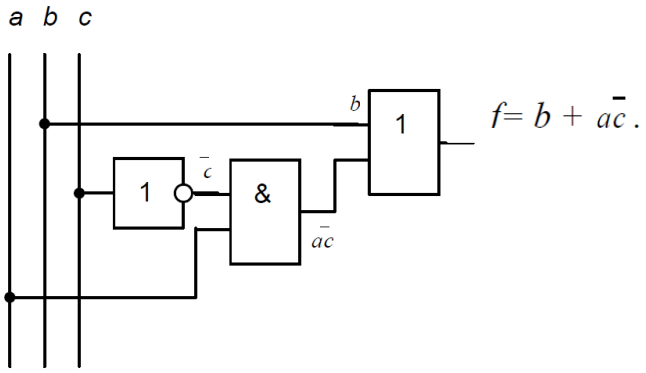
Логическая функция, соответствующая таблице истинности, имеет вид

$$f = \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + ab\bar{c} + abc\,.$$

А минимальная функция, полученная в результате минимизации с помощью карты Карно, имеет следующий вид

$$f = b + a\bar{c}\,.$$

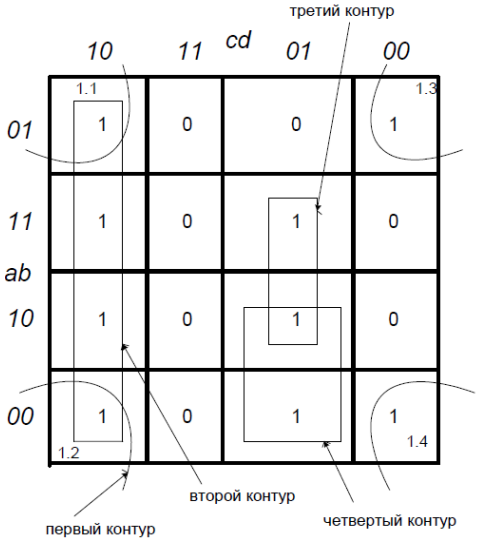
Нарисуем функциональную схему, соответствующую минимальной функции

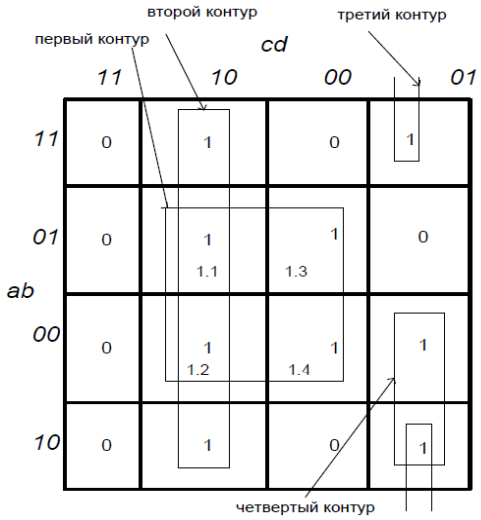


Пример 4. Минимизировать с помощью карты Карно логическую функцию от четырех переменных, представленную таблицей истинности и нарисовать функциональную схему.

a	b	c	d	f(a,b,c,d)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

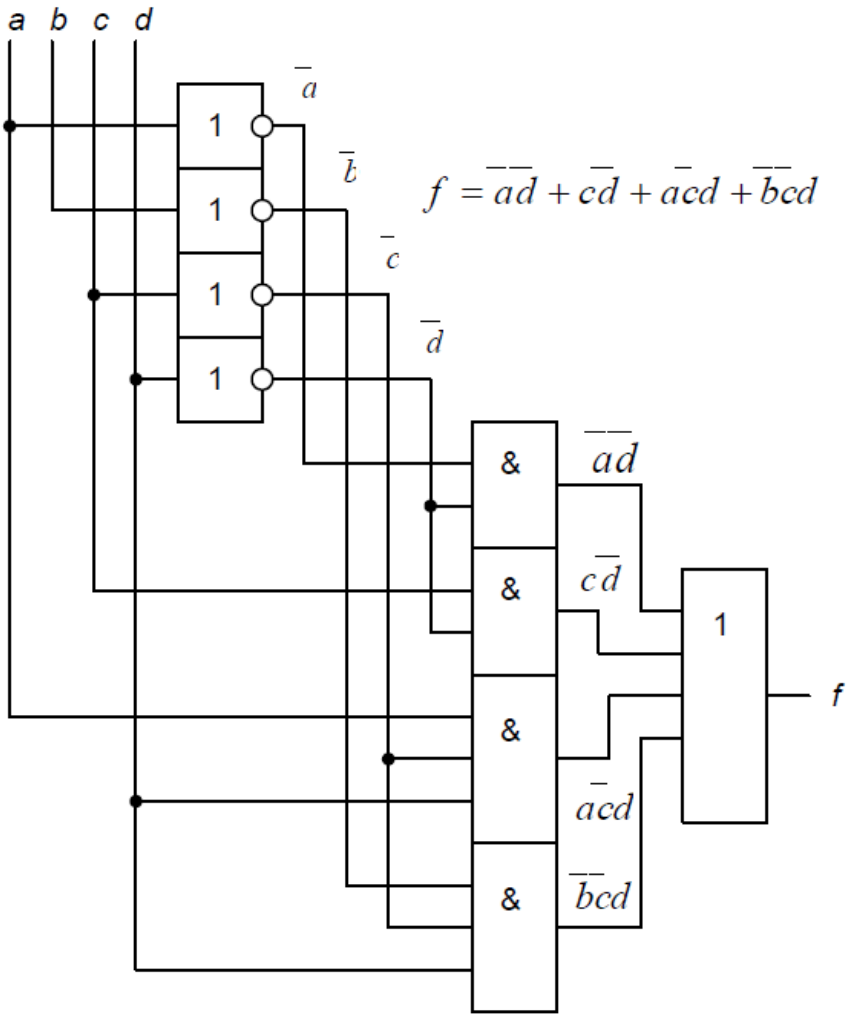
Построим две карты Карно с разной разметкой осей





При первоначально выбранной разметке осей первый контур, состоящий из четырех единиц с номерами 1.1, 1.2, 1.3 и 1.4, расположенных по углам карты, получается разорванным. Если же поменять разметку осей, то контур будет иметь очертания квадрата, а выражение, ему соответствующее, останется без изменений. Учитывая, что крайние столбцы являются соседними и крайние строки являются соседними, **карту Карно** для функции четырех переменных можно представить себе, как торроид, развернутый на плоскости.

Функциональная схема:



Программный принцип управления. Команда, ее структура. Программа. Взаимодействие устройств при выполнении команд.

С созданием первого компьютера связано имя выдающегося теоретика того времени – Джона фон Неймана, разработавшего архитектуру компьютера, которой придерживаются разработчики до нашего времени. Помнению фон Неймана, компьютер должен состоять из центрального арифметико-логического устройства, центрального устройства управления, запоминающего устройства и устройства ввода-вывода информации.

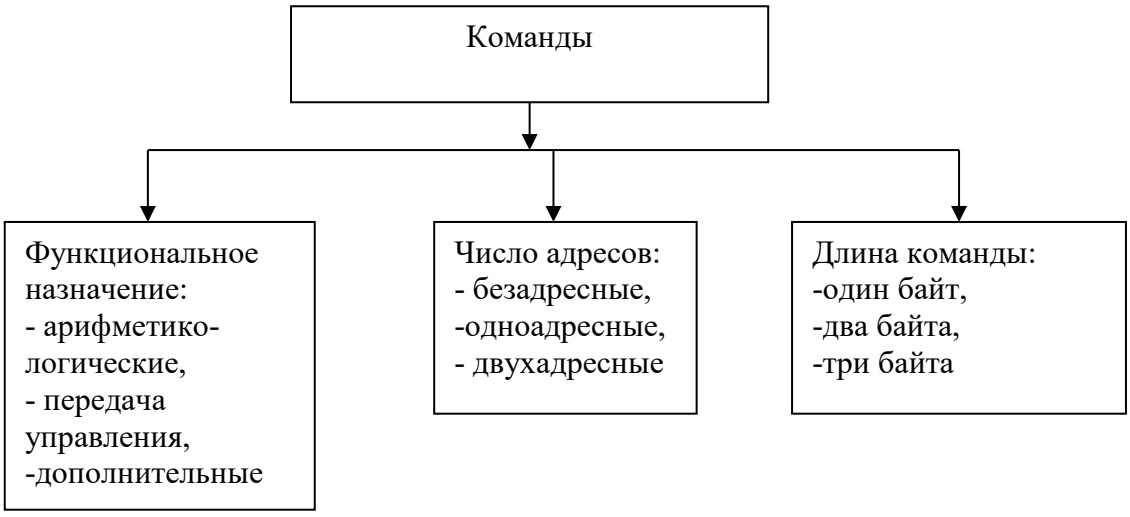
Компьютер, по его мнению, должен работать с двоичными числами, быть электронным (а не электрическим); выполнять операции последовательно.

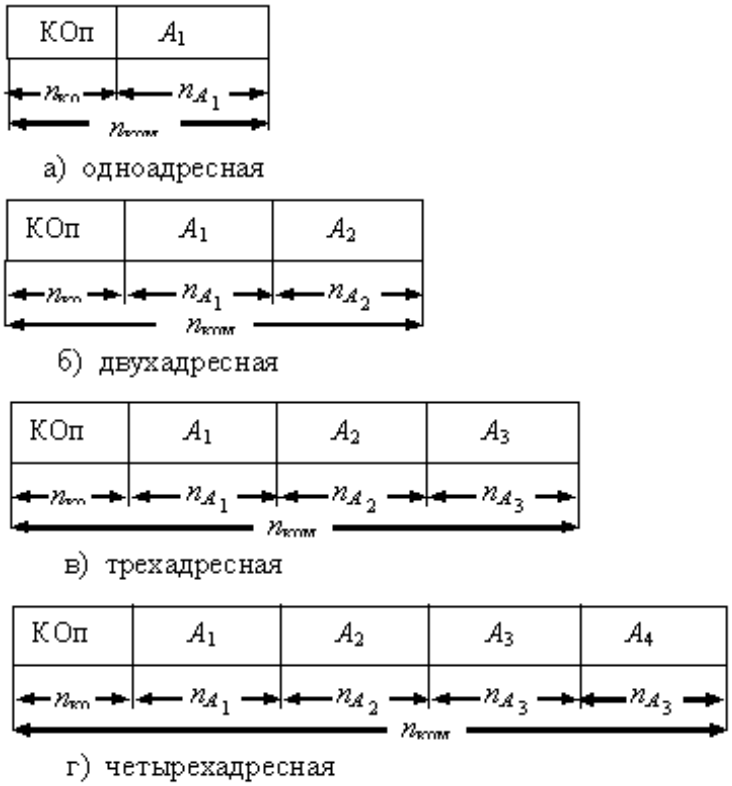
Все вычисления, предписанные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов-команд. Каждая команда содержит указания на конкретную выполняемую операцию, место нахождения (адреса) операндов и ряд служебных признаков. Операнды - переменные, значения которых участвуют в операциях преобразования данных. Список (массив) всех переменных (входных данных, промежуточных значений и результатов вычислений) является еще одним неотъемлемым элементом любой программы.

Для доступа к программам, командам и операндам используются их адреса. В качестве адресов выступают номера ячеек памяти ЭВМ, предназначенных для хранения объектов. Информация (командная и данные: числовая, текстовая, графическая и т.п.) кодируется двоичными цифрами 0 и 1.

Поэтому различные типы информации, размещенные в памяти ЭВМ, практически неразличимы, идентификация их возможна лишь при выполнении программы, согласно ее логике, по контексту.

На рисунке представлена схема классификации команд по трем признакам: функциональному назначению, числу адресов и длине команды.





В памяти компьютера в виде кодов находится выполняемая программа и все исходные данные. Каждая команда, входящая в программу, хранится, как правило, в одной ячейке. Каждая ячейка памяти независимо от того, что в ней хранится: код числа или код команды, - имеет свой номер, называемый адресом ячейки.

Работа ЭВМ по программе начинается с того, что устройство управления (УУ) "заглядывает" в так называемый счетчик адреса команд (СЧАК). Перед началом работы в СЧАК помещается адрес ячейки, в которой хранится самая первая команда программы. С выполнения именно этой команды ЭВМ начинает решение задачи. После выполнения этой команды в СЧАК помещается адрес следующей команды и т.д., пока в СЧАК не окажется номер команды: "конец работы по программе", выполнив которую ЭВМ останавливается.

Устройство управления находит ячейку по ее адресу, затем расшифровывает команду и настраивает арифметическое устройство (АУ) на ее выполнение. УУ обеспечивает подачу в АУ тех чисел, над которыми выполняется операция. Эти числа находятся по адресам, которые содержатся в команде.

Устройство управления после того, как оно снабдило арифметическое устройство необходимыми ему числами и настроило его на выполнение конкретной операции, некоторое время бездействует, дает время арифметическому устройству на выполнение операции.

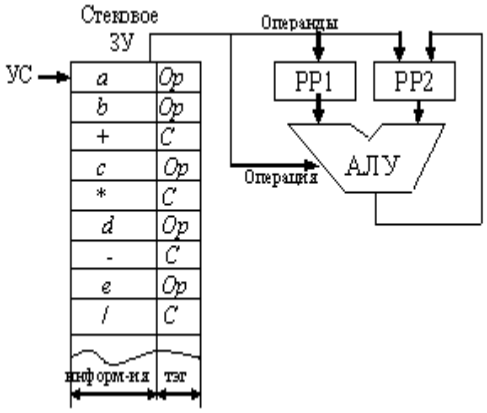
Арифметическое устройство выполняет порученную ему операцию над кодами чисел. Полученный результат АУ содержит на своих выходах. Окончив работу, АУ дает УУ сигнал об этом. Устройство управления направляет результат в ячейку памяти, адрес которой определен либо

программистом, либо транслятором. После этого УУ вновь "заглядывает" в СЧАК и начинает рассмотренный цикл сначала.

Может оказаться так, что очередная команда есть команда ввода (вывода) информации. В этом случае УУ обращается к тому из устройств ввода (вывода), которое предусмотрел программист, и настраивает его на работу. По окончании ввода (вывода) всей подготовленной для этого информации устройство вводы (вывода) дает сигнал УУ. Затем УУ вновь "заглядывает в СЧАК", и рабочий цикл повторяется еще раз.

Так, следуя программе, выполняя одну команду за другой, УУ координирует совместную работу всех блоков ЭВМ.

Несколько особое положение занимает безадресное *кодирование команд*. Оно используются в компьютерах, имеющих стековую организацию памяти. Обращение к ячейкам такой памяти производится последовательно с помощью специального указателя стека (УС), определяющего рабочую в данный момент ячейку. Каждая ячейка снабжена тэгом – специальным признаком хранимой информации. Такая ЭВМ имеет структуру, представленную на рисунке. В ее состав помимо АЛУ входят два специальных буферных регистра РР1 и РР2. Здесь значение тэгов следующее: Op – в данной ячейке хранится операнд, С – признак наличия в ячейке кода операции.



Проиллюстрируем работу такой ЭВМ на примере вычисления выражения ((a + b) * c - d) / e.

На первых двух тактах работы из памяти извлекаются операнды a и b и помещаются в рабочие регистры РР1 и РР2. Считав следующую ячейку стековой памяти, устройство управления по ее тэгу определяет, что данная информация представляет собой код операции. Этот код направляется в АЛУ, где и проводится сложение хранящихся в регистрах операндов с записью результата в один из рабочих регистров. Так как в следующей ячейке хранится операнд, то он направляется в РР, свободный от записанного результата. После этого производится выполнение следующей операции и так далее.