Преобразователи кодов

номер	8421	7421	5421	Код Айкена 2421	Код Грея	Код с избытком 3 *N+3*	Дополнение до 9 *9-N*	Код Джонсона	Дополнение до 10 *10-N*
0	0000	0000	0000	0000	0000	0011	1001	00000	1010
1	0001	0001	0001	0001	0001	0100	1000	10000	1001
2	0010	0010	0010	0010	0011	0101	0111	11000	1000
3	0011	0011	0011	0011	0010	0110	0110	11100	0111
4	0100	0100	0100	0100	0110	0111	0101	11110	0110
5	0101	0101	1000	1011	0111	1000	0100	11111	0101
6	0110	0110	1001	1100	0101	1001	0011	01111	0100
7	0111	1000	1010	1101	0100	1010	0010	00111	0011
8	1000	1001	1011	1110	1100	1011	0001	00011	0010
9	1001	1010	1100	1111	1101	1100	0000	00001	0001

Широкое применение в вычислительной технике находят преобразователи кодов, преобразующие числовую информацию из одной двоичной форму в другую

Коды Грея часто используются в датчиках-энкодерах. Их использование удобно тем, что два соседних значения шкалы сигнала отличаются только в одном разряде. Также они используются для кодирования номера дорожек в жёстких дисках.

Преобразование двоичного кода в код Грея

Коды Грея легко получаются из двоичных чисел путём побитовой операции «Исключающее ИЛИ» с тем же числом, сдвинутым вправо на один бит. Следовательно, *i*-й бит кода Грея *Gi* выражается через биты двоичного кода *Bi* следующим образом:

где – операция «исключающее ИЛИ»; биты нумеруются справа налево, начиная с младшего.

```
function BinToGray(b: integer): integer;

begin
 BinToGray := b xor (b shr 1)
end;
```

Пример: преобразовать двоичное число 10110 в код Грея.

```
10110
01011
---------
11101
```

Код 5421
Правило: если число >=5, то устанавливается бит 5.
Далее все однозначно.

код 5421

0	0000
1	0001
2	0010
3	0011
4	0100
5	1000
6	1001
7	1010
8	1011
9	1100

Унитарный код

Унитарный код (его еще называют кодом "1 из N")
Каждой цифре предоставляем свой разряд
(1 только в одной позиции N-разрядного кода)

Число	Значение
0	0000000001
1	0000000010
2	0000000100
3	0000001000
4	0000010000
5	0000100000
6	0001000000
7	0010000000
8	0100000000
9	1000000000

код 74210

Код 74210 (его еще называют "2 из 5") использует все возможные перестановки из 5 символов с 2 единицами (их 10 штук).
Премущества - если один из битов измениться - сразу видно что ошибка. Потому что число единиц будет нечетное.

Число	74210
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

Перестановка	Число
XXX--	1
XX-X-	2
X-XX-	4
-XXX-	7
XX--X	3
X-X-X	5
-XX-X	8
X--XX	6
-X-XX	9
--XXX	0

BCD с избытком 3

Число	Значение
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Четырёхбитный код Джонсона

- . Код Джонсона — двоичная система счисления, в которой два соседних значения различаются только в одном двоичном разряде.

0000
0001
0011
0111
1111
1110
1100
1000

Операции циклического сдвига

- В Паскаль определены еще две операции над данными целого типа, имеющие тот же уровень приоритета, что и операции **and, *, /, div** и **mod**. Это операции **shl** и **shr**, которые сдвигают последовательность битов на заданное число позиций влево или вправо соответственно. При этом биты, которые выходят за разрядную сетку, теряются. При выполнении операции **shl** освободившиеся справа биты заполняются нулями. При выполнении операции **shr** освободившиеся слева биты заполняются единицами при сдвиге вправо отрицательных значений и нулями в случае положительных значений.
- С помощью операции **shl** возможна замена операции умножения целых чисел на степени двойки. Следующие пары выражений приводят к одинаковому результату: (a shl 1) = a * 2, (a shl 2) = a * 4, (a shl3) = a * 8.

Пример побитовых операций и циклического сдвига

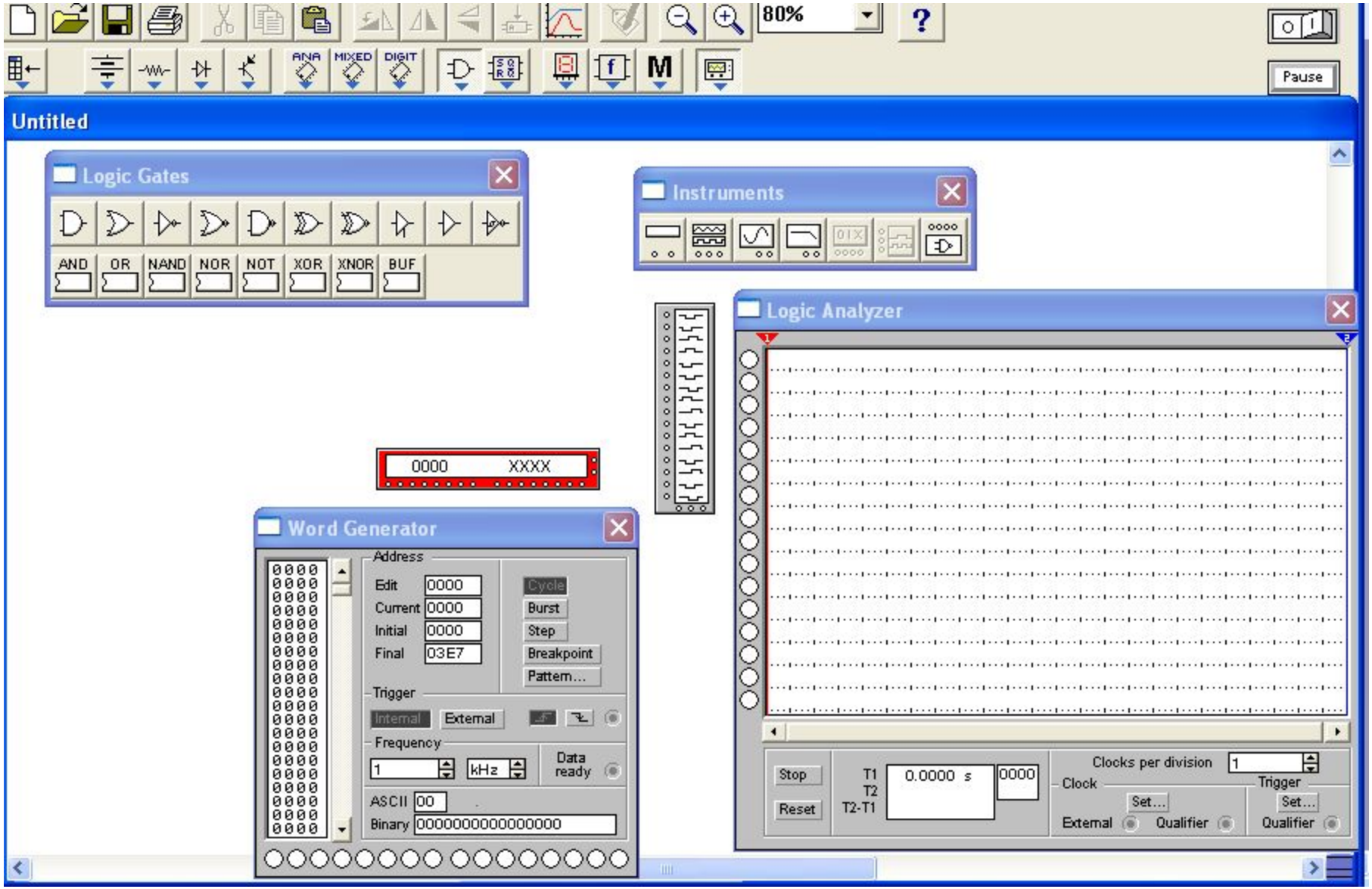
```
var
    A, B: byte;
begin
    A := 11;                                {00001011}
    B := 6;                                 {00000110}
    writeln('A=', A);
    writeln('B=', B);
    writeln('not A = ', not A);             {11110100 = 244}
    writeln('A and B = ', A and B);         {00000010 = 2}
    writeln('A or B = ', A or B);           {00001111 = 15}
    writeln('A xor B = ', A xor B);         {00001101 = 13}
    writeln('A shl 1 = ', A shl 1);         {00010110 = 22}
    writeln('B shr 2 = ', B shr 2);         {00000001 = 1}
end.
```

- **Практическое значение побитовых операций**
- Операция **and** практически всегда используется только для достижения одной из двух целей: проверить наличие установленных в единицу битов или осуществить обнуление некоторых битов.
- Подобная проверка нужна, если число представляет набор признаков с двумя возможными значениями (набор флагов). Так, многие системные ячейки памяти содержат сведения о конфигурации компьютера или его состоянии. При этом установка бита с конкретным номером в 1 трактуется как включение какого-либо режима, а в 0 — выключение этого режима.
- Пусть переменная a имеет тип **byte** и является байтом с восемью флагами. Необходимо проверить состояние бита с номером 5 (биты нумеруются справа налево от 0 до 7). Единица в бите 5 — это пятая степень числа 2, т.е. 32 (00100000). Поэтому, если в пятом бите переменной a стоит единица, то выполняется условие (a and 32) = 32, которое можно проверить в операторе ветвления **if**. Если необходимо проверить состояние нескольких одновременно установленных в единицу битов, то нужно вычислить соответствующее число как сумму степеней числа 2, где показатели степени равны номерам битов, установленных в 1. Например, для битов 5, 2 и 0 имеем 32+4+1=37. Если a имеет среди прочих единицы в битах 5, 2, 0, то выполнится условие (a and 37) = 37.
- Пусть нужно обнулить какой-либо бит в переменной a типа **byte** (например, бит 3). Определим сначала число, содержащее единицы во всех битах, кроме третьего. Максимальное число, которое можно записать в тип **byte**, равняется 255. Чтобы в нем обнулить третий бит, вычтем из этого числа третью степень числа 2 (255-8=247). Если это число логически умножить на a, то его единицы никак не скажутся на состоянии переменной a, а нуль в третьем бите независимо от значения третьего бита переменной a даст в результате 0. Итак, имеем a:= a and (255-8). Аналогично можно обнулить несколько битов.
- Операция **or** применяется при установке в единицу отдельных битов двоичного представления целых чисел. Так, чтобы установить бит 4 переменной a в единицу без изменения остальных битов, следует записать a:= a or 16, где 16 — четвертая степень числа 2. Аналогично устанавливаются в единицу несколько битов.
- Операция **xor** применяется для смены значения бита (или нескольких битов) на противоположное (1 на 0 или 0 на 1). Так, чтобы переключить в противоположное состояние 3-й бит переменной a, следует записать a:= a xor 8, где 8 — третья степень числа 2.

About Electronics Workbench

Electronics Workbench

Version 5.12


80%
Pause
Untitled
Logic Gates
AND
OR
NAND
NOR
NOT
XOR
XNOR
BUF
Instruments
Logic Analyzer
0000 XXXX
Word Generator
Address
Edit 0000
Current 0000
Initial 0000
Final 03E7
Cycle
Burst
Step
Breakpoint
Pattern...
Trigger
Internal
External
Frequency
1
kHz
Data ready
ASCII 00
Binary 0000000000000000
Stop
Reset
T1
T2
T2-T1
0.0000 s
0000
Clocks per division
1
Clock
Set...
External
Qualifier
Trigger
Set...
Qualifier