Лабораторная работа №1

Тема: Представление чисел в различных системах счисления.

Вариант 1 нечетные задания
Вариант 2 четные задания

1) Как записывается число 567_8 в двоичной системе счисления?

   1) 1011101_2 2) 100110111_2 3) 101110111_2 4) 11110111_2

2) Как записывается число $A87_{16}$ в восьмеричной системе счисления?

   1) 435_8 2) 1577_8 3) 5207_8 4) 6400_8

3) Как записывается число 754_8 в шестнадцатеричной системе счисления?

   1) 738_{16} 2) $1A4_{16}$ 3) $1EC_{16}$ 4) $A56_{16}$

4) Для хранения целого числа со знаком используется один байт. Сколько единиц содержит внутреннее представление числа **(-128)**?
5) Для хранения целого числа со знаком используется один байт. Сколько единиц содержит внутреннее представление числа **(-35)**?
6) Дано: $a = 9D_{16}$, $b = 237_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

   1) 10011010_2 2) 10011110_2 3) 10011111_2 4) 11011110_2

7) Дано: $a = F7_{16}$, $b = 371_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

   1) 11111001_2 2) 11011000_2 3) 11110111_2 4) 11111000_2

8) Дано: $a = DD_{16}$, $b = 337_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

   1) 11011010_2 2) 11111110_2 3) 11011110_2 4) 11011111_2

9) Дано: $a = EA_{16}$, $b = 354_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

   1) 11101010_2 2) 11101110_2 3) 11101011_2 4) 11101100_2

10) Дано: $a = D1_{16}$, $b = 333_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

    1) 11100011_2 2) 11011010_2 3) 10101101_2 4) 11011101_2

11) Сколько единиц в двоичной записи числа 64? (без вычислений)
12) Сколько единиц в двоичной записи числа 127? (без вычислений)
13) Сколько значащих нулей в двоичной записи числа 48?
14) Сколько значащих нулей в двоичной записи числа 254?

15) Какое из чисел является наименьшим?

1) $E6_{16}$ 2) 347_8 3) 11100101_2 4) 232

16) Какое из чисел является наибольшим?

1) $9B_{16}$ 2) 234_8 3) 10011010_2 4) 153

17) Дано: $a = A7_{16}$, $b = 251_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 10101100_2 2) 10101010_2 3) 10101011_2 4) 10101000_2

18) Дано: $a = DD_{16}$, $b = 337_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 11011010_2 2) 11111110_2 3) 11011111_2 4) 11011110_2

19) Дано: $a = 222_8$, $b = 94_{16}$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 10001010_2 2) 10001110_2 3) 10010011_2 4) 10001100_2

20) Дано: $a = EA_{16}$, $b = 354_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 11101010_2 2) 11101110_2 3) 11101100_2 4) 11101011_2

21) Дано: $a = AA_{16}$, $b = 255_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 10101010_2 2) 10111100_2 3) 10100011_2 4) 10101100_2

22) Сколько единиц в двоичной записи числа 173?

23) Дано: $a = 70_{10}$, $b = 40_{16}$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $b < C < a$?

1) 1000000_2 2) 1000110_2 3) 1000101_2 4) 1000111_2

24) Дано: $a = 91_{16}$, $b = 352_8$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $a < C < b$?

1) 10001001_2 2) 10001100_2 3) 11010111_2 4) 11111000_2

25) Дано: $a = 11100110_2$, $b = 271_8$. Какое из чисел С, записанных в шестнадцатеричной системе счисления, удовлетворяет неравенству $a > C > b$?

1) AA_{16} 2) $B8_{16}$ 3) $D6_{16}$ 4) $F0_{16}$

26) Дано: $x = 1F4_{16}$, $y = 701_8$. Какое из чисел Z, записанных в двоичной системе счисления, удовлетворяет неравенству $y < Z < x$?

1) 111111001_2 2) 111100111_2 3) 110111100_2 4) 110110111_2

27) Дано: $a = 10110111_2$, $b = A6_{16}$. Какое из чисел С, записанных в двоичной системе счисления, удовлетворяет неравенству $b < C < a$?

1) 10111010_2 2) 10101010_2 3) 101010100_2 4) 10100010_2

28) Сколько единиц в двоичной записи десятичного числа 513?

29) Сколько значащих нулей в двоичной записи десятичного числа 497?

30) Для каждого из перечисленных ниже десятичных чисел построили двоичную запись. Укажите число, двоичная запись которого содержит ровно 3 единицы.

1) 1 2) 11 3) 3 4) 33

31) Для каждого из перечисленных ниже десятичных чисел построили двоичную запись. Укажите число, двоичная запись которого содержит ровно 2 единицы.

1) 7 2) 11 3) 12 4) 15

32) Даны 4 числа, они записаны с использованием различных систем счисления. Укажите среди этих чисел то, в двоичной записи которого содержится ровно 5 единиц. Если таких чисел несколько, укажите наибольшее из них.

1) $31_{10} * 8_{10} + 1_{10}$ 2) $F0_{16} + 1_{10}$ 3) 351_8 4) 11100011_2

33) Укажите наименьшее четырёхзначное восьмеричное число, двоичная запись которого содержит 6 единиц. В ответе запишите только само восьмеричное число, основание системы счисления указывать не нужно.

34) Укажите набольшее четырёхзначное восьмеричное число, двоичная запись которого содержит 4 единицы. В ответе запишите только само восьмеричное число, основание системы счисления указывать не нужно.

35) Укажите наименьшее четырёхзначное шестнадцатеричное число, двоичная запись которого содержит ровно 5 значащих нулей. В ответе запишите только само шестнадцатеричное число, основание системы счисления указывать не нужно.

36) Укажите наибольшее четырёхзначное шестнадцатеричное число, двоичная запись которого содержит ровно 6 значащих нулей. В ответе запишите только само шестнадцатеричное число, основание системы счисления указывать не нужно.

37) Вычислите: $10101010_2 - 252_8 + 7_{16}$. Ответ запишите в десятичной системе счисления.

38) Вычислите: $10101011_2 - 253_8 + 6_{16}$. Ответ запишите в десятичной системе счисления.

39) Определите количество натуральных чисел, удовлетворяющих неравенству: $(170_8 + FE_{16}) \le x \le (200_8 + 11111111_2)$.

40) Определите количество натуральных чисел, удовлетворяющих неравенству: $(96_{16} + 18_{16}) < x < (240_8 + 33_8)$.

41) Укажите наибольшее четырёхзначное восьмеричное число, четверичная запись которого содержит ровно 2 тройки, нестоящие рядом. В ответе запишите только само восьмеричное число, основание системы счисления указывать не нужно.

42) Определите количество натуральных чисел, кратных основанию четверичной системы счисления и удовлетворяющих неравенству: $734_8 \le x < 1E4_{16}$

43) Задан отрезок [*a, b*]. Число *a* – наименьшее число, восьмеричная запись которого содержит ровно 3 символа, один из которых – 3. Число *b* – наименьшее число, шестнадцатеричная запись которого содержит ровно 3 символа, один из которых – F. Определите количество натуральных чисел на этом отрезке (включая его концы).
44) Задан отрезок [*a, b*]. Число *a* – наибольшее число, восьмеричная запись которого содержит ровно 2 символа, один из которых – 6. Число *b* – наибольшее число, шестнадцатеричная запись которого содержит ровно 2 символа, один из которых – C. Определите длину этого отрезка (ответ запишите в десятичной системе).

Лабораторная работа №2

Тема: Элементарные логические функции. Формы предоставления логических функций.

- условные обозначения логических операций

  ¬ A, $\overline{A}$ не A (отрицание, инверсия)

  A ∧ B, $A \cdot B$ A и B (логическое умножение, конъюнкция)

  A ∨ B, $A + B$ A или B (логическое сложение, дизъюнкция)

  A → B импликация (следование)

  A ≡ B эквивалентность (равносильность)
- операцию «импликация» можно выразить через «ИЛИ» и «НЕ»:

  A → B = ¬ A ∨ B или в других обозначениях **A → B =** $\overline{A} + B$
- иногда для упрощения выражений полезны формулы де Моргана:

  ¬ (A ∧ B) = ¬ A ∨ ¬ B $\overline{A \cdot B} = \overline{A} + \overline{B}$

  ¬ (A ∨ B) = ¬ A ∧ ¬ B $\overline{A + B} = \overline{A} \cdot \overline{B}$
- если в выражении нет скобок, сначала выполняются все операции «НЕ», затем – «И», затем – «ИЛИ», «импликация», и самая последняя – «эквивалентность»
- таблица истинности выражения определяет его значения при всех возможных комбинациях исходных данных
- если известна только часть таблицы истинности, соответствующее логическое выражение однозначно определить нельзя, поскольку частичной таблице могут соответствовать несколько *разных* логических выражений (не совпадающих для других вариантов входных данных);
- количество *разных* логических выражений, удовлетворяющих неполной таблице истинности, равно 2^k, где k – число *отсутствующих* строк; например, полная таблица истинности выражения с тремя переменными содержит 2^3=8 строчек, если заданы только 6 из них, то можно найти $2^{8-6}=2^2$=4 *разных* логических выражения, удовлетворяющие этим 6 строчкам (но отличающиеся в двух оставшихся)
- логическая сумма A + B + C + … равна 0 (выражение ложно) тогда и только тогда, когда все слагаемые одновременно равны нулю, а в остальных случаях равна 1 (выражение истинно)

- логическое произведение A · B · C · … равно 1 (выражение истинно) тогда и только тогда, когда все сомножители одновременно равны единице, а в остальных случаях равно 0 (выражение ложно)
- логическое следование (импликация) A→B равна 0 тогда и только тогда, когда A (посылка) истинна, а B (следствие) ложно
- эквивалентность A≡B равна 1 тогда и только тогда, когда оба значения одновременно равны 0 или одновременно равны 1

45) Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F (см. таблицу справа). Какое выражение соответствует F?

1) **X ∨ ¬Y ∨ Z** 2) **X ∧ Y ∧ Z** 3) **X ∧ Y ∧ ¬Z** **4**) **¬X ∨ Y ∨ ¬Z**

X	*Y*	*Z*	*F*
0	1	0	0
1	1	0	1
1	0	1	0

46) Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F (см. таблицу справа). Какое выражение соответствует F?

1) **¬X ∨ Y ∨ ¬Z** 2) **X ∧ Y ∧ ¬Z** 3) **¬X ∧ ¬Y ∧ Z** **4**) **X ∨ ¬Y ∨ Z**

X	*Y*	*Z*	*F*
0	0	0	1
0	0	1	0
0	1	0	0

47) Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F (см. таблицу справа). Какое выражение соответствует F?

1) **X ∧ Y ∧ Z** 2) **¬X ∧ ¬Y ∧ Z** 3) **X ∧ Y ∧ ¬Z** **4**) **¬X ∧ ¬Y ∧ ¬Z**

48) Дан фрагмент таблицы истинности выражения F.

x1	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	*F*
0	1	0	1	1	1	1	1
1	0	1	0	1	1	0	0
0	1	0	1	1	0	1	1

Какое выражение соответствует F?

1) **x1 ∧ ¬x2 ∧ x3 ∧ ¬x4 ∧ x5 ∧ x6 ∧ ¬x7**
2) **¬x1 ∨ x2 ∨ ¬x3 ∨ x4 ∨ ¬x5 ∨ ¬x6 ∨ x7**
3) **¬x1 ∧ x2 ∧ ¬x3 ∧ x4 ∧ x5 ∧ x6 ∧ x7**
4) **x1 ∨ ¬x2 ∨ x3 ∨ ¬x4 ∨ ¬x5 ∨ ¬x6 ∨ ¬x7**

49) Логическая функция *F* задаётся выражением $((y \vee z) \to x) \vee (x \equiv z)$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции *F*, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции *F* соответствует каждая из переменных *x*, *y*, *z*.

?	**?**	**?**	**F**
0		**0**	**0**
		0	**0**

В ответе напишите буквы *x*, *y*, *z* в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

50) Логическая функция F задаётся выражением $(y \to (z \wedge x)) \vee (x \equiv y)$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z.

?	?	?	F
0		0	0
		1	0

В ответе напишите буквы x, y, z в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

51) Логическая функция F задаётся выражением $(x \vee y) \wedge \neg z \wedge \neg(z \equiv x)$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z.

?	?	?	F
0		0	1
		0	1

В ответе напишите буквы x, y, z в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

52) Логическая функция F задаётся выражением $(y \to x) \wedge z \wedge \neg(z \equiv y)$. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z.

?	?	?	F
0		0	1
		1	1

В ответе напишите буквы x, y, z в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

53) Логическая функция F задаётся выражением

$$((x \to y) \wedge (y \to w)) \vee ((z \equiv (x \vee y)).$$

На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w.

?	?	?	?	F
1			1	0
1				0
	1		1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

54) Логическая функция F задаётся выражением (¬x □ □y □ z) □ w. На рисунке приведён частично заполненный фрагмент таблицы истинности функции F,

содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции *F* соответствует каждая из переменных *x, y, z,w*.

?	?	?	?	F
	0			**1**
			0	**1**
0	**0**			**1**
0	**0**			**1**

В ответе напишите буквы *x, y, z, w* в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Лабораторная работа №3

Тема: Назначение типовых элементов. Основные логические элементы

Программа Electronics Workbench (EWB) создана в 1989 г. канадской фирмой Interactive Image Technologies для схемотехнического моделирования аналоговых и цифровых радиоэлектронных устройств различного назначения. Программа является англоязычной, но несмотря на это в ней легко можно создавать различные логические конструкции и осуществлять проверку решений.

Используемые тип файлов - .ewb. Эти файлы обычно находятся в каталоге **Samples.** Например, файл Rs-ff.ewb содержит схему RS-триггера на логических элементах ИЛИ-НЕ.

Для реализации алгоритма работы логических схем в программе Electronics Workbench используется математический аппарат алгебры логики. При этом Булева агебра оперирует двумя понятиями: высказывание истинно (логическая "1") или высказывание ложно (логический "0"). Для высказываний определены базовые операции: отрицание (инверсия) обозначаемой чертой над соответствующим символом, логическое сложение (дизъюнкция), обозначаемой +, и логическое умножение (конъюнкции), обозначаемой знаком точкой. Эквиваленция, импликация, исключающее или, отрицание дизъюнкции, отрицание конъюнкции, все эти операции можно вывести с помощью тождественных преобразований из базовых.

На схеме рис. 1 входы логического элемента **"ИЛИ-НЕ"** подключены к генератору слов, формирующего последовательность двоичных чисел 00, 01, 10 и 11. Правый (младший) двоичный разряд каждого числа соответствует логической переменной X1, левый (старший)– логической переменной X2. К входам логического элемента также подключены **логические пробники**, которые загораются красным светом при поступлении на этот вход логической "1". Выход логического элемента подключен к логическому пробнику, который загорается красным светом при появлении на выходе логической "1".

Рассмотрим пример построения схемы исследования логического элемента "ИЛИ-НЕ"

Интерфейс программы Electronics Workbench представлен на рисунке 1. На новом листе размещаем диалоговые окна, необходимые для построения логической схемы.

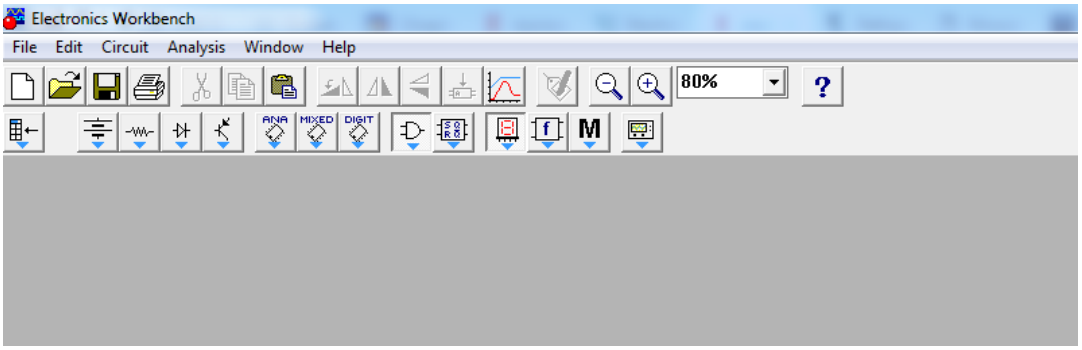


Рисунок 1. Интерфейс Electronics Workbench.

Разместим панель логических элементов нажатием на кнопку рис. 2, внешний вид которой представлен на рисунке 3.

Рисунок 2. Логические элементы

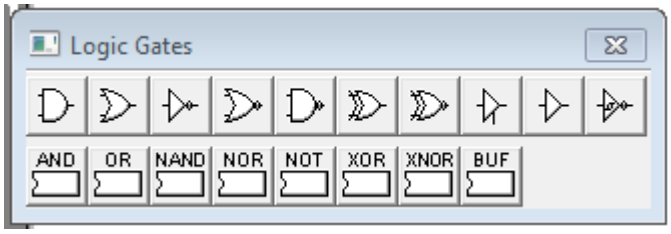


Рисунок 3. Панель логических элементов

На новом листе с Логической панели возьмем первый элемент Логическое умножение (конъюнкцию).

Затем разместим панель Индикаторы рис. 4

Рисунок 4. Индикаторы

Из появившегося окна последовательно выбираем три пиктограммы логических пробников (рис. 5).

Рисунок 5. Логический пробник

На панели инструментов вбираем команду Поворот (рис. 6). Разворачиваем логические пробники последовательно, таким образом, чтобы они размещались вертикально. Для этого на панели функций воспользуемся кнопкой поворота .

Рисунок 6 . Поворот

Следующая панель необходимая для построения схемы называется Инструменты (рис.7), Генератор слов

Рисунок 7. Инструменты

Размещаем на схеме Генератор слов (рис. 8)

Рисунок 8. Генератор слов.

Используя метод буксировки пиктограмм элементов так, как показано на рисунке 8 соединяем элементы и получаем логическую конструкции конъюнкции.

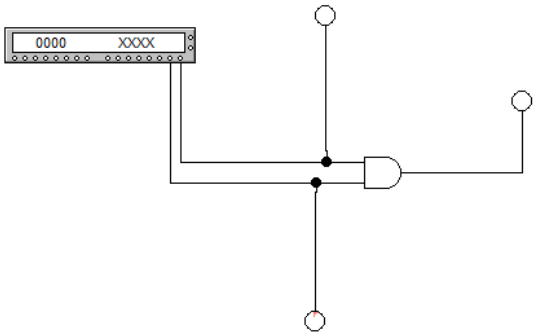


Рисунок 8. Логическая схема с операцией конъюнкция

Двойным щелчком кнопки мыши нажимаем на пиктограмму Г**енератор слов**.

В левой части панели **генератора слов** отображаются кодовые комбинации в шестнадцатеричном коде, а в нижней части размещен в двоичном коде (рис. 9).

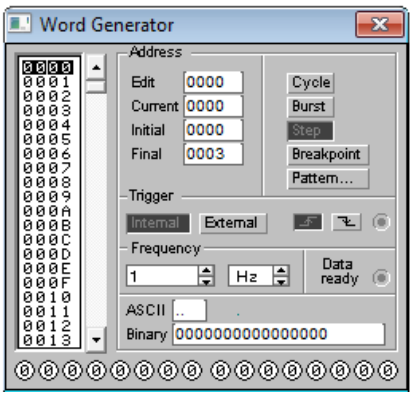


Рисунок 9. Параметры генератора.

Начальное значение остается неизменным в ячейки EDIT – 0000, изменяем конечное значение на 0003. Кнопка **Pattern** позволяет выбирать различные опции, например осуществление итерации сверху вниз командой **Up counter** и нажатие по кнопке **Accept** для применения данного действия.

В окне **Frequency** изменяем частоту формирования кодовых комбинаций равной 1 Гц.

Последовательности двоичных чисел 00, 01, 10 и 11 соответствует в шестнадцатеричном коде - 0, 1, 2, 3. Запрограммируем генератор на периодическое формирование указанной последовательности чисел.

Для запуска работы логической схемы нажимаем на кнопку **Cycle.**

Кнопка **Step** позволяет перемещаться между значениями с удобным интервалом, для фиксации результатов. На рисунке 10 приведена работы схемы при значении 11 в двоичной системе счисления, т.е. на вход подается две единицы о чем свидетельствуют две светящиеся лампочки (, и результатом является единица.

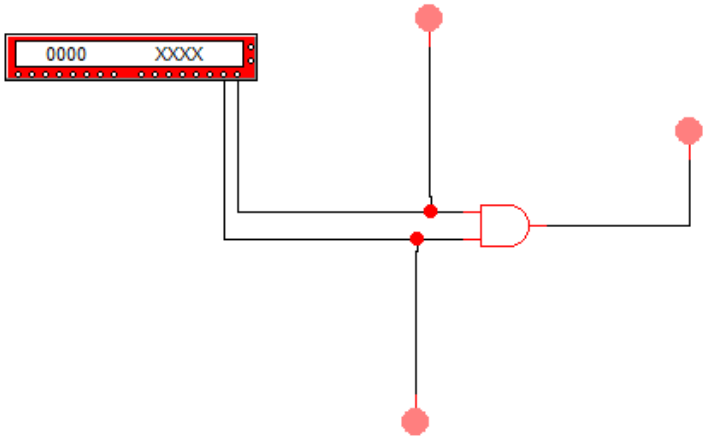


Рисунок 10. Работы логической конъюнкции при значении 11 в двоичной системе счисления.

Иллюстрация возможностей САПР Electronics Workbench при проектировании логических схем по таблице истинности.

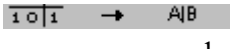
1. Создайте новый файл.
2. Щелкните по кнопке

панели библиотек компонентов и контрольно-измерительных приборов. Из появившегося окна индикаторов вытащите пиктограмму **Логического Преобразователя** (**Logic converter**)

.

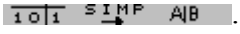
3. Двойным щелчком на пиктограмме откройте лицевую панель. На панели отображены клеммы-индикаторы входов **A**, **B**, **C**, **D**, **E** …. и одного выхода – последний столбик справа **Out**. Последовательно щелкните по четырем круглым кнопкам на верхней части панели логического преобразователя, соответствующим символам **A**, **B**, **C** и **D**. На левой панели должны появиться числа, соответствующие входным переменным логической схемы. В столбце **Out**, соответствующему выходному значению ПФ, находятся нули.

4. Выделяя нужные позиции в столбце **Out**, введите при помощи клавиатуры единицы в соответствии с **Таблицей 1**. В результате получим таблицу истинности рассмотренного выше примера. Логический преобразователь подготовлен к работе.

5. Нажмите при помощи мышки кнопку 

. В результате в нижней строке логического преобразователя появится логическая функция, соответствующая заданной таблице истинности. Занесите эту функцию в отчет.

В качестве символа отрицания в программе **Electronics Workbench** принят апостроф (').

6. Нажмите при помощи мышки кнопку  . В результате в нижней строке логического преобразователя появится минимизированная логическая функция, соответствующая заданной таблице истинности. Занесите эту функцию в отчет.

7. Нажмите при помощи мышки кнопку  . В результате в окне программы **Electronics Workbench** появится логическая схема, соответствующая заданной ПФ. Результат покажите преподавателю.

Иллюстрация возможностей САПР при проектировании простых логических схем по логической функции

Переключательная функция (ПФ) может быть задана в дизъюнктивной нормальной форме (ДНФ) или в конъюнктивной нормальной форме (КНФ). **Логический Преобразователь** (**Logic converter**) программы **Electronics Workbench** работает только с ПФ, заданной в дизъюнктивной нормальной форме.

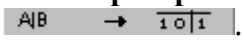
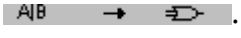
Задача исследования:

- Преобразовать ПФ

$$F = A'BD + C'D + ACD' \qquad (1)$$

в форму таблицы истинности и синтезировать по ней логическую схему.

Порядок выполнения:

1. Создайте новый файл. Вытащите пиктограмму **Логического Преобразователя** (**Logic converter**). Откройте лицевую панель.
2. Введите при помощи клавиатуры в нижнюю строку **Логического преобразователя** выражение (1) переключательной функции и нажмите на кнопку  .
3. Для получения функциональной схемы в базисе НЕ. 2И, 2ИЛИ, соответствующей заданной ДНФ, нажмите на кнопку  .

Покажите полученную таблицу истинности и логическую схему преподавателю.

Самостоятельная работа.

Задание № 1.

- Проведите синтез логических схем в соответствии с заданными вариантами таблиц истинности.
- Проведите исследование полученной логической схемы (подключите **генератор слов** ко входу схемы и логический пробник к ее выходу, запрограммируйте генератор слов, проверьте правильность работы схемы).

Примечание: Младший двоичный разряд **генератора слов** подключайте ко входу **A** схемы, следующий ко входу **B** и т.д.

Вариант 1

N	1	2	3	4	6	7	8	9	11	12
F	1	1	1	0	0	0	1	1	1	0

Вариант 2

N	0	2	3	5	6	7	8	9	13	15
F	0	0	1	1	0	0	1	1	1	0

Вариант 3

N	1	2	3	4	6	7	9	12	13	14
F	0	0	0	1	1	1	0	1	0	1

Вариант 4

N	0	2	3	4	5	6	7	8	10	13
F	0	1	0	1	1	0	1	0	0	0

Примечание: Переключающие функции в этих таблицах приведены в сокращенной форме, значения выходной переменной, соответствующей пропущенным десятичным числам, например, N = 0, 1, 2, 3, 5, 10 принять равным 0.

Результаты покажите преподавателю.

Задание № 2.

- минимизируйте выражение для заданной ПФ.
- Проведите синтез логических схем в соответствии с заданными вариантами ПФ в виде ДНФ.

Вариант 1 **F= AB’C + A’BD’ + BC’D**

Вариант 2 **F= AB’CD’ + A’BC + BC’D**

Вариант 3 **F= ACD’ + A’BCD’ + BC’D**

Вариант 4 **F= B’CD’ + A’BC + ABC’D**

Результаты синтеза покажите преподавателю.

Лабораторная работа №4-5.

Тема: Типовые элементы вычислительной техники. Исследование дешифраторов

Цель:

Овладение практическими навыками исследования дешифраторов с использованием средств САПР Electronics Workbench.

Результат обучения:

После успешного завершения занятия пользователь должен:

- Знать назначение и функциональные возможности дешифратора;
- Уметь создавать модели логических схем путем графической сборки схемы соединений базовых компонентов;
- Уметь находить переключательную функцию по функциональной схеме средствами САПР Electronics Workbench.

Используемые программы:

Electronics Workbench в. 5.0

I. Краткое описание характеристик дешифратора

Дешифратор является логической интегральной схемой (ИС), содержащей n входов и 2^n выходов. На входы дешифратора подается двоичное число, в результате чего один из его выходов изменяет состояние. Обычно при подаче на вход дешифратора какого-либо двоичного числа (например, число $3_{10} = 011_2$) изменяет состояние его соответствующий (третий по порядку) выход Y3. Промышленность выпускает ИС дешифраторов с 2, 3 или с 4 входами. На рис. 1 приведен дешифратор с тремя входами C, B, A и соответственно с 8 выходами Y0, Y1, Y2, Y3, Y4, Y5, Y6 и Y7.

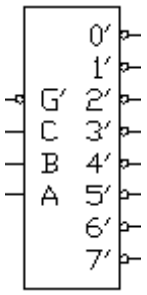


Рис 1. Дешифратор 3 x 8 (вход – 3 разряда, выходов – 8).

Во многих сериях микросхем активным состоянием выхода является низкий уровень – состояние логического "0". Таким образом при подаче на вход дешифратора (см. рис. 1) какого-либо двоичного числа (например, 011) соответствующий (третий по порядку) выход – 2’ будет находиться в состоянии лог. "0" (низкий уровень напряжения) а остальные выходы – в состоянии лог. "1" (высокий уровень напряжения).

ИС дешифратора обычно имеет дополнительный вход управления G (прямой или инверсный), который называется входом выбора микросхемы. Только при подаче сигнала на этот вход (при выборе данной ИС) дешифратор будет работать. У дешифраторов с инверсным входом управления активным уровнем является уровень логического нуля, а с прямым входом управления - уровень логической единицы. Часто ИС дешифратора имеет несколько управляющих входов. Тогда его можно использовать в качестве демультиплексора, логической схемы, подключающей сигнал ("1" или "0") с сигнального входа (второго входа управления) к одному из выходов Y0, Y1, Y2, Y3, Y4, Y5, Y6 и Y7.

1.1. Исследование дешифратора

Схема исследования дешифратора, представлена на рис. 2.

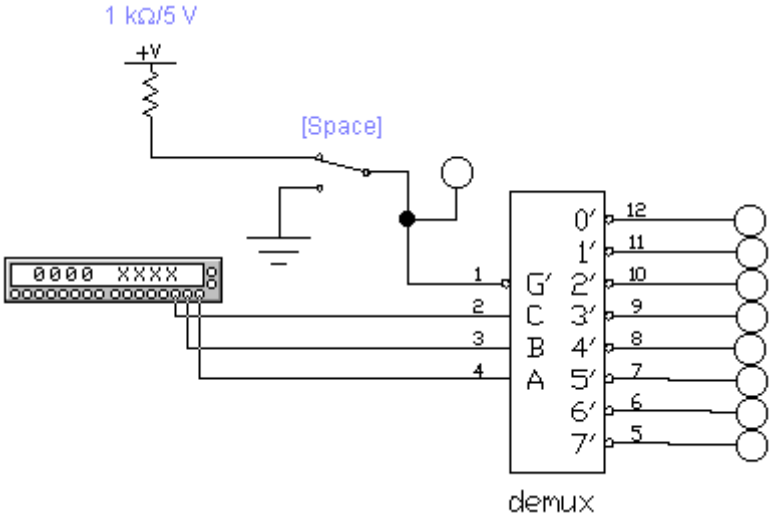


Рис. 2

На схеме рис. 2 входы C, B, A дешифратора подключены к выходам генератора слов. Выходы дешифратора Y0, Y1, Y2, Y3, Y4, Y5 Y6 и Y7 подключены к логическим пробникам. Вход выбора G дешифратора подключен к источнику питания 5В через контакты, управляемые клавишей "**Space - пробел**". Если контакты "**Space**" замкнуты, то на инвертирующий вход G подается напряжение высокого уровня (Лог. 1) и дешифратор не работает.

Построение схемы исследования дешифратора

Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**.

Построение схемы рис. 1 произведем в два этапа: сначала разместим как показано на рис. 1 пиктограммы элементов, а затем последовательно соединим их.

1. Щелкните по кнопке

панели библиотек компонентов и контрольно-измерительных приборов. Из появившегося окна цифровых микросхем вытащите пиктограмму 

. В появившемся диалоговом окне выберите **Generic 8-to-1 DEMUX** и нажмите на кнопку **Accept.**

2. Последовательно вытащите пиктограммы заземления и (из группы основных элементов) переключателя, источника питания 5В
3. Щелкните по кнопке

Из появившегося окна последовательно вытащите пиктограммы логических пробников  .

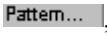
4. Разверните логические пробники, так как показано на рис. 2. Для этого на панели функций воспользуйтесь кнопкой поворота


.

5. Щелкните по кнопке

панели библиотек компонентов и контрольно-измерительных приборов. Из появившегося окна индикаторов вытащите пиктограмму **генератора слов**


.

6. Расположите методом буксировки пиктограммы элементов так, как показано на рис. 2 и соедините элементы согласно рисунку.
7. Двойным щелчком кнопки мыши откройте лицевую панель **генератора слов**.
   В левой части панели **генератора слов** отображаются кодовые комбинации в шестнадцатеричном коде, а в нижней части - в двоичном.
8. Заполним окно шестнадцатеричного кода кодовыми комбинациями, начиная с 0 в верхней нулевой ячейке и далее с прибавлением 1 в каждой последующей ячейке. С этой целью щелкните по кнопке 

, в появившемся окне предустановок включите опцию **Up counter** и щелкните по кнопке **Accept**.

   Запрограммируем генератор на периодическое формирование последовательности двоичных чисел, соответствующих 0,1,3,3,4,5,6,7 в шестнадцатеричном коде.

9. Наберите в окне **Final** число **0007** и щелкните на кнопке **Step**.
10. Запустите процесс моделирования при помощи выключателя. Щелкая по кнопке **Step**, наблюдайте, при каких сочетаниях входных сигналов на выходе логического элемента появится "1". Остановите процесс моделирования при помощи выключателя. Переключите контакты "**Space**", нажимая на клавишу "**Space**". Запустите процесс моделирования при помощи выключателя. Щелкая по кнопке **Step**, заполните в Отчете таблицу истинности.
11. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_11_01** .

1.2. Исследование схемы каскадирования дешифраторов

Для увеличения разрядности дешифратора применяется каскадирование. Для этого входное двоичное число (код) на входе делится на части, при этом разрядность младших разрядов соответствует разрядности имеющейся ИС дешифратора. Оставшиеся старшие

разряды входного числа поступают на комбинационную схему, выбирающую один из дешифраторов. Для этого подается сигнал выбора на вход G соответствующей ИС.

На рис. 3 изображена схема исследования каскадирования двух дешифраторов типа 3 х 8.

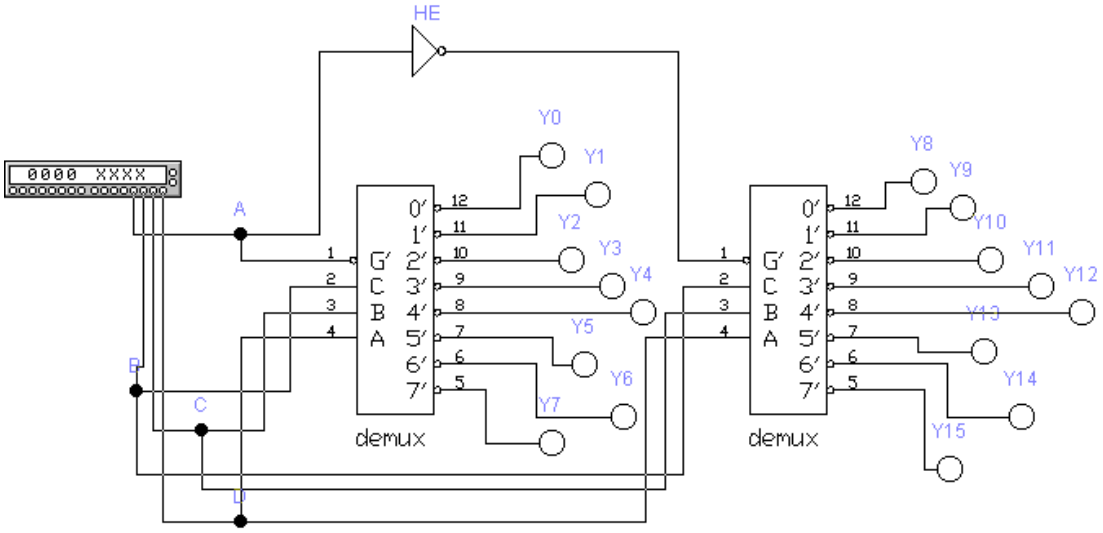


Рис. 3

Входной четырехразрядный код, формируемый генератором слов делится на две части: три младших разряда **D**, **C**, **B** подаются на входы двух дешифраторов параллельно. Старший разряд **A** подается на вход выбора G первого дешифратор и через инвертор – на вход G второго. В результате, если **A** = 0, то работает первый дешифратор, а если **A** = 1, то выбирается второй. В результате получаем дешифратор 4 х 16. Например, при подаче на вход схемы числа $11_{10} = 1011_2$ выходной сигнал появится на выходе Y11 второго дешифратора.

Построение схемы исследования дешифратора

1. Создайте новый файл.
2. Щелкните по кнопке

панели библиотек компонентов и контрольно-измерительных приборов. Из появившегося окна логических элементов вытащите пиктограмму логического элемента **NOT** ("НЕ").

.

3. Последовательно вытащите пиктограммы остальных элементов схемы (располагая их, так как показано на рис. 3). Соедините элементы.
4. Двойным щелчком кнопки мыши откройте лицевую панель генератора слов.
5. Заполним окно шестнадцатеричного кода кодовыми комбинациями, начиная с 0 в верхней нулевой ячейке и далее с прибавлением 1 в каждой последующей ячейке. С этой целью щелкните по кнопке Pattern..., в появившемся окне предустановок включите опцию Up counter и щелкните по кнопке Accept.
6. В окне Frequency установите частоту формирования кодовых комбинаций равной 1 Гц.

7. Запрограммируем генератор на периодическое формирование указанной последовательности чисел. Для этого наберите в окне **Final** число **000F** и щелкните на кнопке **Cycle**.
8. Запустите процесс моделирования при помощи выключателя. Наблюдайте за работой схемы, результаты моделирования покажите преподавателю.
9. Остановите процесс моделирования при помощи выключателя. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_18_02 .**

II. Самостоятельная работа. Исследование логической схемы на базе дешифратора.

Рекомендуемое время

40 минут

Задание.

- Собрать схему дешифратора с логической схемой на выходе, приведенную на рис. 4, 5, 6 согласно варианту. Сохранить файл в папке с вашей **Фамилией** под именем **Zan_18_03**.
- Получите выражение минимизированной логической функции схемы, реализованной соединением дешифратора с логической схемой на выходе. Полученное выражение занести в отчет.
- Экспериментальным путем получить таблицу истинности. Результаты моделирования покажите преподавателю.

Примечание:

1. Вытащите пиктограмму логического преобразователя, подключите входы логического преобразователя **A**, **B**, **C** ко входам дешифратора **C, B**, **A** соответственно и выход логического преобразователя к выходу логической схемы.
2. Откройте лицевую панель логического преобразователя. Нажатием при помощи мышки на кнопку  получите таблицу истинности, а затем при помощи кнопок  и  получите выражение логической функции, реализованной схемой.
3. Отключите логический преобразователь, подключить генератор слов ко входу схемы, а логический пробник к выходу.

   Вариант 1

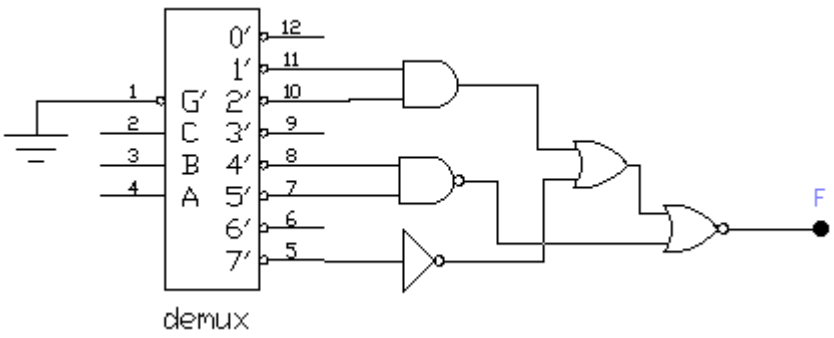


Рис. 4

Вариант 2

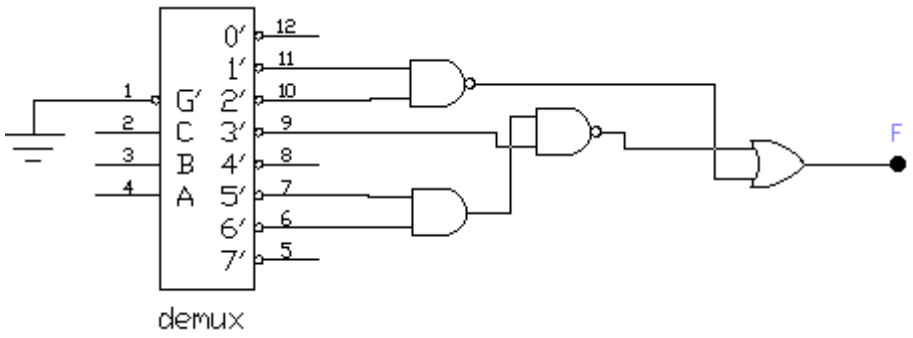


Рис. 5

Вариант 3

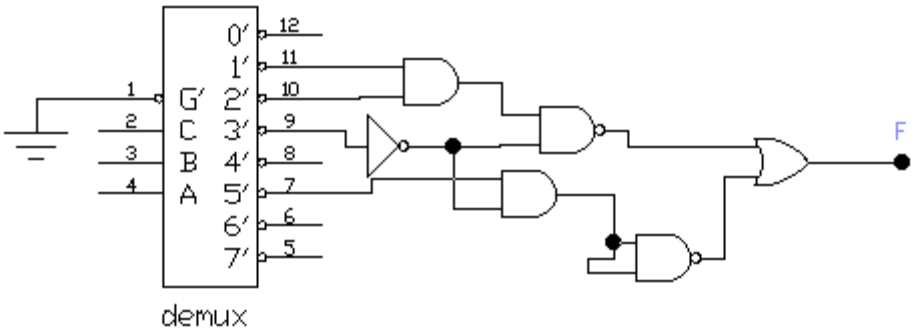


Рис. 6

Лабораторная работа №6-7

Тема: Типовые элементы вычислительной техники. Исследование характеристик триггеров

Цель: Овладение практическими навыками исследования характеристик триггеров D, RS, JK, T типа средствами САПР Electronics Workbench.

Результат обучения:

После успешного завершения занятия пользователь должен:

- Знать назначение и функциональные возможности триггеров D, RS, JK, T типа.
- Уметь проводить анализ этих элементов средствами САПР **Electronics Workbench**.

Используемые программы:

Electronics Workbench в. 5.0

I. Исследование характеристик триггеров

1.1. Краткое описание назначения и основных функциональных возможностей триггера

Триггер – основной элемент цифровых узлов последовательного или накапливающего типа. В рассмотренных на предыдущих занятиях комбинационных схемах выходное состояние Y определялось только текущим состоянием входов – X_1, X_2,

…, X_N. В цифровых узлах накапливающего типа выходное состояние определяется также и внутренними состояниями системы Q_1, Q_2, …., Q_M.

Состояние схемы комбинационного типа описывается логической функцией $Y = F(X_1, X_2, …, X_N)$. Состояние схемы накапливающего типа описывается логической функцией $Y = F(X_1, X_2, …, X_N , Q_1, Q_2, …., Q_M)$. Это означает, что выходное состояние схемы накапливающего типа зависит не только от текущего входного состояния, но и от предшествующих входных состояний, хранимых в памяти устройства.

Триггер – логический элемент с двумя устойчивыми состояниями. Обычно одно из устойчивых состояний обозначают Q = 1, второе – $\overline{Q} = 0$. Это означает, что триггер является элементарной ячейкой памяти, хранящей 1 бит информации. *По логике работы* триггеры разделяются на RS, D, T и JK.

1.2. Исследование характеристик триггера типа RS средствами САПР

Триггер типа RS имеет два входа (R и S) и два выхода (Q и $\overline{Q}$). Условное графическое обозначение триггера типа RS приведено на рис. 1.

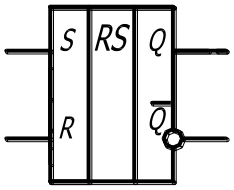


Рис. 1. Условно-графическое изображение RS триггера.

Если на вход S подать лог. "1", то выход Q будет находиться в состоянии лог. "1", а выход Q – в состоянии лог. "0". Если на вход R подать лог. "1", то выход Q будет находиться в состоянии лог. "0", а выход $\overline{Q}$ – в состоянии лог. "1". Это два устойчивых состояния триггера. Условия перехода триггера из одного устойчивого состояния в другое обычно описывается при помощи двух таблиц – таблицы переходов и таблицы функционирования. Таблица функционирования триггера - представленный в табличном виде закон переключения триггера, определяющий реакцию триггера на входные воздействия (в качестве набора переменных- аргументов заданы текущее состояние триггера Q_t и сигналы на всех входах).

Схема исследования триггера типа RS, представлена на рис. 2.

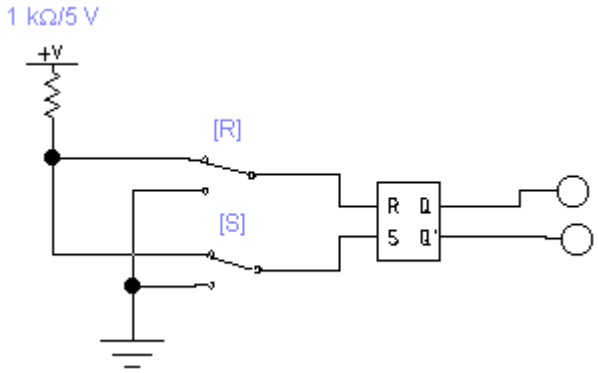


Рис. 2 Схема исследования RS триггера

Входы триггера R и S подключены при помощи переключателей к напряжению высокого уровня (+ 5 В), либо низкого уровня (0 В, "земля"). Переключатели управляются клавишами R и S на клавиатуре (латинский регистр).

Построение схемы исследования триггера типа RS

Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**.

1. Щелкните по кнопке  панели библиотек компонентов и контрольно-измерительных приборов. Из появившегося окна цифровых микросхем вытащите пиктограмму  .
2. Последовательно вытащите пиктограммы заземления и (из группы основных элементов ) переключателей, источника питания 5В.
3. Щелкните по кнопке  Из появившегося окна последовательно вытащите пиктограммы логических пробников  .
4. Разверните элементы, так как показано на рис. 2. Для этого на панели функций воспользуйтесь кнопками поворота

 и  .

5. По умолчанию переключатель управляется нажатием клавиши **Space**, назначим клавишам R и S функцию управления переключателем. С этой целью последовательно откройте диалоговые окна для задания их параметров и перейдите на вкладку **Value**. В поле **Key** введите соответствующее имя клавиши.
6. Соедините элементы согласно рис. 2.

Задача исследования №1: экспериментальным путем получить таблицу перехода и занести ее в Отчет.

1. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Последовательно подавайте на схему следующие входные сигналы:
   1. R = 0, S = 1 - триггер установится в состояние Q = 1;
   2. R = 0, S = 0 - триггер сохраняет свое состояние Q = Q_t;
   3. R = 1, S = 0 - триггер установится в состояние Q =0;
   4. R = 1, S = 1 - триггер находится в неопределенном состоянии Q = -,
2. Заполните таблицу в отчете. Остановите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

Задача исследования №2: экспериментальным путем получить таблицу функционирования и занести ее в Отчет.

1. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.
2. Установите триггер в состояние Q =0 (подайте сигналы R = 1, S = 0), что соответствует начальному состоянию триггера Qt =0, приведенному в первой строке таблицы.
3. Подайте на схему сигнал R = 0. Убедитесь – состояние триггера сохранится Qt+1 = Qt =0. Т.о. состояние триггера в данном случае безразлично к значению R.
4. Последовательно подайте на схему сигналы:
   a. S = 1 Убедитесь - триггер перейдет из начального состояния $Q_t = 0$ в состояние $Q_{t+1} = 1$;
   b. R = 1, S = 0 Убедитесь - триггер перейдет из начального состояния $Q_t = 1$ в состояние $Q_{t+1} =0$;
   c. Установите триггер в состояние Q =1 (R = 0, S = 1). Убедитесь – состояние триггера в этом случае безразлично к значению сигнала по S. Остановите процесс моделирования.

1.3. Исследование характеристик триггера типа JK средствами САПР.

Триггер типа JK является универсальным и имеет большие по сравнению с RS триггером функциональные возможности. Условное графическое обозначение триггера приведено на **рис. 3**.

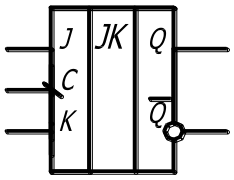


Рис. 3. Условно-графическое изображение JK триггера.

Триггер имеет два раздельных установочных входа J и K, аналогичных входам S и R RS триггера, и два выхода $\overline{Q}$ и Q. Третий вход триггера C является тактирующим или счетным. Установочные входы имеют приоритет по отношению к тактирующему. По этим входам триггер является асинхронным, т.е. его состояние изменяется в момент поступления входного сигнала. Если на установочные входы подать сигналы лог. "0", то состояние триггера будет определяться сигналом, поступающим на тактирующий вход. По тактирующему входу триггер – синхронный. В синхронном триггере состояние изменяется только в момент подачи специального тактового сигнала на дополнительный вход C (от первой буквы слова Clock).

Схема исследования триггера типа RS, представлена на рис. 3.

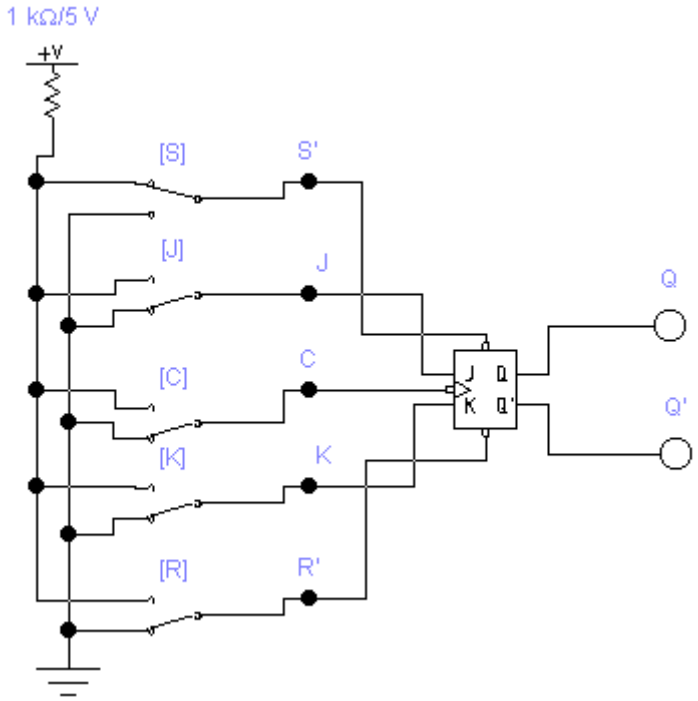


Рис. 3

JK триггер *управляется отрицательным фронтом синхроимпульса*, к его выходам Q и $\overline{Q}$ подключены логические пробники. Все входы JK триггера подключены при помощи переключателей к напряжению высокого уровня (+ 5 В) и низкого уровня (0 В, "земля"). Переключатели управляются клавишами R, S, J, K и C на клавиатуре (латинский регистр).

Замечание: *установочные входы R S инверсные и управляются низким уровнем напряжения.*

Установка начальных состояний триггера:

- Для получения состояния триггера $Q_t = 1$ необходимо подать сигнал S = 0 (R=1),
- Для получения состояния триггера $Q_t = 0$ - сигнал R = 0 (S=0).

Переход триггера в следующее состояние производится по отрицательному фронту (переключению от высокого уровня напряжения к низкому) на входе C.

Порядок исследования работы приоритетных входов R и S установки

1. Создайте новый файл и соберите схему исследования триггера рис. 3.
2. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_19_02** .

Примечание: Пиктограмма исследуемого триггера в окне цифровых микросхем пятая по счету слева направо.

3. Подайте сигналы S = 0, R = 1. Триггер установится в состояние Q = 1 независимо от сигналов на остальных входах..
4. Подайте сигналы S = 1, R = 0. Триггер установится в состояние Q = 0 независимо от сигналов на других входах.
5. Заполните таблицы функционирования JK триггера в Отчете. Для этого:
   - Приоритетные входы установки переключите в состояния R = 1, S = 1.
   - Подайте сигналы J = 1, K = 0.
   - Включите и выключите вход синхронизации C. Триггер переключится в состояние Q = 1, $\overline{Q}$ = 0.
   - Подайте сигналы J = 0, K = 1.
   - Включите и выключите вход синхронизации C. Триггер переключится в состояние Q = 0, $\overline{Q}$ = 1.

1.4. Исследование характеристик триггера типа JK в счетном режиме (T триггер) при помощи САПР.

Триггер типа T (счетный триггер) имеет один тактовый вход. Его состояние изменяется при поступлении на этот вход сигнала лог. "1". Промышленность отдельных микросхем T триггеров не выпускает. Обычно для реализации счетного режима используется JK или D триггер

На рис. 5 представлена схема T триггера на базе триггера JK.

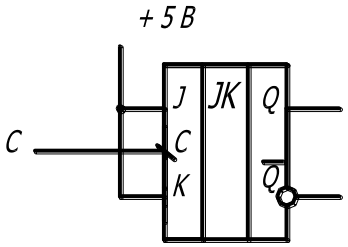


Рис. 5. Схема включения JK триггера в режим T триггера.

Для организации счетного режима работы JK триггера необходимо подать сигнал лог. "1" на входы J K. Тогда при каждом отрицательном фронте тактирующего сигнала C (см. Таблицы 3 и 4 в Отчете) состояние триггера будет изменяться на противоположное.

Схема исследования JK триггера, представлена на рис. 6.

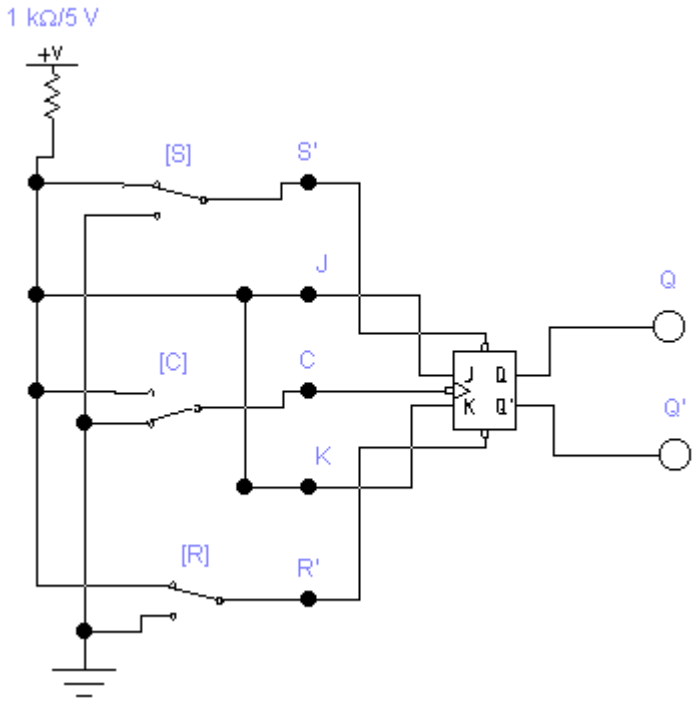


Рис. 6. Схема исследования JK триггера в счетном режиме (T триггера).

JK триггер *управляется отрицательным фронтом синхроимпульса*, к его выходам Q и $\overline{Q}$ подключены логические пробники. Входы R, S, J, K JK триггера подключены к напряжению высокого уровня (+ 5 В). Вход синхронизации C при помощи переключателя (управляется клавишей C на клавиатуре (английский регистр)) может подключаться к напряжению высокого и низкого уровня.

1. Создайте новый файл и соберите схему исследования рис. 6.
2. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_19_03** .
3. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Изменяя состояние входа C при помощи ключа, зарисуйте в Отчете временную диаграмму работы T триггера.

II. Самостоятельная работа. **Исследование характеристик триггера типа D в обычном и счетном режиме (T триггер).**

Рекомендуемое время

80 минут

Задание: №1. Экспериментальным путем получить таблицы перехода и функционирования для триггера типа D.

Триггер типа D является наиболее простым и широко применяемым при построении цифровых узлов. Триггер этого типа имеет один информационный вход D (Data) и тактирующий вход C , обычно работающий по положительному или отрицательному

фронту тактового импульса. Условное графическое обозначение триггера приведено на рис. 7.

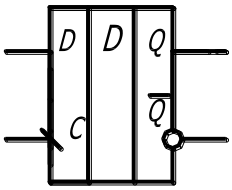


Рис. 7. Условно-графическое изображение D триггера.

Информация с входа D заносится в триггер в момент поступления фронта тактирующего импульса и сохраняется до поступления следующего тактирующего импульса. Кроме того, существуют варианты исполнения D триггеров, в которых тактирующий вход C управляется уровнем сигнала. Для последнего варианта D триггера существует явление "прозрачности", усложняющее проектирование цифровых узлов.

Во многих сериях микросхем D триггер дополнительно содержит установочные асинхронные входы R и S, которые являются приоритетными.

Схема исследования D триггера представлена на рис. 8.

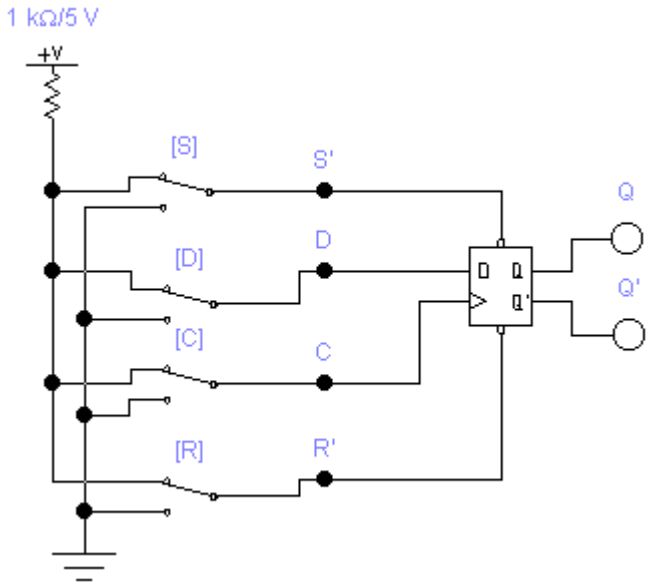


Рис. 8. Схема исследования D триггера.

Данный вариант D триггера управляется *положительным фронтом сигнала*, приходящего на вход синхронизации **C**.

Порядок исследования работы приоритетных входов R и S установки

1. Создайте новый файл и соберите схему исследования рис. 8. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_19_04**

2. Подайте сигналы S = 0, R = 1. Триггер установится в состояние Q = 1 независимо от сигналов на остальных входах.
3. Подайте сигналы S = 1, R = 0. Триггер установится в состояние Q = 0 независимо от сигналов на других входах.
4. Установите состояния R=1, S= 1. Заполните таблицы функционирования D триггера в Отчете. Для этого подайте необходимые комбинации сигналов на входы D и C, учитывая приведенные выше указания.
5. Зарисуйте временные диаграммы работы триггера при разных вариантах сигналов на входах D и C.

Задание: №2. Экспериментальным путем снять временную диаграмму работы триггера типа D в счетном режиме (T триггер).

На рис. 9 представлена схема T триггера на базе триггера D.

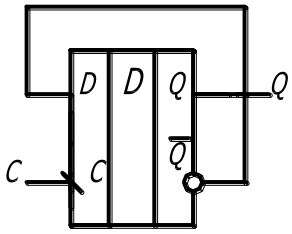


Рис. 9. Схема включения D триггера в режим T триггера.

Для организации счетного режима работы D триггера необходимо подать сигнал обратной связи с выхода $\overline{Q}$ на вход D. Тогда на каждом отрицательном фронте тактирующего сигнала C состояние триггера будет изменяться на противоположное.

Схема исследования D триггера в счетном режиме (T триггера) представлена на рис. 10.

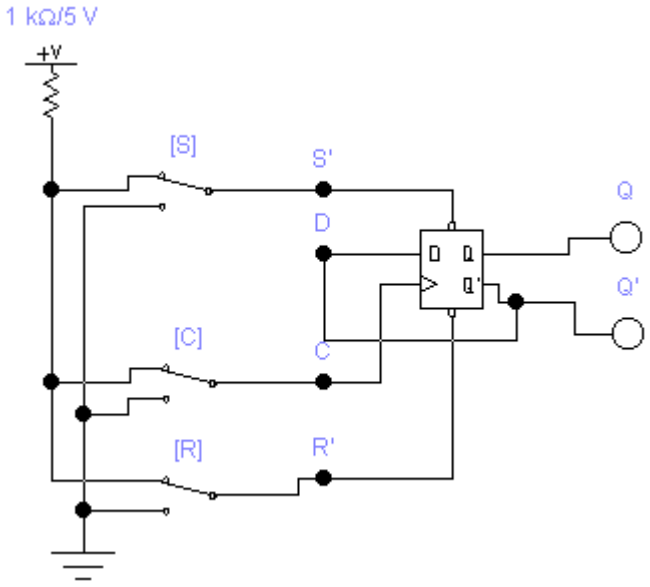


Рис. 10. Схема исследования D триггера в счетном режиме (Т триггера).

Данный вариант D триггера управляется *положительным фронтом сигнала*, приходящего на вход синхронизации С.

1. Соберите схему исследования рис. 10. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_19_05**
2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.
3. Переключайте вход С при помощи клавиши С. Нарисуйте временную диаграмму в Отчете.

Лабораторная работа №8-9

Тема: Исследование характеристик счетчиков

Цель: Овладение практическими навыками исследования характеристик счетчиков средствами САПР Electronics Workbench.

Результат обучения:

После успешного завершения занятия пользователь должен:

- Знать назначение и функциональные возможности счетчиков
- Уметь проводить исследование этих элементов средствами САПР **Electronics Workbench**.

Используемые программы:

Electronics Workbench в. 5.0

I. Исследование характеристик счетчиков

1.1. Краткое описание назначения и основных функциональных возможностей счетчиков.

Счетчиками называются цифровые узлы, которые под действием входных импульсов переходят из одного состояния в другое, изменяя свое состояние на единицу. В результате счетчик сохраняет число или код, соответствующий числу поступивших на его вход импульсов. Увеличение числа в счетчике на единицу называется операцией инкрементации, уменьшение на единицу – операцией декрементации.

Модуль счета или емкость счетчика **M** определяется как максимальное число различных состояний счетчика. После поступления **M** импульсов счетчик сбрасывается в начальное (нулевое) состояние и цикл счета повторяется.

Классификация счетчиков

- По способу кодирования внутренних состояний счетчики делятся на двоичные, двоично-десятичные и др.
- По направлению счета счетчики делятся на суммирующие, вычитающие и реверсивные.
- По организации межразрядных связей счетчики делятся на счетчики с последовательным или параллельным переносом.

Состояние счетчика определяется по выходам разрядных схем Q_{n-1} Q_{n-2} … Q_0, при этом входные импульсы поступают на младший разряд счетчика Q_0.

Счетчики бывают суммирующие и вычитающие. В суммирующем счетчике каждый импульс на входе увеличивает число на единицу, в вычитающем – уменьшает на единицу.

1.2. Исследование суммирующих двоичных счетчиков с последовательным переносом.

Двоичный счетчик имеет модуль $M = 2^n$, где n –целое число. Внутренние состояния двоичного счетчика образуют естественную последовательность, эквивалентную последовательности десятичных чисел 0, 1, ... M – 1. Счетчики обычно строятся на счетных T триггерах, построенных на базе D и JK триггеров.

Схема исследования простейшего последовательного двоичного счетчика, построенного на D триггерах с синхронизацией по переднему фронту синхроимпульса, представлена на рис. 1.

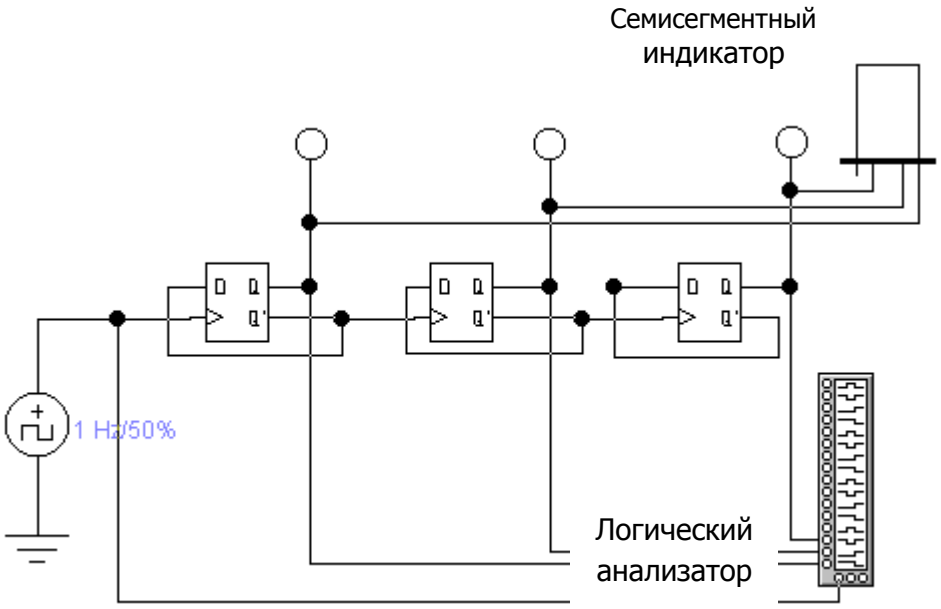


Рис.1. Схема исследования суммирующего двоичного счетчика с последовательным переносом

Счетчик собран на D триггерах, работающих в режиме счетного триггера. К выходу Q каждого триггера подключены логические пробники, выходы триггеров также подключены к логическому анализатору. Логический анализатор предназначен для отображения на экране лицевой панели временной диаграммы сигналов на выходах триггеров. Последовательность импульсов поступает на вход С левого триггера. Сигнал с его инверсного выхода $\overline{Q}$ поступает на вход С триггера второго разряда. Входы Q триггеров подключены к счетным входам триггеров следующего разряда.

Задача исследования: Получить временную диаграмму функционирования счетчика.

Построение схемы исследования идеальной модели двоичного счетчика

Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**.

1. Последовательно вытащите пиктограммы:
   - заземления, генератора импульсного напряжения (из группы источников);
   - D-триггеры без входов предустановки (из окна цифровых микросхем);
   - логических пробников, семисегментного цифрового индикатора (из группы индикаторов);
   - логического анализатора (из группы).
2. Разверните логические пробники, так как показано на рис. 1. Для этого на панели функций воспользуйтесь кнопкой поворота

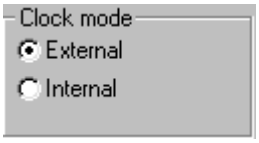
 .

3. Соедините элементы согласно рис. 1.

Настройка приборов

4. Установите значение частоты генератора импульсного напряжения на вкладке **Value** в поле **Frequency** равным **1 Hz**.
5. Откройте лицевую панель логического анализатора. Щелчком мышки по кнопке **Set…** блока **Clock**

откройте диалоговое окно установки параметров запускающих сигналов. Установите переключатель в режим **External**



и нажмите на кнопку **Accept.**

6. Сохраните файл в папке с вашей **Фамилией** под именем **Zan_13_01** .

Моделирование работы счетчика

7. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Наблюдайте процесс работы суммирующего счетчика.
8. Зарисуйте временную диаграмму его функционирования в Отчете.

1.3. Исследование вычитающих двоичных счетчиков с последовательным переносом.

В вычитающем счетчике (рис. 2) изменена схема переключения триггеров по сравнению со схемой суммирующего счетчика (рис. 1)

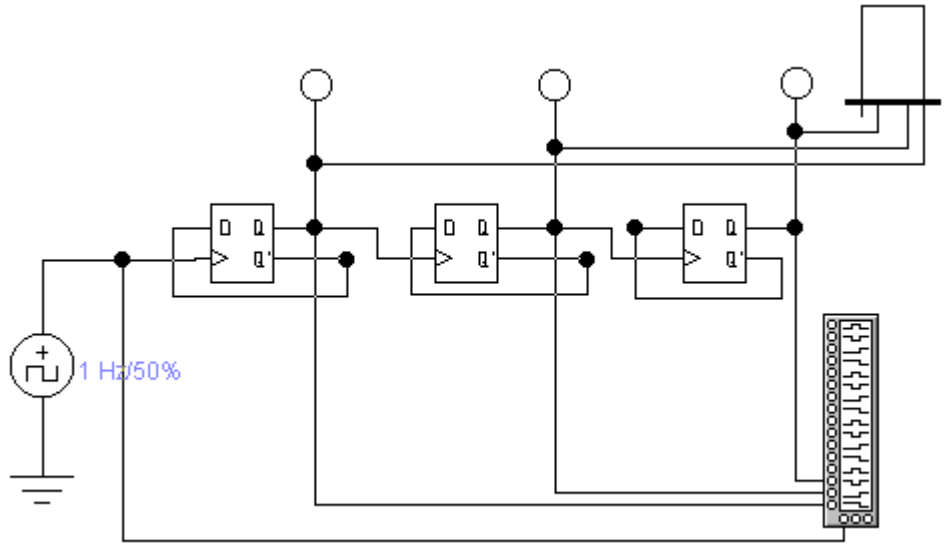


Рис. 2. Вычитающий двоичный счетчик с последовательным переносом.

Задача исследования: Получить и зарисовать временную диаграмму функционирования вычитающего счетчика в Отчете.

II. Самостоятельная работа. **Исследование двоичных счетчиков с последовательным переносом на базе JK триггеров**

Задание: Получить временную диаграмму функционирования суммирующего счетчика на базе JK триггеров (рис. 3). Результат работы покажите преподавателю.

Пояснение к схеме (рис. 3)

В левой верхней части схемы размещен переключатель, формирующий на выходных точках лог. "0" или лог. "1. Переключатель управляется клавишей **Space** и позволяет установить все триггеры в состояние лог. "0". Для этого необходимо на короткое время подключить входы установки триггеров к "земле" нажав на клавишу **Space**. В левой нижней части размещен генератор синхроимпульсов с подключенным к нему логическим пробником.

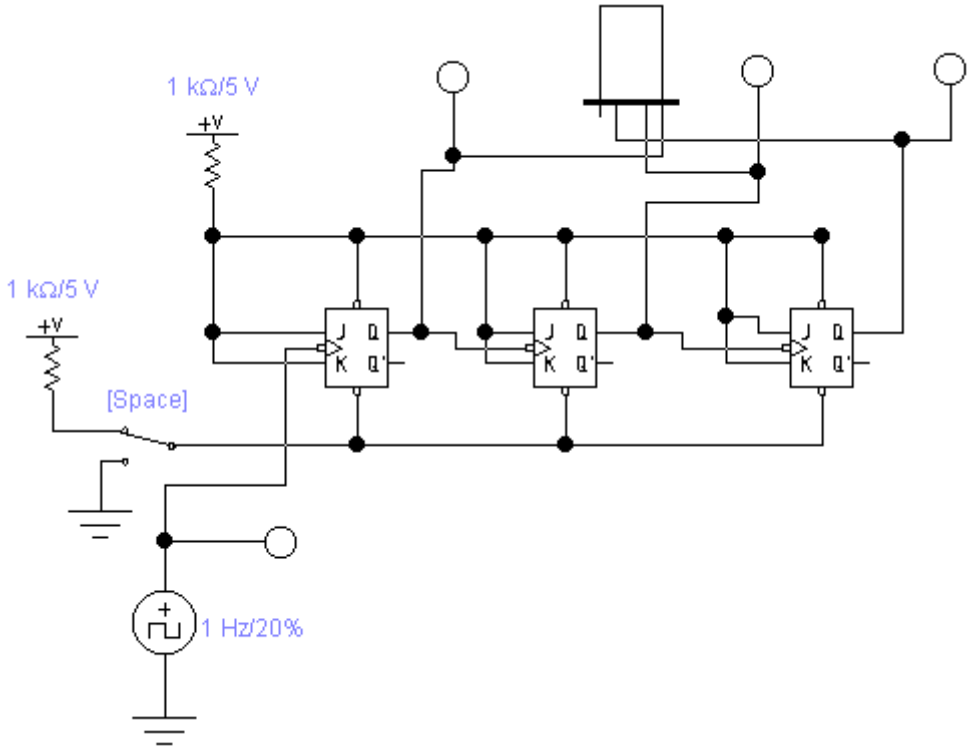


Рис.3. Схема исследования суммирующего двоичного счетчика с последовательным переносом на базе JK триггеров

Лабораторная работа №10-11

Тема: ***Исследование характеристик цифро-аналоговых преобразователей***

Цель: Овладение практическими навыками исследования характеристик цифро-аналоговых преобразователей средствами САПР **Electronics Workbench**.

Результат обучения:

После успешного завершения занятия пользователь должен:

- Знать назначение и функциональные возможности цифро-аналоговых преобразователей.
- Проводить анализ этих элементов средствами САПР **Electronics Workbench**.

1.1. Краткое описание назначения и основных характеристик цифро-аналоговых преобразователей.

Цифроаналоговые преобразователи (ЦАП) предназначены для преобразования цифровой информации, заданной в форме двоичных кодов, в аналоговые величины, обычно в напряжение или ток. Процесс цифроаналогового преобразования предполагает формирование в заданном диапазоне изменения выходного аналогового сигнала его

дискретных значений, отличающихся на определенный шаг, и сопоставление каждому дискретному значению выходного сигнала определенного двоичного кода.

ЦАП в настоящее время выпускаются промышленностью только в виде интегральных схем, которые могут быть как функционально завершенные, так и функционально незавершенные. Во втором случае необходимо использование внешних элементов, как правило источника опорного напряжения и операционного усилителя. Применение внешнего источника опорного напряжения позволяет разделить все ЦАП на две группы: *умножающие* и *неумножающие*. Первая группа допускает работу с изменяющимся во времени опорным напряжением, вторая – только с фиксированным значением опорного напряжения.

Технические характеристики ЦАП разделяются на *статические* и *динамические*. К статическим относятся:

1. *Разрядность* n - число двоичных разрядов кода, которые можно подать на вход ЦАП. Цифровая часть ЦАП разделяет величину опорного напряжения $U_{оп}$, $I_{оп}$ на N частей, связанных с разрядностью следующей формулой

$$N = 2^n$$

ЦАП широкого применения обычно содержат от 8 до 16 двоичных разрядов.

2. *Минимальный шаг изменения выходного напряжения или тока* $\Delta I_{вых}$, $\Delta U_{вых}$, при котором входной код изменяется на единицу. Минимальный шаг связан с разрядностью следующей формулой

$$\Delta I_{вых} = I_{оп} / N, \qquad \Delta U_{вых} = U_{оп} / N.$$

Выходное напряжение $U_{вых}$ или выходной ток $I_{вых}$ идеального ЦАП определяется следующими соотношениями

$$I_{вых} = I_{оп}\, m / N, \qquad U_{вых} = U_{оп}\, m / N,$$

где: m – десятичный эквивалент двоичного кода, поступающего на вход ЦАП, m = 0, 1,… N.

3. *Абсолютная погрешность преобразования* $\delta U_{вых}$, $\delta I_{вых}$, определяемая в конечной точке m = N равна разности выходных напряжений или токов идеального и реального ЦАП.

4. *Нелинейность преобразования* δL - максимальная разность выходных напряжений или токов для идеального и реального ЦАП в промежуточной точке m < N.
5. *Напряжение смещения нуля* – напряжение или ток на выходе ЦАП при подаче на его вход нулевого кода.

К динамическим характеристикам ЦАП относятся:

1. *Максимальная частота преобразования* $f_{с\ макс}$ - максимальная частота изменения входных кодов, допустимая для нормальной работы ЦАП.

2. *Время установления* t_s – время от момента поступления двоичного кода на вход ЦАП до момента установления выходного напряжения или тока.

Обычно характеристики ЦАП связывают с минимальным шагом изменения выходного напряжения $\Delta U_{вых}$ или тока $\Delta I_{вых}$. Так, абсолютная погрешность преобразования и нелинейность преобразования обычно равна минимальному шагу или, в точных ЦАПах, – половине минимального шага.

1.1. Исследование цифро-аналоговых преобразователей с параллельным суммированием токов.

ЦАП с весовыми резисторами или с параллельным суммированием токов (**рис. 1**) содержит источник опорного напряжения Uоп, ключи X0, X1, X2 и X3, управляемые входным двоичным кодом, резисторы R1, R2, R3 и R4, включенные последовательно с ключами и суммирующий усилитель Оу.

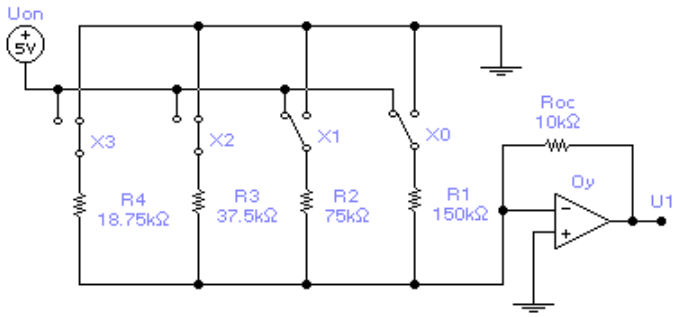


Рис. 1. Схема цифро – аналогового преобразователя с параллельным суммированием токов.

Если разряд двоичного кода на входе ЦАП равен "1", то соответствующий переключатель подключает резистор к источнику опорного напряжения, если равен "0" – к "земле" (на схеме на вход ЦАП подано двоичное число (0011)). Сопротивление резисторов от младшего к старшим разрядам уменьшаются

R2 = R1/2, R3 = R2/2, R4 = R3/2 и т.д.

В результате величина (вес) тока, проходящего по каждому резистору, будет разным. Минимальный суммарный ток на выходе резисторной схемы будет при поступлении двоичного числа (0001) I_0 = Uоп/R1. В результате на выходе Оу появится напряжение U1, равное минимальному шагу изменения выходного напряжения. Если на

вход подать число 4_{10} = (0100_2), то величина тока на выходе резисторной схемы будет равна I = Uоп/R3 = 4·Uоп/R1 = 4·I_0 . Поэтому и величина напряжения U1 на выходе Оу будет в четыре раза больше.

Схема ЦАП с параллельным суммированием токов содержит минимальное число резисторов. Однако все резисторы имеют разное сопротивление, которое нужно получить с высокой точностью. Так, для 10-ти разрядного ЦАП точность подгонки резисторов должна быть 0.1%, что технологически выполнить чрезвычайно сложно. Поэтому схема ЦАП с параллельным суммированием токов применяется лишь при малом числе двоичных разрядов.

1.2.1. Исследование цифро-аналоговых преобразователей с параллельным суммированием токов средствами САПР.

Задача исследования: определить минимальный шаг изменения выходного напряжения, построить характеристики вход – выход для идеального и реального ЦАП, провести их сравнительный анализ.

1. Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте папку **0646/ПТ_646_21**, а в ней файл **Z21_01.ewb**, в котором представлена схема экспериментального исследования ЦАП с параллельным суммированием токов и идеальным ОУ.

2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

а) ***Определение минимального шага изменения выходного напряжения*****.**

3. Подайте на вход ЦАП при помощи переключателей, управляемых клавишами A, B, C и D, двоичное число $(0001)_2$. Младшему разряду X0 соответствует клавиша A. Вольтметр, подключенный к выходу ОУ, покажет напряжение, соответствующее минимальному шагу. Запишите результат в Отчет.

б) ***Построение характеристики вход – выход*** **для идеального ЦАП.**

4. Подайте на вход ЦАП последовательность двоичных чисел и запишите показания вольтметра в первый столбец Таблицы 1 в Отчете.

Таблица 1.

Сравнение характеристик вход – выход идеального и реального ЦАП.

N	Двоичное число	Выходное напряжение идеального ЦАП	Выходное напряжение реального ЦАП
0	0000		

1	0001		
2	0010		
3	0011		
4	0100		
5	0101		
6	0110		
7	0111		
8	1000		
9	1001		
10	1010		
11	1011		
12	1100		
13	1101		
14	1110		
15	1111		

5. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_01_1.ewb**, в котором представлена схема экспериментального исследования ЦАП с параллельным суммированием токов и реальным ОУ LM 1875.

6. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

в) ***Определение напряжения смещения нуля.***

7. Подайте на вход схемы двоичное число $(0000)_2$. Запишите показания вольтметра во вторую графу Таблицы 1 в Отчете.

г) ***Определение минимального шага изменения выходного напряжения*****.**

8. Подайте на вход ЦАП двоичное число $(0001)_2$. Младшему разряду X0 соответствует клавиша A. Запишите показание вольтметра во вторую строку второго столбца Таблицы 1 в Отчете. Найдите разность напряжений, измеренных в экспериментах г) и в). Это напряжение, соответствующее минимальному шагу. Запишите результат в Отчет.

д) ***Построение характеристики вход – выход для реального ЦАП.***

9. Подайте на вход ЦАП последовательность двоичных чисел и запишите показания вольтметра во второй столбец Таблицы 1 в Отчете.

Определите:

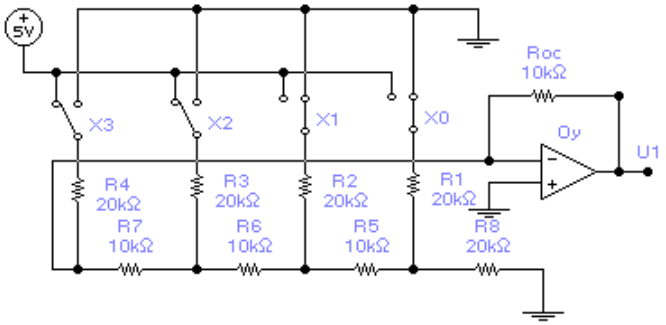
- *Абсолютную погрешность преобразования*, определяемую в конечной точке m = 15 $\delta U_{вых}$. Она равна разности выходных напряжений идеального и реального ЦАПов.

- *Нелинейность преобразования* δL. Она равна максимальной разности выходных напряжений для идеального и реального ЦАП в промежуточной точке

Запишите результаты в Отчет.

1.3. Исследование цифро-аналоговых преобразователей с последовательным суммированием токов. Вариант №1.

ЦАП с последовательным суммированием токов содержат резисторы только двух номиналов: R и 2R. Технология изготовления таких ЦАП проще и поэтому они находят широкое применение. Схема ЦАП с последовательным суммированием токов (ЦАП лестничного типа) представлена на **рис. 2**.



3 2 1 0

Рис. 2. Схема цифро – аналогового преобразователя с последовательным суммированием током.

Если разряд двоичного кода на входе ЦАП равен "1", то соответствующий переключатель подключает резистор к источнику опорного напряжения, если равен "0" – к "земле" (на схеме на вход ЦАП подано двоичное число (1100)).

Принцип работы матрицы R-2R этого варианта ЦАП состоит в следующем (на схеме резисторы R – R5, R6, R7 (по 10 кОм), резисторы 2R – R1, R2, R3, R4 и R8 (по 20 кОм)). Если ключ X0 подключен к "земле", то суммарное сопротивление между узлом схемы "0" и "землей" равно сопротивлению двух параллельно соединенных резисторов R1 и R8. Это сопротивление равно $R_э = R = 10$ кОм. Аналогично сопротивление между узлом "1" и "землей" равно сопротивлению двух параллельных цепей : R2 и R5 + $R_э$, то есть также равно R = 10 кОм. Аналогично получаем такой же результат и для остальных узлов "2" и "3". Когда на вход ЦАП поступает двоичное число, то соответствующие ветви матрицы сопротивлений включаются параллельно и токи, соответствующие двоичным разрядам с "1" получают веса, аналогичные весам двоичных разрядов в ЦАПе с параллельным суммированием токов.

Напряжение на выходе ЦАП с последовательным суммированием токов находится по формуле $$\mathbf{U1 = U_{оп}\frac{R_{ос}}{R}\left(\frac{L}{2^1}+\frac{K}{2^2}+\frac{J}{2^3}+\frac{I}{2^4}\right);}$$

В этой формуле буквами (L, K, J и I) обозначены "0" или "1" двоичного числа $(L\ K\ J\ I)_2$, поступающего на вход ЦАП (младший разряд справа).

ЦАП с последовательным суммированием токов значительно технологичнее Цап с параллельным суммированием токов. Это связано с тем, что в его схеме применяются резисторы двух номиналов R и 2R ($R_{ос}$ всегда можно сделать равным R). Поэтому этот тип ЦАП нашел самое широкое распространение.

1.3.1. Исследование цифро-аналоговых преобразователей с последовательным суммированием токов средствами САПР. Вариант №1.

Задача исследования: определить минимальный шаг изменения выходного напряжения, построить характеристики вход – выход для идеального и реального ЦАП, провести их сравнительный анализ.

1. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_02.ewb**, в котором представлена схема экспериментального исследования ЦАП с последовательным суммированием токов и идеальным ОУ.

2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

а) ***Определение минимального шага изменения выходного напряжения*****.**

3. Подайте на вход ЦАП при помощи переключателей, управляемых клавишами A, B, C и D, двоичное число $(0001)_2$. Младшему разряду X0 соответствует клавиша A. Вольтметр, подключенный к выходу ОУ, покажет напряжение, соответствующее минимальному шагу. Запишите результат в Отчет.

б) ***Построение характеристики вход – выход для идеального ЦАП*****.**

4. Подайте на вход ЦАП последовательность двоичных чисел и запишите показания вольтметра в первый столбец Таблицы 2 в Отчете.

Таблица 2.

Сравнение характеристик вход – выход идеального и реального ЦАП.

N	Двоичное число	Выходное напряжение идеального ЦАП	Выходное напряжение реального ЦАП
0	0000		
1	0001		
2	0010		
3	0011		
4	0100		
5	0101		
6	0110		
7	0111		
8	1000		
9	1001		
10	1010		
11	1011		
12	1100		
13	1101		
14	1110		
15	1111		

5. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_02_1.ewb**, в котором представлена схема экспериментального исследования ЦАП с параллельным суммированием токов и реальным ОУ LM 841.

6. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

в) ***Определение напряжения смещения нуля.***

7. Подайте на вход схемы двоичное число $(0000)_2$. Запишите показания вольтметра во вторую графу Таблицы 1 в Отчете.

г) ***Определение минимального шага изменения выходного напряжения*****.**

8. Подайте на вход ЦАП двоичное число $(0001)_2$. Младшему разряду X0 соответствует клавиша A. Запишите показание вольтметра во вторую строку второго столбца Таблицы 1 в Отчете. Найдите разность напряжений, измеренных в экспериментах г) и в). Это напряжение, соответствующее минимальному шагу. Запишите результат в Отчет.

д) ***Построение характеристики вход – выход для реального ЦАП*****.**

9. Подайте на вход ЦАП последовательность двоичных чисел и запишите показания вольтметра во второй столбец Таблицы 1 в Отчете.

Определите:

- *Абсолютную погрешность преобразования*, определяемую в конечной точке m = 15 $\delta U_{вых}$. Она равна разности выходных напряжений идеального и реального ЦАП.

- *Нелинейность преобразования* δL. Она равна максимальной разности выходных напряжений для идеального и реального ЦАП в промежуточной точке

Запишите результаты в Отчет.

1.4. Исследование цифро-аналогового преобразователя с последовательным суммированием токов. Вариант №2.

Изучим работу ЦАП, в котором в качестве коммутирующего устройства используется двоично – десятичный счетчик 74160 (К155ИЕ9) (**рис. 3**).

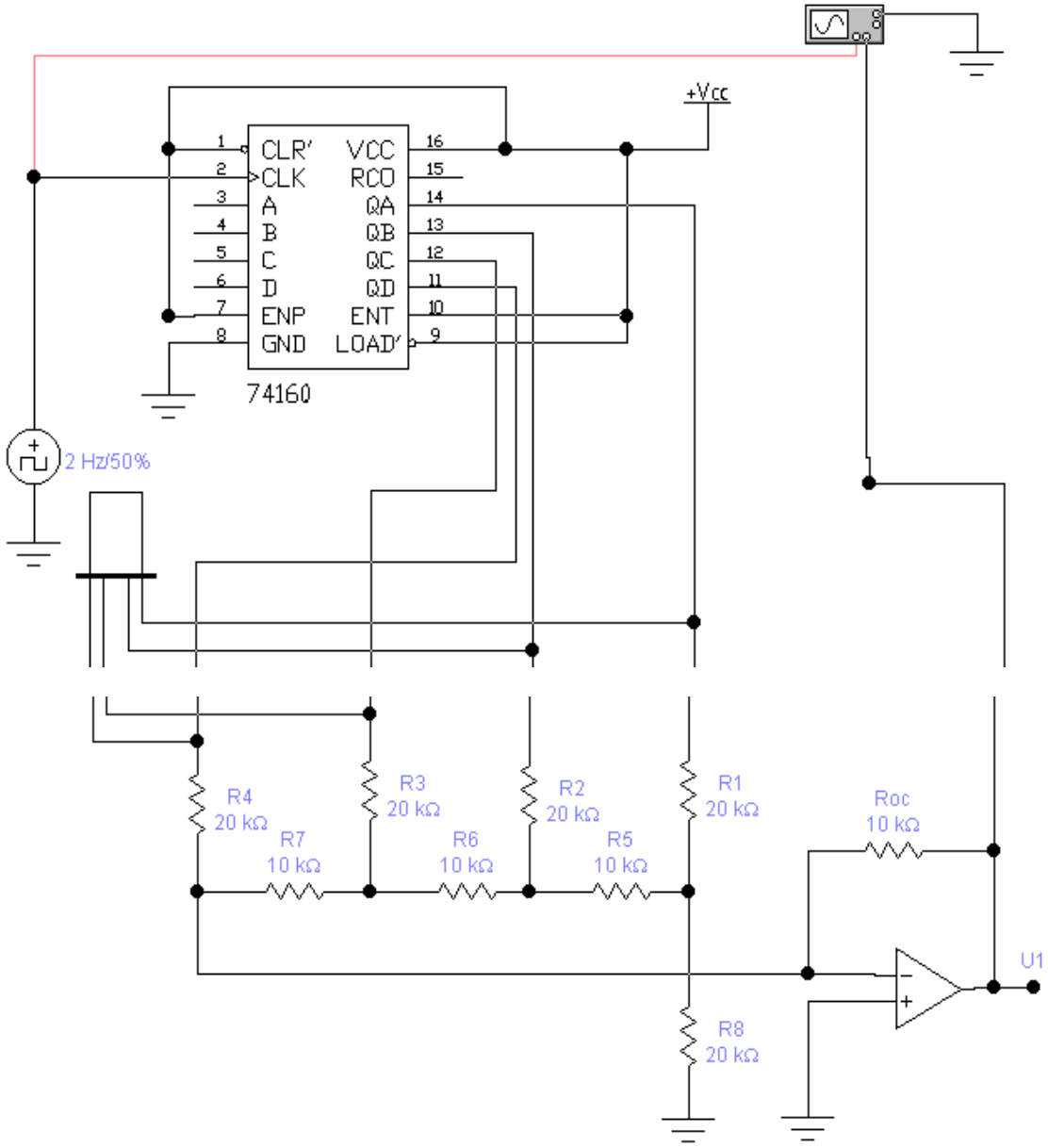


Рис. 3. Схема ЦАП с последовательным суммированием токов и управлением от счетчика К155ИЕ9.

Роль ключей, управляющих матрицей резисторов R-2R,в этой схеме выполняют выходные каскады счетчика, подающие на входы матрицы напряжения высокого и низкого уровней в стандарте ТТЛ. Поэтому в этом варианте ЦАП отсутствует источник опорного напряжения.

После включения схемы на вход синхронизации счетчика начинают поступать импульсы от генератора прямоугольных импульсов  . Состояние счетчика выводится на семисегментный индикатор. Выходное напряжение с ЦАП и синхроимпульсы выводятся на экран двухлучевого осциллографа (**рис. 4**).

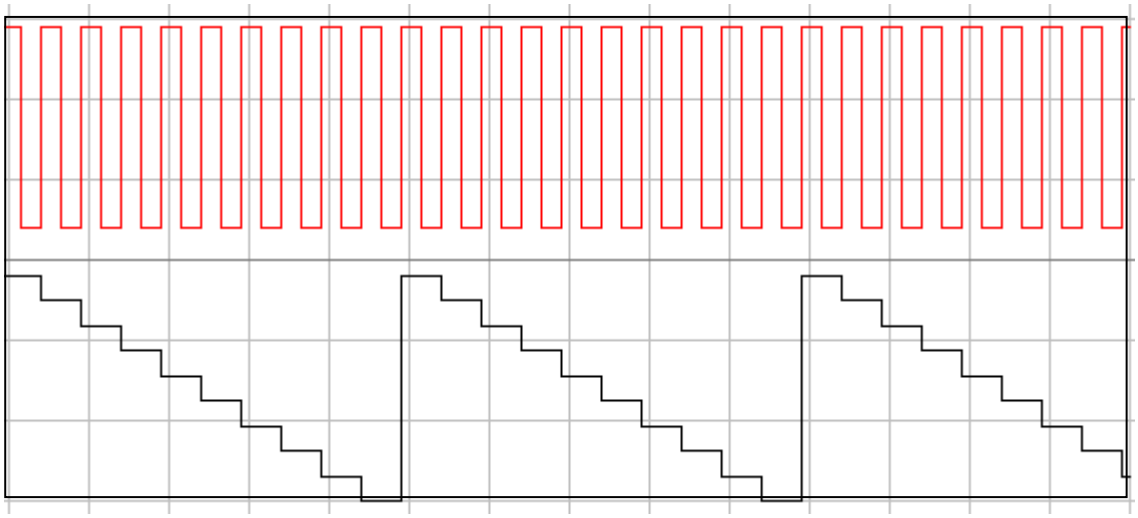


Рис. 4. Осциллограмма выходного напряжения ЦАП (черная) и синхроимпульсов (красная). Масштаб для выходного напряжения – 1 В/дел., масштаб по оси времени – 1 с/дел.

Из осциллограммы видно, что переключение счетчика (и ЦАП) происходит по заднему фронту синхроимпульсов. Эта осциллограмма с учетом масштабов по осям позволяет измерить *минимальный шаг изменения выходного напряжения* и *время установления* выходного напряжения.

1.4.1. Исследование цифро-аналоговых преобразователей с последовательным суммированием токов средствами САПР. Вариант №2.

1. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_03.ewb**, в котором представлена схема экспериментального исследования ЦАП с последовательным суммированием токов и идеальным ОУ.

2. Откройте изображение осциллографа, два раза щелкнув мышкой по его пиктограмме.

3. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Зарисуйте форму импульса выходного напряжения в Отчет.

Определение минимального шага изменения выходного напряжения при помощи осциллографа программы **Electronics Workbench**. По осциллограмме выходного напряжения определите минимальный шаг. Для этого передвиньте при помощи мышки красный (1) и синий (2) визиры на экране осциллографа на две соседние площадки осциллограммы выходного напряжения (**рис. 5**) по их пиктограммам.

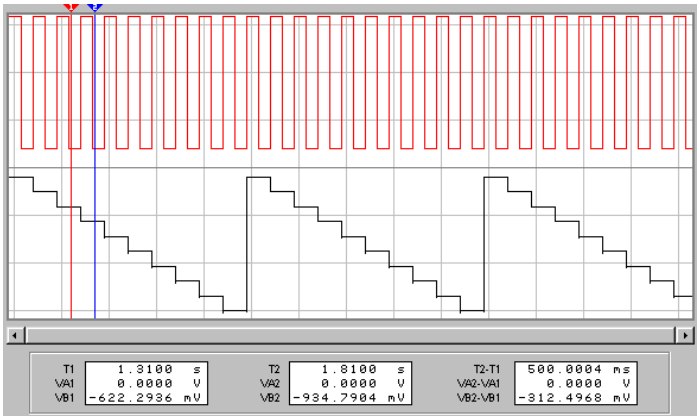


Рис. 5. Измерения при помощи осциллографа программы **Electronics Workbench**.

Показания в окнах:

T1 = 1.31 с – время развертки до красного визира 1,

VA1 = 0.00 В – напряжение в точке пересечения красного визира 1 и осциллограммы канала А (красного),

VB1 = - 0.6249986 В - напряжение в точке пересечения красного визира 1 и осциллограммы канала В (черного)

T2, VA2 и VB2 – те же величины для синего визира 2.

T2 – T1 = 0.5 с – время развертки между синим и красным визирами,

VA2 – VA1 = 0.00 – разность напряжений в точках пересечения синего и красного визиров с осциллограммой канала А.

VB2 – VB1 = - 0.3124993 В – разность напряжений в точках пересечения синего и красного визиров с осциллограммой канала В.

Показания в правом окне осциллографа позволяют определить минимальный шаг изменения выходного напряжения $\Delta U_{вых}$ = VB2 – VB1 = - 0.313 В.

4. Остановите процесс моделирования при помощи выключателя.

1.5. Построение функциональных генераторов на базе ЦАП.

Цифро-аналоговые преобразователи позволяют создавать различные варианты схем функциональных генераторов для таблично заданных (сеточных) функций вида y = f(t). Действительно, разобьем отрезок изменения аргумента t на равные части и составим таблицу значений функции в этих точках. Эти значения представим наиболее близкими к

ним числами в двоичном коде. Для получения двоичного кода значений функции необходимо ее пронормировать на величину опорного напряжения ЦАП.

Значения функции y = sin (t) на отрезке 0˚ – 90˚ представлены в **Таблице 3**.

Таблица 3.

Значения функции y = sin (t)

t, град	sin (t)	X7	X6	X5	X4	X3	X2	X1	X0
0	0.0	0	0	0	0	0	0	0	0
10	0.1736	0	0	0	1	1	1	1	0
20	0.3420	0	0	1	1	1	0	1	0
30	0.5	0	1	0	1	0	1	0	1
40	0.6428	0	1	1	0	1	1	0	1
50	0.7660	1	0	0	0	0	0	1	0
60	0.8660	1	0	0	1	0	0	1	1
70	0.9397	1	0	1	0	0	0	0	0
80	0.9848	1	0	1	0	0	1	1	1
90	1.0	1	0	1	0	1	0	1	0

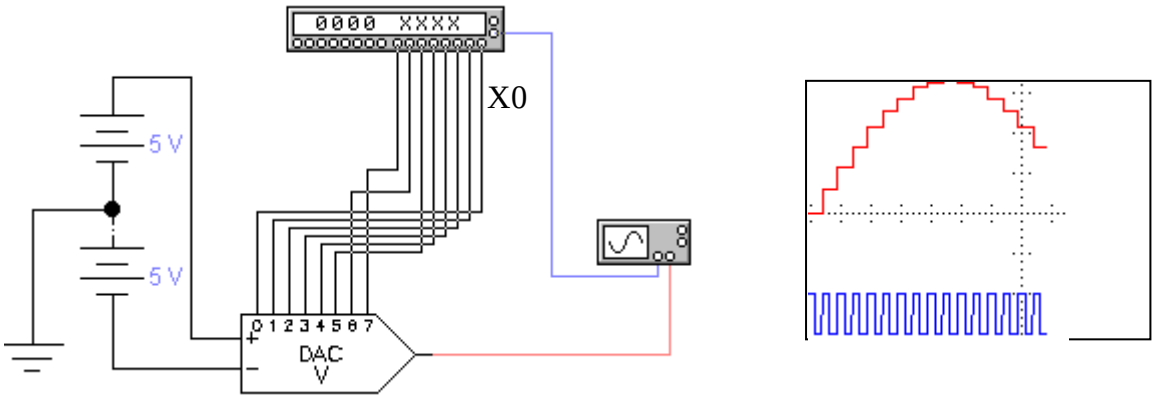


Рис. 6. Схема функционального генератора на базе ЦАП.

Функциональный генератор (**рис. 6**) содержит 8-ми разрядный ЦАП, цифровые входы которого подключены к Генератору двоичных слов (на схеме вверху). К выходу ЦАП и выходу синхронизации Генератора слов подключен двухлучевой осциллограф. К ЦАП подключены источники опорного напряжения: + 5 В и – 5 В.

В Генератор двоичных слов занесены двоичные слова из **Таблицы 3**. Эти слова по очереди в постоянном темпе выводятся на входы ЦАП. В результате на экране осциллографа строится ступенчатая функция (красный цвет), аппроксимирующая заданную функцию y = sin (t). Синим цветом внизу экрана показаны синхроимпульсы.

Понятно, что для повышения точности реализации заданной функции необходимо уменьшить шаг или, что тоже самое, увеличить число значений функции в таблице. Для реализации такой схемы необходимо использовать ПЗУ достаточной емкости.

1.5.1. Анализ работы функционального генератора на базе ЦАП при помощи САПР.

1. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_04.ewb**, в котором представлена схема экспериментального исследования функционального генератора на базе 8-ми разрядного ЦАП.

2. Откройте изображения Генератора слов и осциллографа (если их изображения скрыты), два раза щелкнув мышкой.

3. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Наблюдайте график функции, который строится на экране осциллографа.

4. Остановите процесс моделирования при помощи выключателя.

II. Самостоятельная работа.

Задание:

Проведите анализ работы функционального генератора на базе ЦАП при помощи САПР.

1. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z21_05.ewb**, в котором представлена схема экспериментального исследования функционального генератора на базе 8-ми разрядного ЦАП.

2. Запрограммируйте Генератор слов функционального генератора в соответствии с заданными вариантами таблично заданных функций:

Вариант 1.

Функциональный генератор для функции y = tg (t). Значения функции y = tg (t) занесены в **Таблицу 4**.

Таблица 4.

Значения функции y = tg (t).

t, град	tg (t)	X7	X6	X5	X4	X3	X2	X1	X0
0	0	0	0	0	0	0	0	0	0
5	0.08748	0	0	0	0	0	1	0	0
10	0.1763	0	0	0	0	0	1	1	1
15	0.2679	0	0	0	0	1	0	1	1
20	0.3640	0	0	0	0	1	1	1	1
25	0.4663	0	0	0	1	0	1	0	0
30	0.5774	0	0	0	1	1	0	0	1
35	0.7002	0	0	0	1	1	1	1	0
40	0.8391	0	0	1	0	0	1	0	0
45	1.0	0	0	1	0	1	0	1	0
50	1.192	0	0	1	1	0	0	1	1
55	1.4281	0	0	1	1	1	1	0	1
60	1.7321	0	1	0	0	1	0	1	0
65	2.1445	0	1	0	1	1	0	1	1
70	2.2775	0	1	1	1	0	1	0	1
75	4.0107	1	0	0	1	1	1	1	1

Вариант 2.

Функциональный генератор для трапецевидной функции

Значения функции занесены в **Таблицу 5**.

Таблица 5.

Значения трапецевидной функции.

T	F (t)	X7	X6	X5	X4	X3	X2	X1	X0

0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	1	1	1	0
2	2	0	1	0	1	1	1	0	1
3	3	1	0	0	0	1	0	1	1
4	4	1	0	1	1	1	0	0	1
5	5	1	1	1	0	1	0	0	0
6	5	1	1	1	0	1	0	0	0
7	5	1	1	1	0	1	0	0	0
8	5	1	1	1	0	1	0	0	0
9	5	1	1	1	0	1	0	0	0
10	5	1	1	1	0	1	0	0	0
11	4	1	0	1	1	1	0	0	1
12	3	1	0	0	0	1	0	1	1
13	2	0	1	0	1	1	1	0	1
14	1	0	0	1	0	1	1	1	0
15	0	0	0	0	0	0	0	0	0

Вариант 3.

Функциональный генератор для треугольной функции

Значения функции занесены в **Таблицу 6**.

Таблица 6.

Значения треугольной функции.

T	F (t)	X7	X6	X5	X4	X3	X2	X1	X0
0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	1	1	1	0
2	2	0	1	0	1	1	1	0	1

3	3	1	0	0	0	1	0	1	1
4	4	1	0	1	1	1	0	0	1
5	5	1	1	1	0	1	0	0	0
6	4	1	0	1	1	1	0	0	1
7	3	1	0	0	0	1	0	1	1
8	2	0	1	0	1	1	1	0	1
9	1	0	0	1	0	1	1	1	0
10	0	0	0	0	0	0	0	0	0
11	1	0	0	1	0	1	1	1	0
12	2	0	1	0	1	1	1	0	1
13	3	1	0	0	0	1	0	1	1
14	4	1	0	1	1	1	0	0	1
15	5	1	1	1	0	1	0	0	0

3. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана. Перерисуйте осциллограммы работы функционального генератора в Отчет.

Занятие окончено.

Лабораторная работа 12-13

Тема: Исследование характеристик аналого-цифровых преобразователей

Цель: Овладение практическими навыками исследования характеристик аналого-цифровых преобразователей (АЦП) с использованием средств САПР **Electronics Workbench**.

Результат обучения:

После успешного завершения занятия пользователь должен:

- Знать назначение и функциональные возможности аналого-цифровых преобразователей.
- Проводить анализ этих элементов средствами САПР **Electronics Workbench**.

I. Исследование характеристик АЦП.

1.1. Краткое описание назначения и основных характеристик аналого-цифровых преобразователей.

Аналого–цифровые преобразователи предназначены для преобразования аналогового сигнала в цифровой код. Процесс аналого–цифрового преобразования состоит из следующих действий:

- выделение аналогового сигнала в некоторый момент времени или дискретизация аналогового сигнала с заданным шагом по времени,
- квантование аналогового сигнала, т.е. округление с заданным шагом по уровню сигнала,
- сопоставление квантованному аналоговому сигналу некоторого цифрового кода.

Квантование аналогового сигнала по времени и по уровню приводит к потере информации. Так как ширина частотного спектра аналогового сигнала в технических устройствах ограничена, то можно выбрать такой малый шаг дискретизации по времени, что потери информации не будет (**теорема Котельникова**). Квантование сигнала по уровню всегда приводит к потере информации. Эта потеря может быть уменьшена до очень малой величины путем увеличения разрядности АЦП.

Технические характеристики АЦП разделяются на *статические* и *динамические*.

К статическим относятся:

1. *Разрядность* n - число двоичных разрядов кода, которые можно получить на выходе АЦП. Цифровая часть АЦП разделяет величину опорного напряжения $U_{оп}$,$I_{оп}$ на N частей, связанных с разрядностью следующей формулой

$$N = 2^n$$

   АЦП широкого применения обычно содержат от 8 до 16 двоичных разрядов.

2. *Минимальный шаг квантования или минимальный шаг изменения входного напряжения или тока* $\Delta I_{вх}$, $\Delta U_{вх}$, при котором выходной код изменяется на единицу. Минимальный шаг связан с разрядностью следующей формулой

$$\Delta I_{вх} = I_{оп} / N, \qquad \Delta U_{вх} = U_{оп} / N.$$

   Входное квантованное напряжение $U_{вх}$ или входной квантованный ток $I_{вх}$ идеального АЦП определяется следующими соотношениями

$$I_{вх} = I_{оп}\, m / N, \qquad U_{вх} = U_{оп}\, m / N,$$

   где: m – десятичный эквивалент двоичного кода на выходе АЦП. m = 0, 1, … N.

3. *Абсолютная погрешность преобразования* $\delta U_{вх}$, $\delta I_{вх}$, определяемая в конечной точке m = N равна разности входных напряжений или токов идеального и реального АЦП.

4. *Нелинейность преобразования* δL - максимальная разность входных напряжений или токов для идеального и реального АЦП в промежуточной точке $m < N$.

5. *Напряжение смещения нуля* – напряжение или ток на входе АЦП, которое нужно приложить для получения на выходе нулевого кода.

К динамическим характеристикам АЦП относятся:

1. *Максимальная частота преобразования* $f_{с\ макс}$ - максимальная частота дискретизации по времени, допустимая для нормальной работы ЦАПа.

2. *Время преобразования* t_c– время от момента поступления сигнала на вход АЦП до момента установления цифрового кода.

Обычно характеристики АЦП связывают с минимальным шагом изменения входного напряжения $\Delta U_{вх}$ или тока $\Delta I_{вх}$. Так, абсолютная погрешность преобразования и нелинейность преобразования обычно равна минимальному шагу или, в точных АЦП, – половине минимального шага.

1.1. Исследование аналого-цифровых преобразователей с последовательным преобразованием.

Схема исследования АЦП с последовательным преобразованием представлена на **рис. 1**.

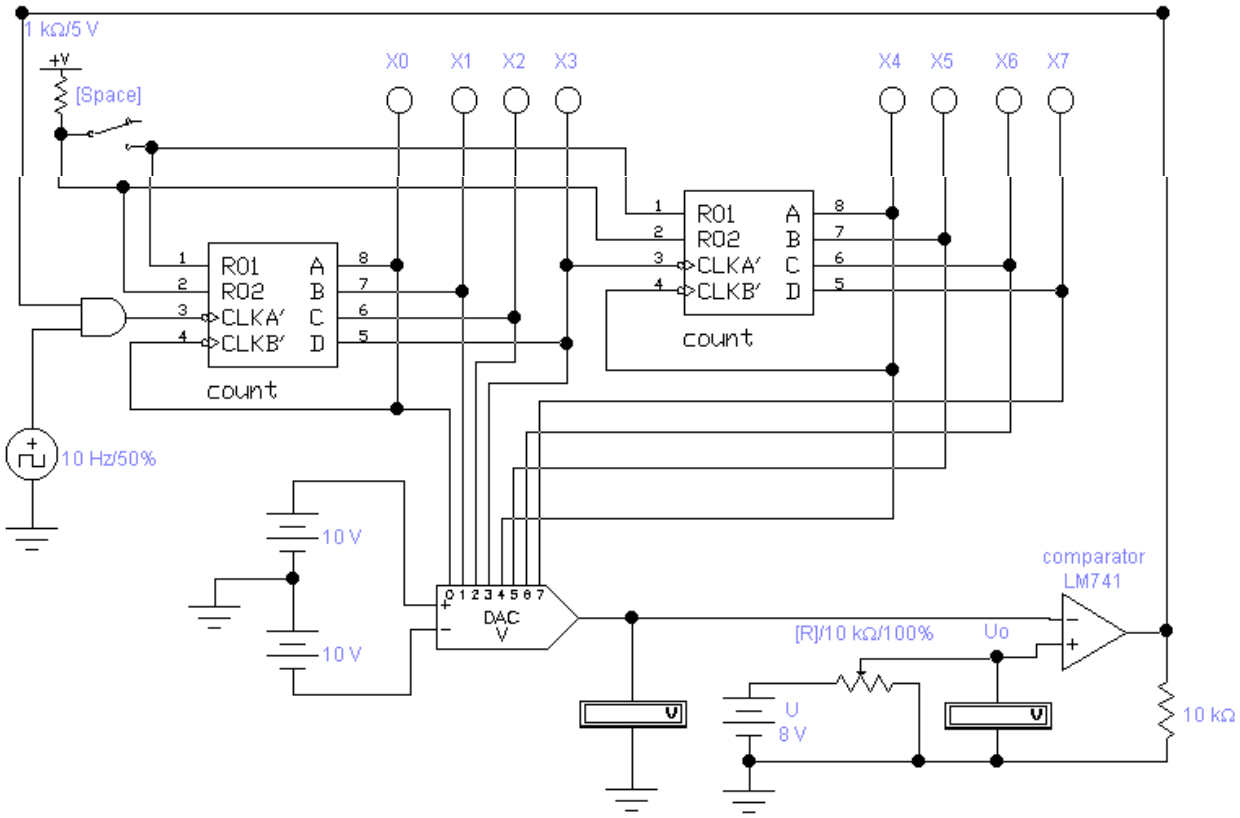


Рис. 1. Схема исследования АЦП с последовательным преобразованием.

АЦП содержит 8-ми разрядный двоичный счетчик, выходы двоичных разрядов которого подключены к цифровым входам 8-ми разрядного ЦАП (DAC). К ЦАП подключены источники опорного напряжения + 10 В и – 10 В. К аналоговому выходу ЦАП подключен инверсный вход операционного усилителя (ОУ), работающего в режиме компаратора напряжения. На второй вход ОУ подается входной аналоговый сигнал АЦП. Компаратор изменяет свое выходное состояние при переходе входного напряжения через заданное пороговое напряжение. В компараторе ОУ работает в режиме ограничения, при

котором его выходное напряжение достигает максимально возможного значения. Пороговое напряжение на прямом входе ОУ задается при помощи переменного резистора, величина сопротивления которого регулируется с помощью кнопок **R** и **Shift – R**. Это напряжение измеряется вольтметром.

Работает АЦП следующим образом. Перед началом преобразования счетчик сбрасывается в ноль. Аналоговый сигнал подается на аналоговый вход Uo и включается счетчик. Для этого тактовые импульсы подаются через ячейку "И" на вход счетчика, счетчик подсчитывает единицы, в результате код числа на выходах двоичных разрядов счетчика и на цифровых входах ЦАП увеличивается. На выходе ЦАП появляется возрастающее пилообразное напряжение, которое в какой то момент превышает напряжение Uo, поданное на аналоговый вход. В этот момент напряжение на выходе ОУ компаратора меняет знак, ячейка "И" закрывается, поступление импульсов на вход счетчика прекращается. Цифровой код на выходах счетчика сопоставляется уровню аналогового сигнала.

Последовательный АЦП является простым по аппаратным затратам, но обладает низким быстродействием. Так, максимальное время преобразования для 8-ми разрядного АЦП составляет $2^8 = 256$ импульсов и резко возрастает по мере роста разрядности.

1. Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте папку **0646/ПТ_646_22**, а в ней файл **Z22_01.ewb**, в котором представлена схема экспериментального исследования АЦП с последовательным преобразованием.

Напряжение на вход АЦП подается с переменного резистора, его сопротивление регулируется клавишами **R –** уменьшение и **Shift – R –** увеличение величины сопротивления. Положение движка потенциометра в процентах показывается рядом с его изображением. Напряжение на входе АЦП показывает правый вольтметр. Второй вольтметр показывает напряжение на выходе ЦАП, которое определяет уровни квантования входного сигнала. Логические пробники вверху схемы подключены к выходам двоичных разрядов счетчика и показывают двоичный код, сопоставляемый уровню аналогового сигнала. Пуск процесса преобразования и сброс счетчика в ноль производится ключом, управляемым клавишей **Space** (пробел).

2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

а) ***Определение минимального шага изменения входного напряжения.***

3. Установите ключ **Space** в замкнутое положение (сброс).

4. Установите переменный резистор в положение 100% (при помощи сочетания клавиш **Shift – R**). Коротким нажатием на клавишу **R** переведите резистор в положение 99% (на один шаг). *Вернуться на один шаг, если это необходимо, можно при помощи клавиш* **Shift – R**. Дальше необходимо сбросить счетчик в ноль клавишей **Space**.

5. Разомкните ключ **Space** и после окончания процесса преобразования прочитайте и запишите в первую строку **Таблицы 1** Отчета показания обоих вольтметров и двоичный код, сопоставленный входному аналоговому сигналу.

6. Замкните ключ **Space** и сбросьте счетчик в ноль. Коротким нажатием на клавишу **R** переведите резистор в положение 98% (на один шаг).

7. Разомкните ключ **Space** и после окончания процесса преобразования прочитайте и запишите во вторую строку **Таблицы 1** Отчета показания обоих вольтметров и

двоичный код, сопоставленный входному аналоговому сигналу. Замкните ключ **Space** и сбросьте счетчик в ноль.

8. Сравните двоичные коды, они в этих двух случаях должны отличаться на единицу младшего разряда. Найдите разность показаний левого вольтметра, подключенного к выходу ЦАП. Это будет напряжение, соответствующее шагу квантования в АЦП, или минимальный шаг изменения входного напряжения. Запишите результат в Отчет.

б) *Построение начального участка характеристики вход – выход* для АЦП.

9. Коротким нажатием на клавишу **R** переведите резистор в положение 97% (на один шаг).

Разомкните ключ **Space** и после окончания процесса преобразования прочитайте и запишите в третью строку **Таблицы 1** Отчета показания обоих вольтметров и двоичный код, сопоставленный входному аналоговому сигналу. Замкните ключ **Space** и сбросьте счетчик в ноль. Продолжайте этот процесс до полного заполнения **Таблицы 1.** Сравнивайте двоичные коды в соседних строках таблицы, они должны отличатся на единицу младшего разряда, а уровни входного аналогового сигнала (показания правого вольтметра) должны лежать в промежутках между двумя соседними уровнями квантования (показания левого вольтметра). В результате в таблице будут занесены точки характеристики вход – выход АЦП.

Обратите внимание на рост времени преобразования по мере увеличения уровня аналогового сигнала на входе АЦП.

Таблица 1

Характеристика вход – выход АЦП.

N п/п	Показания Вольтметра на выходе ЦАП	Показания Вольтметра на входе АЦП	Двоичный код							
			X0	X1	X2	X3	X4	X5	X6	X7
1										
2										
3										
4										
5										
6										
7										
8										

1.2. Исследование аналого-цифровых преобразователей с параллельным преобразованием.

Схема АЦП с параллельным преобразованием представлена на **рис. 2**.

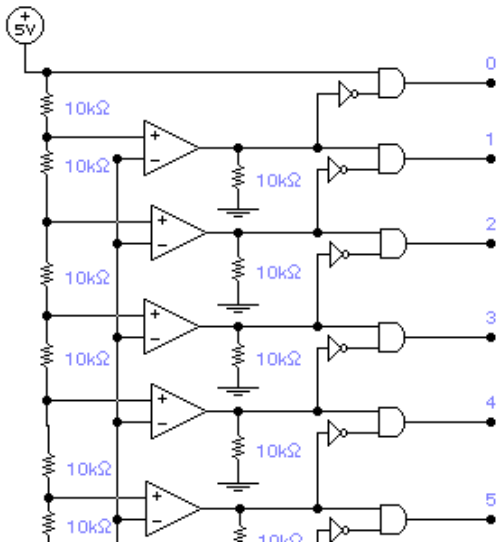


Рис. 2. Схема АЦП с параллельным преобразованием.

В АЦП параллельным преобразованием входной аналоговый сигнал Uo подается на входы всех компараторов и сравнивается с набором опорных напряжений, полученных в резисторном делителе и поданных на вторые входы компараторов. На выходе тех компараторов, на которых входное напряжение Uo меньше опорного, будет сигнал лог. "0". На выходах остальных компараторов будет сигнал лог. "1". Таким образом в АЦП с параллельным преобразованием сигналом о том, что уровень входного аналогового напряжения лежит в одном из интервалов квантования, является переход от лог. "1" к лог. "0" на выходах пары соседних компараторов.

АЦП с параллельным преобразованием является самым быстродействующим. Быстродействие АЦП определяется в основном быстродействием компараторов и может составлять десятки наносекунд. Недостаток АЦП с параллельным преобразованием – аппаратная сложность, так как число компараторов равно 2^n, где n – разрядность АЦП.

На выходе АЦП с параллельным преобразованием устанавливают шифратор, преобразующий сигнал в обычный двоичный код. Начальная часть одного из вариантов такого шифратора показана на **рис. 2**. Выходы каждой пары компараторов подключены к логическим элементам "И", при этом один выход подключен через инвертор. Поэтому открывается только один логический элемент "И", подключенный к выходам компараторов с сигналами лог. "1" и лог. "0", и на одном из выходов АЦП 0, 1, … 7 появится лог. 1. Дальнейшее преобразование проводится в обычном шифраторе.

Развитие схемотехники АЦП привело к созданию ряда промежуточных решений, обладающих меньшей сложностью, чем АЦП с параллельным преобразованием, но быстродействием, лучшим чем у АЦП с последовательным преобразованием. Это, в частности, АЦП с регистром последовательных приближений, АЦП с двухтактным преобразованием и АЦП с двойным интегрированием аналогового сигнала.

Применение АЦП в различных областях привело к созданию трех обособленных групп, отвечающих различным техническим требованиям. Первая группа – сверхбыстродействующие, но не очень точные АЦП с малым числом двоичных разрядов. Вторая группа – АЦП среднего быстродействия и средней точности. Третья группа – медленные прецизионные АЦП с большим числом двоичных разрядов.

1.3.1. Исследование АЦП с параллельным преобразованием средствами САПР.

Схема экспериментального исследования АЦП с параллельным преобразованием представлена на рис.3.

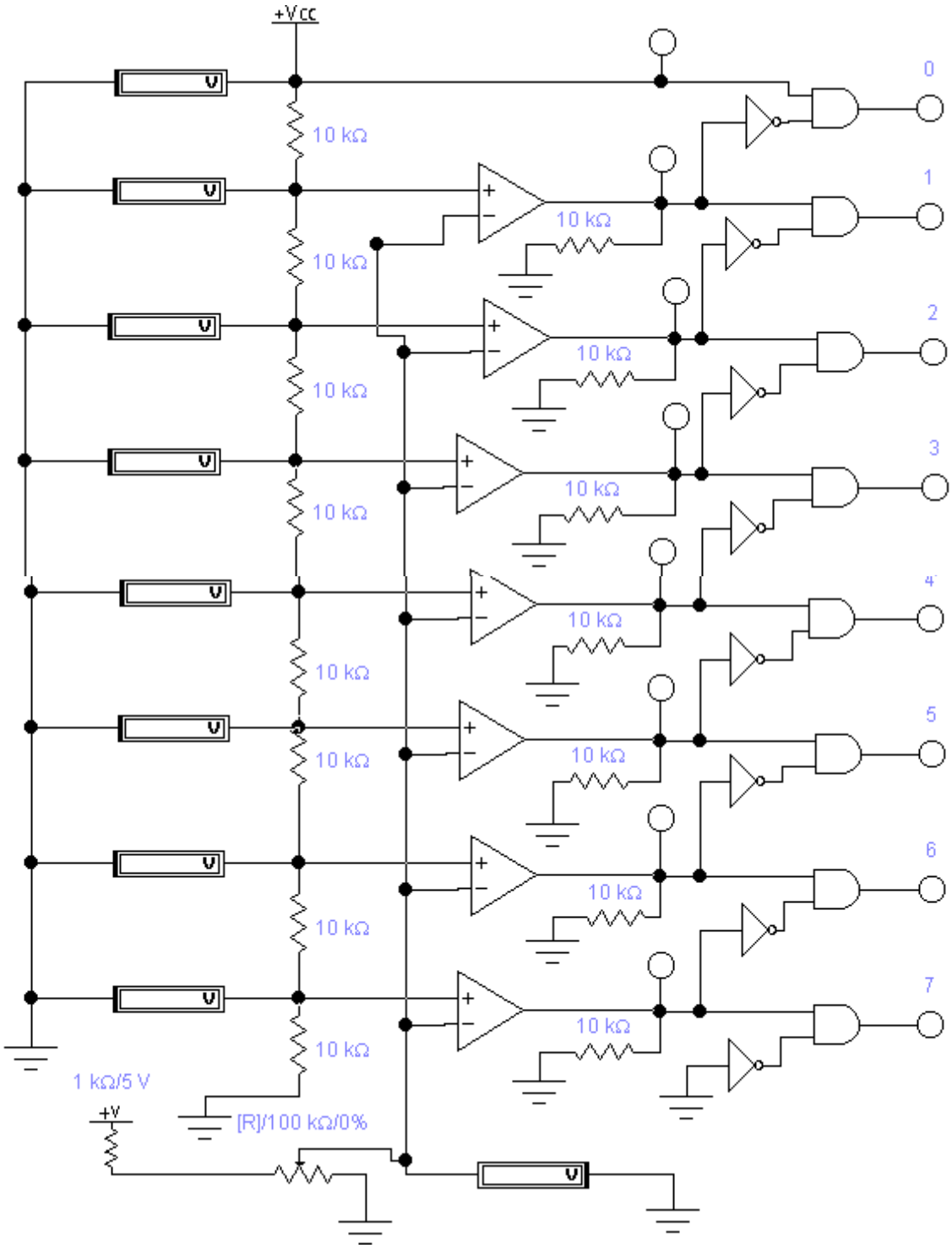


Рис. 3. Схема экспериментального исследования АЦП с параллельным преобразованием.

Напряжение на вход АЦП подается с переменного резистора, управляемого клавишами **R** и **Shift – R**. Положение движка потенциометра в процентах показывается рядом с его изображением. Напряжение на входе АЦП показывает нижний вольтметр. Линейка вольтметров слева на схеме показывает уровни опорного напряжения на входах

компараторов, другими словами, уровни квантования входного сигнала. Логические пробники подключены к выходам компараторов и выходам логических элементов "И".

1. В новом файле соберите схему рис. 3, задавая параметры элементов схемы. Сохраните файл под именем **Z22_02.ewb**.

2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

а) ***Определение минимального шага изменения входного напряжения.***
Для определения минимального шага изменения входного напряжения достаточно вычесть два показания соседних вольтметров, показывающих уровни квантования входного сигнала.

б) ***Изучение процесса преобразования в АЦП с параллельным преобразованием*.**
Установите переменный резистор в положение 100% (при помощи клавиш **Shift – R**). Нажимая на клавишу **R**, переведите резистор в положение 90%. Наблюдайте за работой АЦП по загоранию логических пробников. Обратите внимание на более высокую скорость преобразования и меньшую точность по сравнению с предыдущим экспериментом.

1.3. Применение аналого-цифрового преобразователя для составления таблиц кодирования функциональных генераторов на ЦАП.

При построении функциональных генераторов на ЦАП возникает довольно сложная задача получения последовательности двоичных кодов, управляющих построением заданной функции. Проще всего эта задача решается при помощи АЦП, на вход которого подаются необходимые уровни напряжений, а на выходе получают двоичные коды.

Схема АЦП для решения этой задачи представлена на **рис. 4**. В схеме используется модель 8-ми разрядного АЦП из библиотеки САПР **Electronics Workbench**.

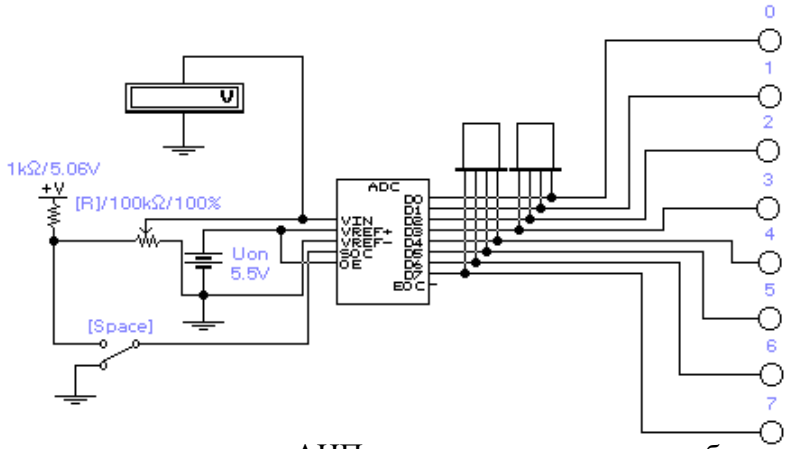


Рис. 4. Схема применения АЦП для составления таблиц кодирования функциональных генераторов.

К выводам АЦП подключаются: VIN – входное аналоговое напряжение, VREF+ и VREF- - источник опорного напряжения, SOC – синхронизация, OE – разрешение на выдачу выходного цифрового кода на выходы D0, … D7. Выход EOS выдает сигнал готовности на вывод выходного цифрового кода.

Входной аналоговый сигнал в АЦП поступает с переменного резистора, управляемого клавишей **R**, и может изменяться в диапазоне от 0 до 5 В. В схеме использован источник опорного напряжения $U_{оп}$ (5.5 В). В качестве источника

синхроимпульсов используется ключ, управляемый клавишей **Space** (пробел). К выходам АЦП подключены семисегментные индикаторы и логические пробники.

1.4.1. Составление таблицы кодирования функционального генератора средствами САПР.

1. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z22_03.ewb**, в котором представлена схема, показанная на **рис. 4**.

Напряжение на вход АЦП подается с переменного резистора, управляемого клавишами **R** и **Shift – R**. Положение движка потенциометра в процентах показывается рядом с его изображением. Величину входного напряжения показывает вольтметр. Логические пробники подключены к выходам АЦП и показывают цифровой код, сопоставляемый аналоговому напряжению на входе АЦП.

В качестве примера составим таблицу кодирования для функции y = ln(x) (**Таблица 2**).

2. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

3. Установите на входе АЦП напряжение, соответствующее одному из значений функции из 2-го столбца **Таблицы 2**. Запустите процесс преобразования при помощи клавиши **Space**. Прочитайте и перепишите в таблицу в Отчете цифровой код на линейке логических пробников.
Заполните всю таблицу кодирования в Отчете.

Таблица 2.

Таблица кодирования для функции y = ln(x).

N п/п	x	ln(x)	Цифровой код							
			X0	X1	X2	X3	X4	X5	X6	X7
1	1	0	0	0	0	0	0	0	0	0
2	10	2.303								
3	20	2.996								
4	30	3.401								
5	40	3.689								
6	50	3.912								

II. Самостоятельная работа.

Задание: Составить таблицы кодирования функционального генератора для функций, представленных в **Калькуляторе** в среде **Windows**.

Последовательность действий:

1. Последовательно нажмите **ПУСК**, **Программы**, **Служебные** и **Калькулятор**.

Вариант 1. Функция y = sin(x)

Вариант 2. Функция $y = \tan(x)$

Вариант 3. Функция $y = \log(x)$

Вариант 4. Функция $y = x^2$

Вариант 5. Функция $y = x^3$

2. Выберите шаг изменения аргумента x, занесите значения аргумента в таблицу в Отчете. Запишите значения функций (значения функций должны лежать в интервале 0.0 – 5.0) во второй столбец таблицы, вычисленные при помощи **Калькулятора**.

Таблица 3.

Таблица кодирования для функции y = f(x).

N п/п	x	f(x)	Цифровой код							
			X0	X1	X2	X3	X4	X5	X6	X7
1										
2										
3										
4										
5										
6										

3. Выберете в меню **Files** команду **Open**. В появившемся диалоговом окне откройте файл **Z22_03.ewb**, в котором представлена схема, показанная на **рис. 4**.

Напряжение на вход АЦП подается с переменного резистора, его сопротивление регулируется клавишами **R –** уменьшение и **Shift** – **R** – увеличение величины сопротивления. Положение движка потенциометра в процентах показывается рядом с его изображением. Величину входного напряжения показывает вольтметр. Логические пробники подключены к выходам АЦП и показывают цифровой код, сопоставляемый аналоговому напряжению на входе АЦП.

4. Запустите процесс моделирования при помощи выключателя в правом верхнем углу экрана.

5. Установите на входе АЦП напряжение, соответствующее одному из значений функции из 2-го столбца **Таблицы 3**. Запустите процесс преобразования при помощи клавиши **Space**. Прочитайте и перепишите в Таблицу Отчета цифровой код на линейке логических пробников.

Заполните всю таблицу кодирования в Отчете.

Лабораторная работа №14-15

Исследование цепей с периодическими несинусоидальными токами

Цель: Овладение практическими навыками моделирования цепей с периодическими несинусоидальными токами, проведения Фурье-анализа (спектрального анализа) с использованием средств САПР Electronics Workbench.

Результат обучения:

После успешного завершения занятия пользователь должен уметь:

- создавать и редактировать простейшие схемы моделирования цепей с периодическими несинусоидальными токами с использованием средств САПР Electronics Workbench;
- проводить Фурье-анализ средствами САПР.

Используемые программы:

Electronics Workbench в. 5.0

Запуск программы:

Предполагается, что требуемые программы уже инсталлированы на диске.

(См. «Инструкцию по установке программы на ПК»)

I. Исследование цепей с периодическими несинусоидальными токами.

1.1. Общие теоретические сведения.

Периодические функции несинусоидальной формы можно представить в виде конечных или бесконечных тригонометрических рядов, называемых рядами Фурье. При этом ряд Фурье представляют в форме

$$F(\alpha) = A_o + \sum_{k=1}^{\infty} A_k \sin(k\alpha\, t + \varphi_k),$$

где $\alpha = 2\pi f_o$ - угловая частота первой гармоники; φ_k - начальная фаза k гармоники.

Если несинусоидальная периодическая функция симметрична относительно оси абсцисс, то постоянная составляющая отсутствует, если несинусоидальная периодическая функция симметрична относительно начала координат, то и постоянная составляющая, и φ_k равны нулю.

1.2. Линейчатый спектр гармонического сигнала.

Задача исследования: Провести Фурье-анализ гармонического сигнала, используя схему его моделирования, представленную на рис. 1.

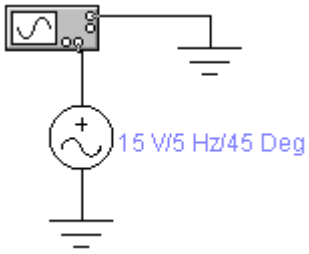


Рис. 1.

1. Запустите при помощи ярлыка на рабочем столе Windows программу **Electronics Workbench**. Соберите схему рис. 1, задайте параметры генератора источника переменного напряжения: на вкладке **Value** установите действующее значение напряжения - 15 В, частоту – 5 Гц, фазу – 45 град и нажмите на кнопку **ОК**.
2. Задайте параметры осциллографа согласно рис. 2

Рис. 2.

3. Запустите процесс моделирования, просмотрите осциллограмму. Убедитесь в том, что осциллограмма несимметрична относительно начала координат.
4. Зададим на схеме рис. 1 режим отображения контрольных точек, в которых анализируется спектр сигнала. В строке меню программы выберите **Circuit,** а в раскрывающемся подменю команду **Schematic Options…** В диалоговом окне **Schematic Options** на вкладке **Show/Hide** в поле **Show nodes** поставьте галочку и нажмите на кнопку **ОК**. Убедитесь в появлении индикации номера контрольной точки на схеме.
5. Для проведения анализа в строке меню программы выберите **Analysis,** а в раскрывающемся подменю команду **Fourier…** В окне настройки Фурье-анализа выбирается основная частота колебаний (частота первой гармоники) f_o, количество гармоник, используемое для разложения исходного сигнала (в данном случае гармонического), в ряд Фурье. Поскольку частота исходного сигнала равна 5 Гц, то примем f_o=5 Гц, а количество гармоник, используемое для разложения исходного сигнала, зададим, например равным 2. Для отображения значений фаз гармоник Фурье преобразования в поле **Display phase** поставьте галочку. В результате окно настройки Фурье-анализа примет вид, представленный на рис. 2

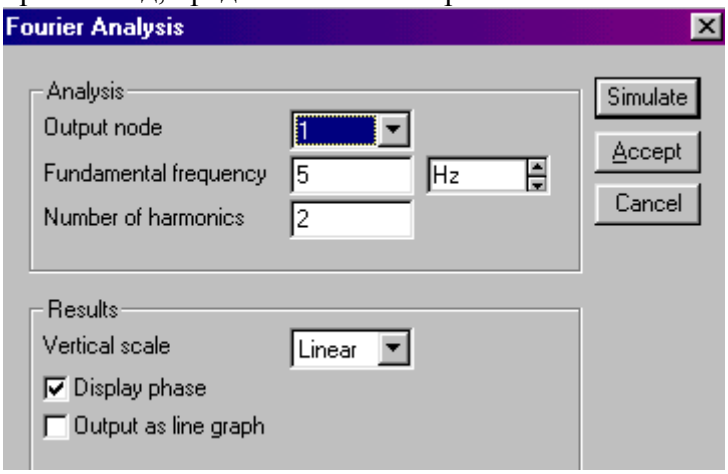


Рис.2.

Нажмите на кнопку **Simulate**. Результаты вычислений появятся в отдельном окне **Analysis Graphs**. На вкладке **Oscilloscope** просмотрите осциллограмму сигнала и

перейдите на вкладку **Fourier**. Результаты вычислений должны иметь вид, показанный на рис. 3.

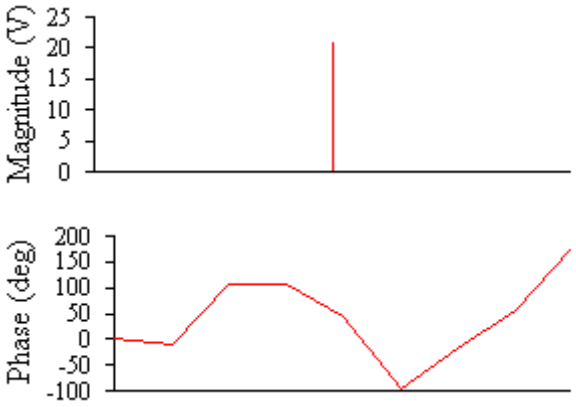


Рис.3.

6. В поле графика распределения амплитуд гармоник щелкните мышкой, а затем нажмите на кнопку  на панели окна **Analysis Graphs**. На графике распределения амплитуд гармоник появятся две визирные линейки и отдельно окно **Magnitude** со значениями частот x1, x2 в точках установки первой и второй визирной линейки, амплитуд гармоник y1 и y2 (см. рис.4).

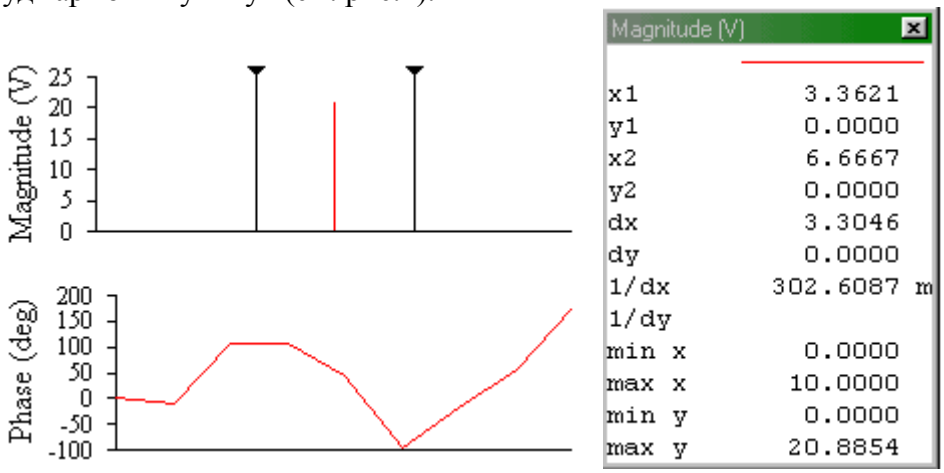


Рис.4.

Совместите первую визирную линейку с линейчатым спектром гармонического сигнала, снимите показания частоты и амплитуды.

7. В поле графика распределения фаз гармоник щелкните мышкой, а затем нажмите на кнопку  . Совместите первую визирную линейку с частотой линейчатого спектра гармонического сигнала – 5 Гц (значение частоты, соответствующее положению визирной линейки, наблюдайте в окне **Phase**) и снимите показания фазы. Закройте окно Фурье-анализа. Сравните полученные результаты с исходными данными.

1.3. Фурье-анализ треугольного сигнала.

Задача исследования: Провести Фурье-анализ треугольного сигнала с амплитудой 21 В, частотой – 5 Гц.

1. Схему рис.1 дополните функциональным генератором, так как показано на рис.5.

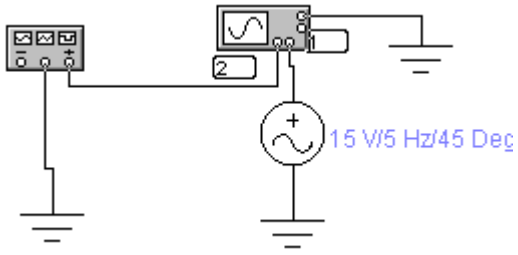


Рис. 5.

2. Настройте параметры функционального генератора согласно рис.6.

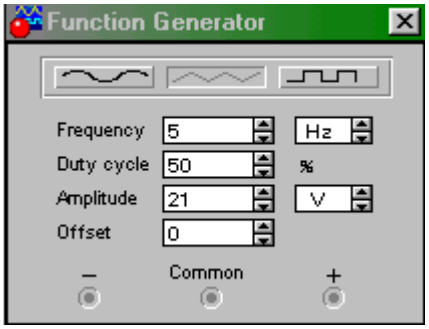


Рис. 6.

3. Для проведения анализа в строке меню программы выберите **Analysis,** а в раскрывающемся подменю команду **Fourier…** В окне настройки Фурье-анализа в раскрывающемся списке **Output node** выберите контрольную точку разветвления на схеме **2**. Поскольку частота исходного сигнала равна 5 Гц, то примем f_o=5, а количество гармоник, используемое для разложения исходного сигнала, в ряд Фурье зададим в первом приближении равным 4. Нажмите на кнопку **Simulate**. Зарисуйте линейчатый спектр в Отчет, снимите показания, занесите их в Отчет. Аналогично постройте линейчатый спектр для количества гармоник равного 10. Результаты вычислений занесите в Отчет, запишите ряд Фурье. Сделайте вывод о влиянии увеличения количества гармоник, используемого для разложения треугольного сигнала, на вид ряда Фурье.

II. Самостоятельная работа.

Задание №1. Провести Фурье-анализ прямоугольного сигнала.

Исходные данные:

- амплитуда сигнала 21 В;
- частота:
  а) 5 Гц;
  б) согласно варианту

Вариант	1	2	3	4	5
f, Гц	2	10	8	20	15

Результаты измерений и расчетов, полученный ряд Фурье занесите в Отчет.

Задание №2. Провести Фурье-анализ сигнала $u(t)^2$,
где $u(t)$ - треугольный сигнал с амплитуда сигнала 21 В, частотой *f*.

Вариант	1	2	3	4	5
f, Гц	2	10	8	20	15

Схема для моделирования сигнала $u(t)^2$ представлена на рис.7.

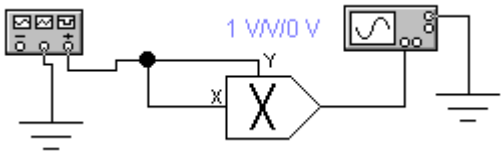


Рис. 7.

Результаты измерений и расчетов, полученный ряд Фурье занесите в Отчет.

Примечание: Блок умножения находится в библиотеке аналоговых вычислительных устройств  .

Задание №3. Провести Фурье-анализ сигнала $u(t)^2$,
где $u(t)$ - гармонический сигнал с амплитуда сигнала 21 В, частотой *f*.

Вариант	1	2	3	4	5
f, Гц	2	10	8	20	15

Схема для моделирования сигнала $u(t)^2$ представлена на рис.7. Результаты измерений и расчетов, полученный ряд Фурье занесите в Отчет.

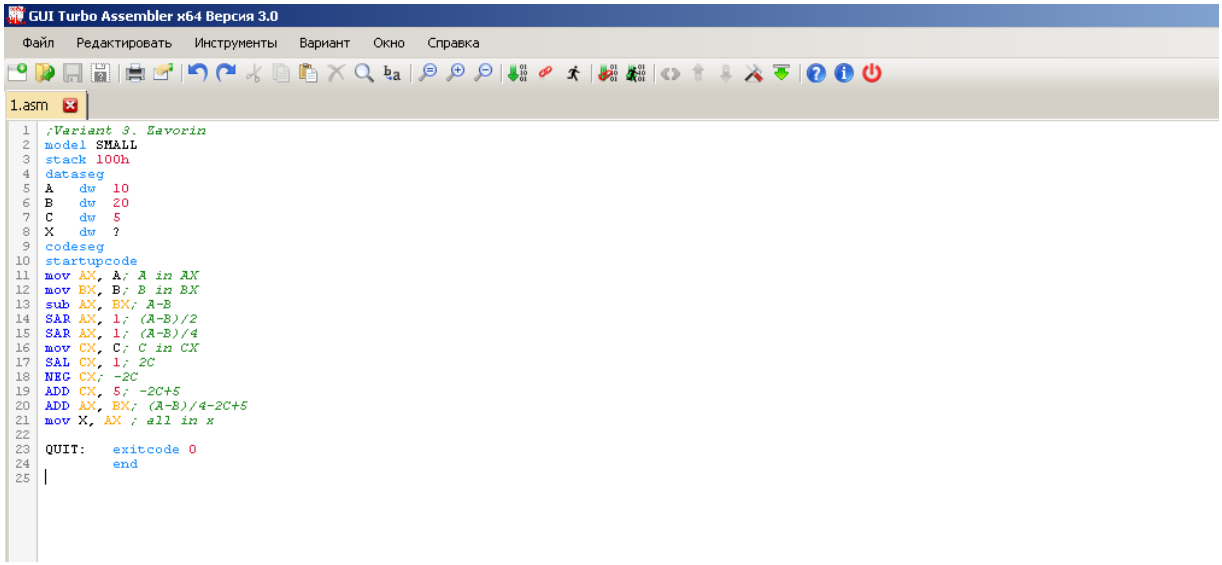
Лабораторная работа 16.

Изучение языка низкого уровня. Ассемблер.

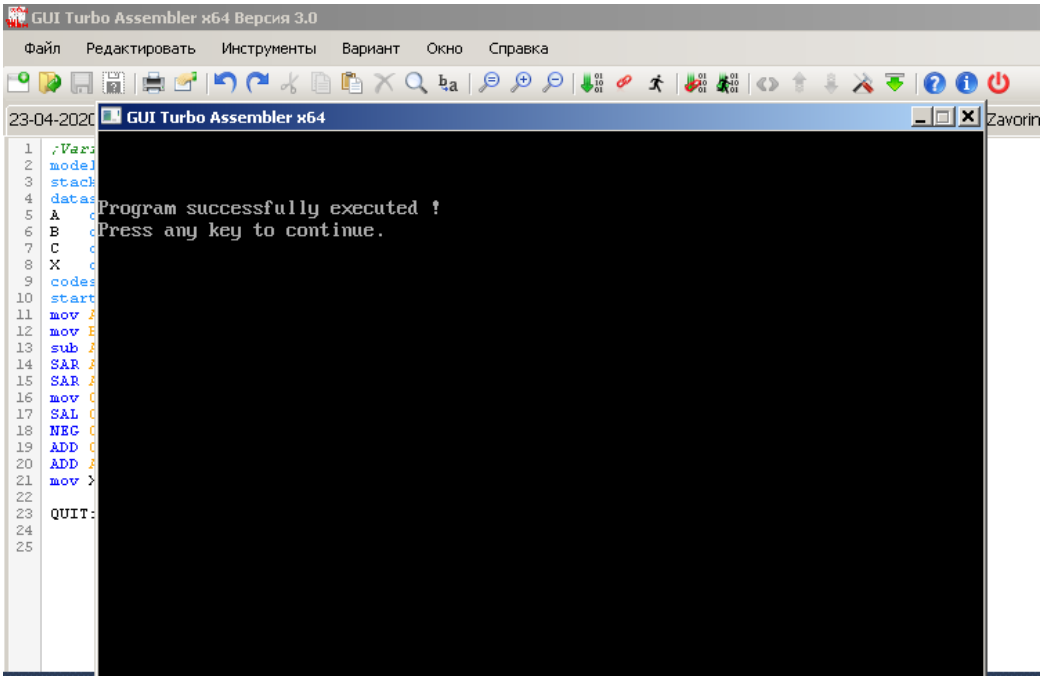
1. Разработать программу на языке программирования Intel 8086, реализующую указанную формулу, исполнить программу с несколькими наборами исходных данных, проверить правильность результатов X=-8A+(B+C)/4+2.
2. Разработать программу на языке программирования Intel 8086, реализующую указанную формулу, исполнить программу с несколькими наборами исходных данных, проверить правильность результатов X = (A/2 +B)/4+C-1.
3. Разработать программу на языке программирования Intel 8086, реализующую указанную формулу, исполнить программу с несколькими наборами исходных данных, проверить правильность результатов X = (A - B) / 4 - 2C + 5.

Пример

Разработать программу, реализующую указанную формулу, исполнить программу с несколькими наборами исходных данных, проверить правильность результатов X = (A - B) / 4 - 2C + 5




```
model SMALL
stack 100h
dataseg
A   dw  10
B   dw  20
C   dw  5
X   dw  ?
codeseg
startupcode
mov AX, A; A in AX
mov BX, B; B in BX
sub AX, BX; A-B
SAR AX, 1; (A-B)/2
SAR AX, 1; (A-B)/4
mov CX, C; C in CX
SAL CX, 1; 2C
NEG CX; -2C
ADD CX, 5; -2C+5
ADD AX, CX; (A-B)/4-2C+5
mov X, AX ; all in x
QUIT:   exitcode 0
end
```



Программа собрана и включается.

По итогу при

1. A = 10, B=20, C = 5 в программе должно получиться X = -7,5
2. A = 20, B=10, C = 5 в программе должно получиться X = 2,5
3. A = 5, B=20, C = 10 в программе должно получиться X = 11,25

Лабораторная работа 17

Пример 1

Ввести в ячейки 120,123,... числа 7,B,...,10F (все данные в 16-ой системе)

Решение

1. Задание начальных значений
(начального числа= 7,
стартового адреса= 120,
числа шагов цикла= 43).

Примечание. Расчет значения счетчика цикла:

CX= (10F - 7):4 + 1 = 108:4 + 1 = 42 + 1 = 43

```
 10F 108¦ 4            16-ая          10-ая
-  7   10 ¦--     1)     10:4 = 4   16:4 = 4
 108 --- 42       2)      8:4 = 2    8:4 = 2
        08
        08
        --
         0
```

3. Завершение работы программы (INT 20)

КОД ПРОГРАММЫ:

```
100 MOV AL,07
102 MOV DI,0120
105 MOV CX,0043

111 INT 20      ; завершение работы
```

120,123,126,... числами 7,B,F,...,10F)

2. Организация цикла: [DI] <- AL
(последовательное заполнение ячеек

```
            ; цикл по перебору ячеек диапазона:
108 MOV [DI],AL     ;[DI] <- AL       (размещение в очередной ячейке памяти очередного числа)
10A ADD AL,04       ; AL  <- AL + 4 (наращивание числа: +4)
10C ADD DI,0003     ; DI  <- DI + 3   (наращивание адреса: +3)
10F LOOP 108        ; CX  <- CX - 1 После чего, если CX <> 0, следует переход по адресу 108 (команда цикла)

                        ; AL  <- 7 (начальное число = 7)
                        ; DI  <- 120      (стартовый адрес = 120)
                        ; CX  <- 43       (число шагов цикла= 43)
```

Проверка

Для проверки решения уменьшим значение счетчика цикла CX до 3.

РЕЗУЛЬТАТ РЕШЕНИЯ:

1-ый шаг: AL = 7
2-ой шаг: DI = 120
3-ий шаг: CX = 3

4-ый шаг: [120] = 7
5-ый шаг: AL = 7 + 4 = B
6-ой шаг: DI = 120 + 3 = 123
7-ой шаг: CX = 3 - 1 = 2 (2 > 0, поэтому следует переход на адрес 108)
8-ой шаг: [123] = B
9-ый шаг: AL = B + 4 = F
10-ый шаг: DI = 123 + 3 = 126
11-ый шаг: CX = 2 - 1 = 1 (1 > 0, поэтому следует переход на адрес 108)
12-ый шаг: [126] = F
13-ый шаг: AL = F + 4 = 13
14-ый шаг: DI = 126 + 3 = 129
15-ый шаг: CX = 1 - 1 = 0 (0 = 0, цикл завершен)

16-ый шаг: INT 20   - завершение работы.

Пример 2. Просмотреть ячейки 120..16F и заменить символ “9” символом “5”
Решение

```
100 MOV DI, 0120
10 MOV AL, 35
105 MOV AH, 39
107 MOV CX, 0050
10A CMP [DI], AH
10C JNZ 0110
10E MOV [DI], AL
110 INC DI
111 LOOP 010A
113 INT 20
```

Задания

1. Ввести в ячейки 120..130,… числа 7,B,F…,(все данные в 16-ой системе)

2. Просмотреть ячейки 140..150 и подсчитать количество символов “3”. Резльтат поместить в ячейку 170.

Лабораторная работа №18

Исследование организации переходов и циклов

1 Цель работы

Изучение команд передачи управления и получение практических навыков отладки разветвляющихся программ.

2 Теоретический материал

2.1 Кодировки символов

Вычислительная машина может обрабатывать не только числа, но и текстовую информацию, представляющую последовательность символов. Под символами подразумеваются буквы алфавита, цифры, знаки препинания, служебные символы и т. д. При этом каждому символу ставиться в соответствие определенная двоичная кодовая комбинация. Соответствие между набором символов и их кодами называется кодировкой символов, а совокупность возможных символов и соответствующих им двоичных кодов образуют **таблицу кодировки**. В настоящее время существует множество таблиц кодировок. Общим для них является весовой принцип, согласно которому коды цифр возрастают с увеличением цифры, а коды букв увеличиваются в алфавитном порядке.

Наряду с **16-и битовой кодировкой Unicode**, которая используется с 1993 года и определяет универсальный набор символов (**UCS, Universal Character Set**), весьма популярными являются восьмиразрядные кодировки. Их недостаток, заключающийся в числе кодируемых символов не превышающим 256, заставил разработчиков создать несколько модификаций таблиц кодировок. Например, американский стандартный код для обмена информацией **ASCII** (**A**merican **S**tandard **C**ode for nformation **I**nterchange) имеет европейские модификации (стандарт **ISO 8859**) для разных языков. Кодировка для русского языка представлена в таблице 7.3.

Коды всех символов этой таблицы представлены в десятичной (**d**) и шестнадцатеричной (**h**) системах. Первые 32 кода (1…32) имеют два значения: управляющие символы и изобразительные символы. Когда DOS пересылает эти коды на монитор или принтер, они выполняют управляющие функции, а не отображают символы, например, 8 – возврат на одну позицию, 9 –горизонтальная табуляция, 10 – перевод строки, 13 – возврат каретки, 32 – пробел. Для получения изображения символов эти коды необходимо занести в буфер экрана (начальный адрес 0B800:0000H).

Для определения кода какого-либо символа из таблицы следует

3.

выполнить простые действия:

– для шестнадцатеричных значений – записать старшую и младшую шестнадцатеричные цифры, которые определяют положение клетки таблицы с нужным символом. Например:**Q= 51h**;

– для десятичных значений – найти сумму десятичных чисел, которые определяют положение клетки с нужным символом. Например: **Q = 80d + 1d = 81d**.

2.2 Команды передачи управления

Адрес следующей выполняемой микропроцессором команды определяется содержимым его регистров **CS:IP** (**EIP** в случае 32-х разрядной адресации). Если все команды находятся в одном сегменте кодов, то при переходе к следующей команде изменяется только содержимое **указателя команд IP**.

В линейных алгоритмах, когда все команды выполняются в том порядке, в котором они записаны, **содержимое IP** автоматически увеличивается на число, равное длине выполняемой команды.

В алгоритмах с ветвлением приходится изменять линейный ход программы. Для этого предназначены **команды передачи управления**. Они подразделяются на команды:

- безусловных переходов;
- условных переходов;
- вызовов;
- возвратов;
- управления циклами;
- прерываний.

При выполнении команд передачи управления флаги не устанавливаются.

Адрес команды, которой передается управление, в ассемблере задается с помощью **метки**, которая является символическим именем команды. С меткой связывается определенная ячейка памяти, в которой находится первый байт помеченной команды.

Команда безусловного перехода имеет мнемонику JMP (JuMP) и записывается в формате

JMP [модификатор] метка; перейти на метку.

В зависимости от того, где находится помеченная команда, машинные коды команд **JMP** различаются и имеют пять разновидностей:

- **прямой короткий переход** (**short**) – адрес перехода лежит в диапазоне **–128 …+127**.

4.

- **прямой ближний переход** (**near**) – помеченная команда находится в текущем сегменте кода на расстоянии **128 …2^{16} адресов** от команды JMP;
- **прямой косвенный ближний переход** – адрес перехода задается косвенно с помощью ссылки на регистр или ячейку памяти в которых он находится;
- **прямой дальний переход** (**far**)– помеченная команда находится в другом сегменте кодов. Изменяется содержимое регистров CS:IP (EIP);
- **прямой косвенный дальний переход** – полный адрес перехода **CS:IP** (**EIP**) содержится в ячейках памяти.

Иногда в программах имеет смысл указывать тип перехода с помощью **модификатора**:

- **short** – прямой короткий переход;
- **near ptr** – прямой ближний переход;
- **far ptr** – прямой дальний переход;
- **wordptr** – косвенный ближний переход;
- **dword ptr** – косвенный дальний переход.

Примеры:

```
JMP  m_1;              ;перейти на метку m_1
JMP near ptr m_1;      ;перейти на метку m_1
MOV BX, offset m_2;    ;загрузить в BX адрес ближнего перехода
JMP BX;                ;перейти на метку адрес которой находится в BX
```

Команды условных переходов организуют передачу управления на метку в случае выполнения заданного условия. **Если заданное условие не выполняется, то происходит переход к следующей по порядку команде**. Команды не влияют на формирование флагов. Обобщенный формат команд условного перехода имеет вид:

Jcond метка;

Jump - прыжок, **Cond**ition – условие, **метка** – метка перехода, которая может находиться только в **текущем сегменте кода**. Отсюда следует, что **все условные переходы являются короткими или ближними** внутрисегментными.

Условия, на основании которых формируется решение о переходе, определяются состоянием флагов микропроцессора. Поэтому при использовании команд условного перехода необходимо следить за тем, чтобы флаги находились в активном состоянии. Активизиро-

5.

вать флаги можно выполнением «безобидной» арифметической команды, либо команды **сравнения CMP**.

Команды **условных переходов по флагам** используют в качестве условия один из арифметических флагов результата операции:

JC метка	;перейти на метку, если **CF = 1**
JNC метка	;перейти на метку, если **CF = 0**
JP метка	;перейти на метку, если **PF = 1**
JNP метка	;перейти на метку, если **PF = 0**
JZ метка	;перейти на метку, если **ZF = 1**
JNZ метка	;перейти на метку, если **ZF = 0**
JS метка	;перейти на метку, если **SF = 1**
JNS метка	;перейти на метку, если **SF = 0**
JO метка	;перейти на метку, если **OF = 1**
JNO метка	;перейти на метку, если **OF = 0**

В мнемоническом обозначении команды второй или третий символ обозначают один из арифметических флагов, **единичное** значение которого используется в качестве условия. Символ **N** - **N**ot означает, что в качестве условия используется **нулевое** значение арифметического флага.

Команды условных переходов по результатам операции сравнения (см. таблицу 7.1) позволяют формировать условия перехода на основе анализа нескольких признаков (флагов), сформированных командой **сравнить CMP**. Такие команды принято делить на команды для анализа беззнаковых чисел (к ним применяется термины **выше** (**a**bove) / **ниже** (**b**elow)) и для анализа чисел со знаком (термины больше (**g**reater) / меньше (**l**ess)). В случае равенства операндов используют термин **равно – e**qual.

Примеры:

```
CMP AX, 0          ;сравнить содержимое AX с 0
JE m_6             ;если равно, перейти на метку m_6
CMP BX, 1024       ;сравнить содержимое AX с числом 1024
JA m_2             ;если выше, перейти на метку m_2.
```

2.3 Организация циклов в Ассемблере

Для организации многократного выполнения некоторого участка программы используется конструкция, называемая циклом. Для организации циклов необходимо выполнить следующую последовательность действий:

Таблица 7.1

Команды перехода по результатам сравнения

Типы операндов	Мнемоника	Условия перехода	Значения флагов
любые	**JE**	op.1 = op.2(равно)	**ZF = 1**
любые	**JNE**	op.1 ≠ op.2 (не равно)	**ZF = 0**
со знаком	**JL/JNGE**	op.1 < op.2 (меньше)	**SF≠OF**
со знаком	**JLE/JNG**	op.1 ≤ op.2 (меньше или равно)	**SF ≠ OF или ZF = 1**
со знаком	**JG/JNLE**	op.1 > op.2(больше)	**SF = OF и ZF = 0**
со знаком	**JGE/JNL**	op.1 ≥ op.2 (больше или равно)	**SF = OF**
без знака	**JB/JNAE**	op.1 < op.2(ниже)	**CF= 1**
без знака	**JBE/JNA**	op.1 ≤ op.2 (ниже или равно)	**CF= 1 или ZF = 1**
без знака	**JA/JNBE**	op.1 > op.2(выше)	**CF= 0 и ZF = 0**
без знака	**JAE/JNB**	op.1 ≥ op.2 (выше или равно)	**CF= 0**

1) перед входом в цикл задать начальные значения переменных, которые изменяются в цикле (параметры цикла);
2) изменять эти переменные перед каждым новым повторением цикла;
3) пометить первую команду тела цикла меткой;
4) проверять условие окончания или повторения цикла;
5) управлять циклом, т. е. переходить к его началу или выходить из него по окончании.

В ассемблере для организации циклов используются специальные команды, относящиеся к командам передачи управления, которые позволяют организовать описанную выше последовательность действий простыми средствами. В качестве **счетчика числа повторений цикла используется регистр CX**, поэтому **максимальное число повторений составляет 2^{16}**. Перед началом цикла регистр CX инициализируется.

Для организации проверки условия окончания и управления циклом используется **команда LOOP метка** (повторить цикл). При выполнении этой команды микропроцессор производит следующие действия:

- содержимое счетчика циклов (регистра CX) **уменьшается на 1** (декремент CX);
- содержимое **CX сравнивается с «0»**;
- если **содержимое CX больше «0»**, осуществляется переход к началу цикла, который должен иметь **метку**. В противном случае выполняется выход из цикла.

Команда **LOOP** позволяет организовать только короткие переходы к началу цикла (в пределах –128 …+127 адресов) и это является ее недостатком. Поэтому для организации длинных циклов используются команды условных и безусловных переходов, которые были рассмотрены.

Команды LOOPE и **LOOPZ** позволяют повторять цикл пока содержимое регистра **CX не равно «0»** и **флаг нуля ZF** установлен в «1».

Команды LOOPNE и **LOOPNZ** позволяют повторять цикл пока содержимое регистра **CX не равно «0»** и **флаг нуля ZF** установлен в «0».

Эти команды отличаются от команды **LOOP** тем, что при их выполнении дополнительно анализируется флаг **нулевого результата ZF**. Это позволяет организовать досрочный выход из цикла, используя этот флаг в качестве индикатора.

2.4 Определение длины символьной переменной

При работе с символьными строками часто требуется знать их длину. Для вычисления длины символьной строки используется простой прием:

в **сегменте данных**, где объявлена строка, следует записать математическое выражение, которое при трансляции будет вычисляться и записываться в константу **length**:

```
.DATA
 TEXT  DB ‘ Turbo Assembler’, 13, 10, ‘$’   ;объявить строку
 length = ($ - TEXT) – 3                    ;вычислить длину
                                            ;символьной строки
```

Поскольку переопределенный символ $ является счетчиком символов в строке, разность **($ - TEXT)** определяет длину символьной строки. Однако в объявленной строке присутствует на 3 символа больше: два управляющих символа **13 – перевод строки**, **10 – возврат каретки** и завершающий строку символ **‘$’**. Поэтому найденную длину строки следует уменьшить на 3 байта.

2.5 Алгоритм программы CHANGE

Алгоритм программы замены строчных букв заглавными основан на анализе каждого символа исходной строки с целью определения принадлежности его к строчным или заглавным. Если символ строчный, то его код изменяется с помощью процедуры COR. Анализ происходит в цикле с числом повторений равным числу символов в строке.

Определение принадлежности символа к строчным или к заглавным происходит на основе таблицы 7.3 с **ASCII кодами**. Из нее следует, что строчные английские символы занимают диапазон кодов **61h…7Ah**. Поэтому коды символов, которые попадают в этот диапазон нужно корректировать. Коды символов не попадающих в этот диапазон корректировать не следует.

3 Подготовка к работе

3.1. Изучить методические указания и рекомендованную литературу.

3.2. Подготовить ответы на контрольные вопросы.

4 Задание на выполнение работы

4.1 Проанализировать приведенную ниже программу **CHANGE**. Создать и отладить исполняемый модуль программы **CHANGE**, выполнив этапы ассемблирования и компоновки. Добавить в исходный модуль программы недостающие комментарии.

```
TITLE CHANGE
;Программа заменяет строчные буквы заглавными в символьной
;строке и выводит на экран преобразованную строку на экран
;Входные параметры:
;текстовая переменная MYTEXT
.MODEL SMALL
.STACK 256
.DATA
 MYTEXT  DB ‘Our NativeTown’, 13, 10, ‘$’   ;объявляем текстовую
                                            ;переменную
;---------------------- процедура коррекции кода символа ----------------------
COR PROC NEAR
NOP
        AND AH, 0DFh
MOV [BX], AH
RET
COR ENDP
```

```
;-----------------------------основная программа-----------------------------------
Start:
        MOV AX, @DATA
        MOV DS, AX
        XOR AX, AX
        LEA BX, MYTEXT
        MOV CX, 15                      ;загрузить счетчик циклов
;--------------------------------начало тела цикла---------------------------------
MT1: MOV AH, [BX]
        CMP AH, 61h
        JB MT2
        CMP AH, 7Ah
        JA MT2
        CALL COR
MT2:  INC BX
;-----------------------конец тела цикла-------------------------------------------
LOOP MT1                                ;повторить цикл, если (CX) ≠0
        LEA DX, MYTEXT                  ;вывести переменную
        MOV AH, 09h                     ;MYTEXT
        INT 21h                         ;на экран
        NOP                             ;холостая команда
        MOV AX, 4C00h                   ;завершить
        INT 21h                         ;программу
   END  Start
```

4.2 Загрузите созданную программу в отладчик. В окне **Watches** введите имя символьной переменной **MYTEXT**. Произведите пошаговое выполнение, используя клавиши **F7** и **F8**. Обратите внимание, что использование **F7** позволяет пошагово выполнять команды внутри цикла, в то время как **F8** инициирует однократное выполнение цикла полностью. Наблюдайте и анализируйте результаты выполнения команд и изменения, происходящие с переменной в окне **Watches**.

4.3 Заново загрузите программу в отладчик. Клавишей **F2** установите ловушку (точку останова) на команде **NOP**, следующей после команд вывода символьной строки на экран. Запустите выполнение

программы клавишей **F9**. Программа должна выполниться до указанной точки останова.

4.4 Перейдите в окно **WINDOW** отладчика и пронаблюдайте результат выполнения программы, выбрав режим **USER SCREEN**. Повторным нажатием F9 продолжите выполнение программы до ее окончания.

4.5 Внесите изменения в программу добавив автоматическое вычисление длины строки и загрузку вычисленным значением счетчика циклов CX.

4.6 Проверьте работу модифицированной программы **CHANGE** для другой символьной строки, содержащей не менее 20 строчных и заглавных букв английского алфавита.

4.7 Создайте и отладьте программу **CHANGE_1**, чтобы она обеспечивала выполнение задания в соответствии с вариантом из таблицы 7.2. Предусмотрите вывод на экран исходных и модифицированных строк текста.

Таблица 7.2

Варианты заданий

№ варианта	Заменить	№ варианта	Заменить
1	а) “a” на “A”, “s” на “S” б) все заглавные строчными	9	а)“1” на “A”, “2” на “B” б) все заглавные строчными
2	а) от “a” до “f ” на “A” до “F” б) все заглавные строчными	10	а)от “d” до “j” на “D” до “J” б) все заглавные строчными
3	а) “b” и “c ” заглавными б) все заглавные строчными	11	а)“D” на “J”, “d” на “j” б) все заглавные строчными
4	а) от “f” до “z” заглавными б) все заглавные строчными	12	а)“3” на “C”, “4” на “D” б) все заглавные строчными
5	а) символ “ (” на “) ” б) все заглавные строчными	13	а)все пробелы на “!” б) все заглавные строчными
6	а) “y” на “Y”, “z” на “Z” б) все заглавные строчными	14	а) все пробелы на “=” б) все заглавные строчными
7	а) “o” на “O”, “r” на “R” б) все заглавные строчными	15	а) “s” на “S”, “t” на “T” б) все заглавные строчными
8	а) “Y” на “y”, “Z” на “z” б) все заглавные строчными		

Лабораторная работа №19

Работа с подпрограммами и процедурами

1 Цель работы

Практическое овладение навыками составления и отладки подпрограмм и процедур на языке Ассемблера.

2 Теоретический материал

2.1 Подпрограммы и процедуры

Подпрограммы и процедуры являются одним из средств разработки модульных программ. Они представляет собой законченную командную последовательность к которой можно обращаться из любого места программы с помощью команд вызова **CALL**.

Достоинства использования подпрограмм-процедур объясняется следующим:

– сложную программу можно разбить на сравнительно небольшие и достаточно простые модули, разработка и отладка которых может производиться автономно несколькими программистами;

– в виде процедур оформляются фрагменты программ, которые используются многократно, следовательно использование процедур сокращает длину программ;

– из отлаженных процедур формируются библиотеки, которые можно использовать в других программах.

Организуются **подпрограммы** очень просто:

```
метка:                  ;начало подпрограммы
      ..................
      команды, входящие
      в подпрограмму
      (тело подпрограммы)
      ..................
RET                     ;возврат из подпрограммы.
```

– первая команда подпрограммы помечается **меткой**, которая служит точкой входа в подпрограмму;

– завершается подпрограмма командой возврата **RET**.

– вызов подпрограммы производится командой вызова **CALL**, которая имеет следующий формат:

CALL <метка>.

6.

Любую **подпрограмму** можно оформить в виде **процедуры**. При этом следует использовать следующие правила:

– Начинается процедура директивой **PROC** (Procedure), а завершается директивой **ENDP** (END Procedure), между которыми располагается тело процедуры.

– Директивы начала и окончания процедуры обязательно должны иметь **имя**, которое является именем процедуры. Имя используется как метка первой команды процедуры и означает точку входа в процедуру.

– Некоторые процедуры могут требовать передачу из вызывающей программы входных параметров и возвращать в нее результаты вычислений. В этом случае их нужно оформлять в виде списка входных и возвращаемых параметров (см. п. 2.3).

```
имя PROC [тип]                    ;начало процедуры
    [ARG список аргументов]       ;входные параметры
    [RETURN список элементов]     ;возвращаемые параметры
.....
    тело
    процедуры
.....
имя ENDP                          ;конец процедуры.
```

– После ключевого слова **PROC** указывается **тип процедуры NEAR** (ближняя) или **FAR** (дальняя) (по умолчанию принимается тип **NEAR**). **Если процедура находится в том же сегменте кода, что и вызывающая команда, она относится к типу NEAR. Если процедура и вызывающая команда находится в разных сегментах, то она относится к типу FAR.**

– Для вызова процедуры используется команда вызова **CALL**, которая имеет следующий формат:

CALL <имя процедуры>.

Если вызываемая процедура относится к типу **NEAR**, то генерируется короткая команда внутрисегментного вызова, т. е. машинный код команды **CALL** содержит внутрисегментное смещение точки входа в процедуру, а **адрес возврата** (содержимое указателя команд **IP**) **автоматически помещается в стек**.

Если вызываемая процедура относится к типу **FAR**, то генерируется длинная команда межсегментного вызова, т. е. машинный код команды **CALL** содержит сегментный адрес и внутрисегментное сме-

щение точки входа в процедуру, а **адрес возврата** (содержимое регистра **CS** и указателя команд **IP**) **автоматически помещается в стек**.

– Тело процедуры должно завершаться безоперандной командой возврата **RET**. Эта команда восстанавливает в регистрах микропроцессора запомненный в стеке **адрес возврата**. В процедурах типа **NEAR** восстанавливается содержимое указателя команд – регистра **IP**, а в процедурах типа **FAR** содержимое двух регистров **CS** и **IP**.

Все сказанное для команд вызова **CALL**и возврата **RET** справедливо и для организации подпрограмм.

Элементарный пример процедуры, которая заносит в регистр **CX** среднее значение содержимого регистров **AX** и **DX** приведен ниже:

```
Average PROC NEAR      ;начать процедуру с именем Average
    MOV CX, AX         ;переслать  (AX) в (CX)
    ADD CX, DX         ;сложить (CX) и (DX)
    RCR  CX, 1         ;сдвинуть (CX) вправо на один разряд
    RET                ;возврат из процедуры
Average ENDP           ;закрыть процедуру именем Average
```

Вызов этой процедуры производится из любого места программы командой **CALL Average**.

Поскольку и в вызывающей программе, и в процедурах используются одни и те же регистры микропроцессора, важно, чтобы при возврате в вызывающую программу они оказались немодифицированными. Это решается выполнением двух действий:

– сохранение содержимого необходимых регистров в стек в самом начале процедуры. Для этого используются команды **PUSH** или команда **PUSHA** (в случае использования расширенной системы команд);

– извлечения из стека содержимого всех сохраненных регистров с помощью команд **POP** или **POPA** (в случае использования расширенной системы команд).

2.2 Размещение процедур в программе

Размещаться процедуры могут в любом месте программы, однако программист должен принимать соответствующие меры, защищая ее от несанкционированного выполнения. Самым простым способом такой защиты являются расположение процедур выше тех программ, которые их вызывают, а также отсутствие вложенных процедур (когда одна процедура полностью находится внутри другой).

В виде одной процедуры можно оформить сегмент кодов несложной программы, например так, как в приводимом ниже примере.

8.

При этом **команда возврата RET – отсутствует, а имя этой процедуры должно быть указано в директиве END, завершающей исходный модуль программы.**

```
TITLE Prog_1          ;Пример регистровых операций.
.MODEL SMALL          ;Модель памяти ближнего типа.
.STACK 100h           ;Отвести под стек 256 байт.
.CODE                 ;Открыть сегмент кодов.
Begin PROC NEAR       ;Начать процедуру с именем Begin.
PUSH DS               ;Сохранить DS в стеке.
SUB AX, AX            ;Очистить AX.
PUSH AX               ;Сохранить AX в стеке.
MOV AX, 0123h         ;Передать константу в регистр.
ADD AX, 0025h         ;Сложить (AX) с константой.
MOV BX, AX            ;Передать из регистра в регистр.
ADD BX, AX            ;Сложить содержимое регистров AX и BX.
MOV CX, BX            ;Передать из регистра в регистр
SUB CX, AX            ;Вычесть (AX) из (CX).
NOP                   ;Нет операции (задержка).
   MOV AX, 4C00h      ;Выход
   INT 21h            ;в DOS.
Begin ENDP            ;Закрыть процедуру.
END  Begin            ;Закрыть программу.
```

2.3 Входные и выходные параметры процедур

Процедура обычно разрабатывается как функциональный блок, который формирует набор выходных данных путем преобразования определенного набора входных данных. Значения, которые процедура использует как входные данные, называются **параметрами**. Параметрами могут быть численные значения, адреса, содержимое ячеек памяти и т. д. Имеется несколько способов передачи параметров процедурам. Один из простых способов заключается в том, чтобы значения параметров разместить в регистрах микропроцессора, например так, как это сделано в процедуре **Average**. Два входных параметра передаются процедуре через регистры **AX** и **DX**. Эти параметры должны быть загружены в указанные регистры до вызова процедуры.

Еще один способ передачи параметров связан с использованием общей области данных, доступной и вызывающей программе и процедуре. В большинстве программ на языках высокого уровня передача параметров процедурам организуется через стек. В ассемблерных программах такой способ организуется также сравнительно просто. Выбор способа передачи параметров процедуре зависит от конкретных функций выполняемых ею.

9.

Процедура, которая возвращает одно значение, называется **процедурой-функцией**. Примером такой процедуры является рассмотренная **Average**, которая возвращает среднее значение в регистр **CX**. Чтобы упростить объединение исходных модулей, составленных на языках высокого уровня и ассемблере, желательно использовать следующее соглашение:

– значение переменной типа **WORD** возвращается в аккумуляторе **AX**;

– значение переменной типа **BYTE** возвращается в аккумуляторе **AL**;

– короткий указатель адреса (смещение) возвращается в регистре **BX**;

– длинный указатель адреса (база : смещение) возвращается в паре регистров **ES : BX**.

2.4 Вычисление полиномов

Для вычисления полиномов n–й степени вида

$$Y = a_1 X^n + a_2 X^{n-1} + \ldots + a_n X + a_{n+1}$$

удобно использовать формулу Горнера

$$Y = \left(\ldots\left(\left(a_1 X + a_2\right)X + a_3\right)X + \ldots + a_n\right)X + a_{n+1}.$$

Если выражение, стоящее внутри скобок, обозначить через Y_i, тогда значение выражения в следующих скобках Y_{i+1} можно вычислить, используя рекуррентную формулу $Y_{i+1} = Y_i X + a_{i+1}$. Значение полинома ***Y*** получается после повторения этого процесса в цикле ***n*** раз. Начальное значение Y_1 должно быть равно a_1, а цикл следует начинать с ***i*=2**.

Приведенный на рисунке 6.1 алгоритм, позволяет вычислить значение полинома ***n***-й степени, значение которого находится в диапазоне **–32768 … +32767**. Вычисления сопровождаются выводом на экран сообщений, характеризующих результат:

– если *Y≥ 0* – "**Result plus**";

– если *Y< 0* – "**Result minus**";

– если *–32768 > Y > 32767* – "**Overflow**".

Исходными данными являются:

– коэффициенты полинома a_1, a_2, … a_5, которые хранятся в памяти в виде одномерного массива **MAS_A**. Размер каждого элемента массива –**WORD**;

– ***N*** – переменная для хранения порядка полинома;

– ***X*** – переменная для хранения аргумента X полинома;

– ***MES_1*** – строка символов '**Overflow**';

10.

– ***MES_2*** – строка символов '**Result minus**';
– ***MES_3*** – строка символов '**Result plus**';
– ***Y*** – двухбайтовая переменная для хранения результата вычисления полинома.

Регистры 16-ти разрядного микропроцессора распределены следующим образом:

– **регистр SI** используется для хранения индексов элементов массива **MAS_A**;
– **регистр CX** используется как счетчик циклов, который нужно повторить ***N*** раз;
– **регистры DX** и **AX** используются для формирования результата вычисления полинома: в **DX** формируется старшее слово результата (оно не используется), в **AX** – младшее слово результата.

В соответствии с **формулой Горнера**, основными операциями при вычислении полинома являются умножение и сложение, которые выполняется в цикле. Вычисление произведения $a_i X$ производится по команде **IMUL X** (блок 3). Результат умножения – двоичное число со знаком помещается в пару регистров: в **DX** – старшая часть, в **AX** – младшая часть. Если в старшей половине (в регистре **DX**) находятся значащие цифры, значит результат выходит за диапазон –32768…+32767, а флаги **CF** и **OF** устанавливаются в 1. **Эта ситуация рассматривается как переполнение**. Проверка на переполнение производится в блоке 4, и если переполнение есть (**OF** = 1), то на экран выводится сообщение '**Overflow**' (блок 14) и работа программы завершается.

Если переполнения нет, то к полученному в регистре **AX** произведению прибавляется значение $\mathbf{a_{i-1}}$ (блок 6). Для этого содержимое регистра указателя индекса **SI** требуется увеличить на 2 (массив **MAS_A** объявлен как **WORD**) (блок 5). В блоке 7, получившаяся сумма вновь проверяется на переполнение с помощью флага **OF**. Если **OF** = 1 (переполнение), то на экран выводится сообщение '**Overflow**' и работа программы завершается. Если переполнения нет, то вычисления продолжаются. В блоках 8 и 9 содержимое **счетчика циклов CX** уменьшается на 1 и если содержимое **CX** не равно 0, то происходит переход к началу цикла. После четырех проходов тела цикла содержимое счетчика **CX** = 0, происходит выход из цикла и результат, сформированный в регистре **AX**, пересылается в переменную **Y** (блок 10).

В блоке 11 проверяется знак результата и если он отрицательный, то на экран выводится сообщение **MES_2** '**Result minus**' (блок 12). В противном случае выводится **MES_3** '**Result plus**' (блок 15). В блоке 13 осуществляется стандартный выход в DOS.

11.

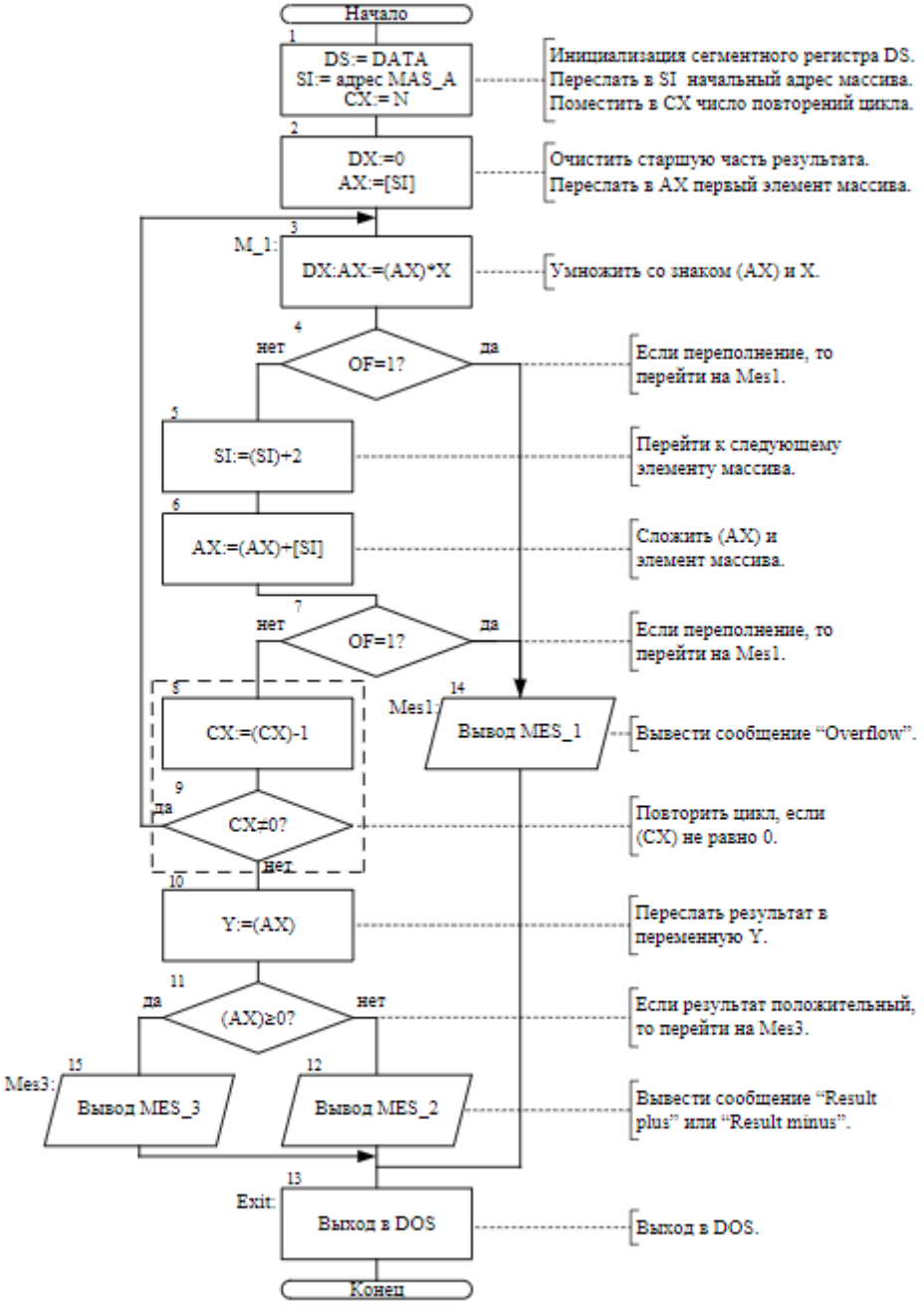


Рис.6.1 – Алгоритм вычисления полинома

2.5 Преобразование двоичных чисел в ASCII коды

Десятичные числа, которые используются в программах, хранятся в памяти или в регистрах микропроцессора либо в **двоичном виде** (двоичнокодированные беззнаковые и знаковые числа), либо в виде **двоично-десятичных чисел** (BCD – Binary-Coded Decimal) упакованного или неупакованного формата. Однако для того, чтобы вывести десятичное число на экран дисплея, необходимо каждую его цифру представить в виде символа. Для кодировки символов клавиатуры (в том числе и цифр) в Ассемблере используются **коды ASCII**, значения которых приведены в таблице 7.3 лабораторной работы №7.

Процесс преобразования **двоичнокодированных чисел в коды ASCII** заключается в последовательном целочисленном делении двоичного числа на 10d и выделении остатков деления до тех пор, пока частное от деления не окажется равным 0. Все получившиеся остатки образуют двоичные коды десятичных цифр, начиная с младшего разряда. Затем эти коды преобразуются в формат ASCII вписыванием числа **3d = 0110b** в старшую тетраду каждого байта.

Алгоритм, реализующий описанный процесс, приведен на рисунках 6.2 и 6.3. При этом следует учитывать:

– исходное преобразуемое число храниться в переменной **Number**;

– информация о знаке числа храниться в переменной **Sign**;

– преобразованное число в символьном виде храниться в переменной **Y_ASCII**;

– регистр **CX** – счетчик циклов;

– регистр **SI** – указатель адреса символов в переменной **Y_ASCII**.

Функционально алгоритм можно разделить на три участка. В блоках 1…5 выполняются подготовительные операции и определяется знак исходного числа, которая заносится в переменную **Sign**. Для этого исходное число **Number** помещается в регистр **AX** (бл.2) и по умолчанию считается положительным, т.е. в переменную **Sign** заносится **символ пробела** (знак плюс опускаем) (бл.3). Проверка знака осуществляется по флагу **SF**, который устанавливается после выполнения сравнения (AX) с 0 (бл.4). Если **SF = 0**, то число считается положительным и начинается его преобразование. В противном случае в переменную **Sign** записывается **символ минус** (‘–‘), и исходное число преобразуется из дополнительного в прямой двоичный код (бл.6).

12.

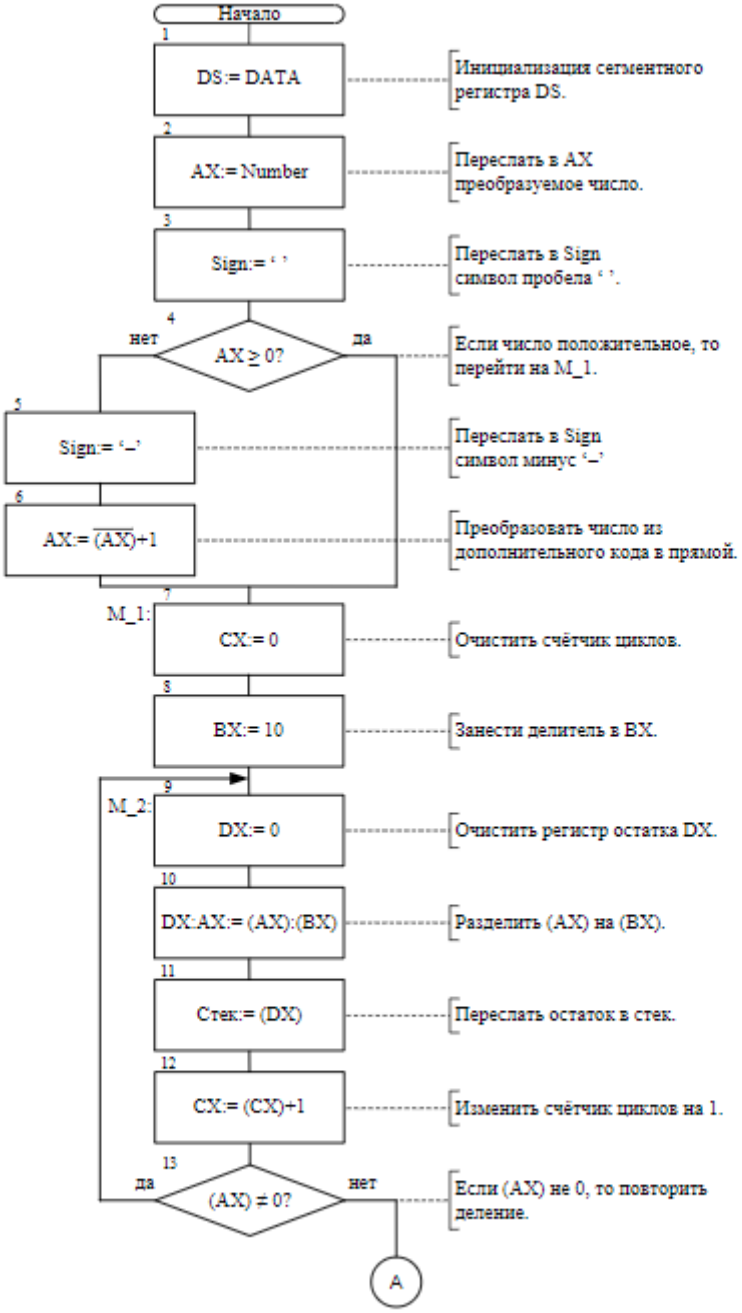


Рис. 6.2 – Алгоритм преобразования двоичного числа в кодыASCII

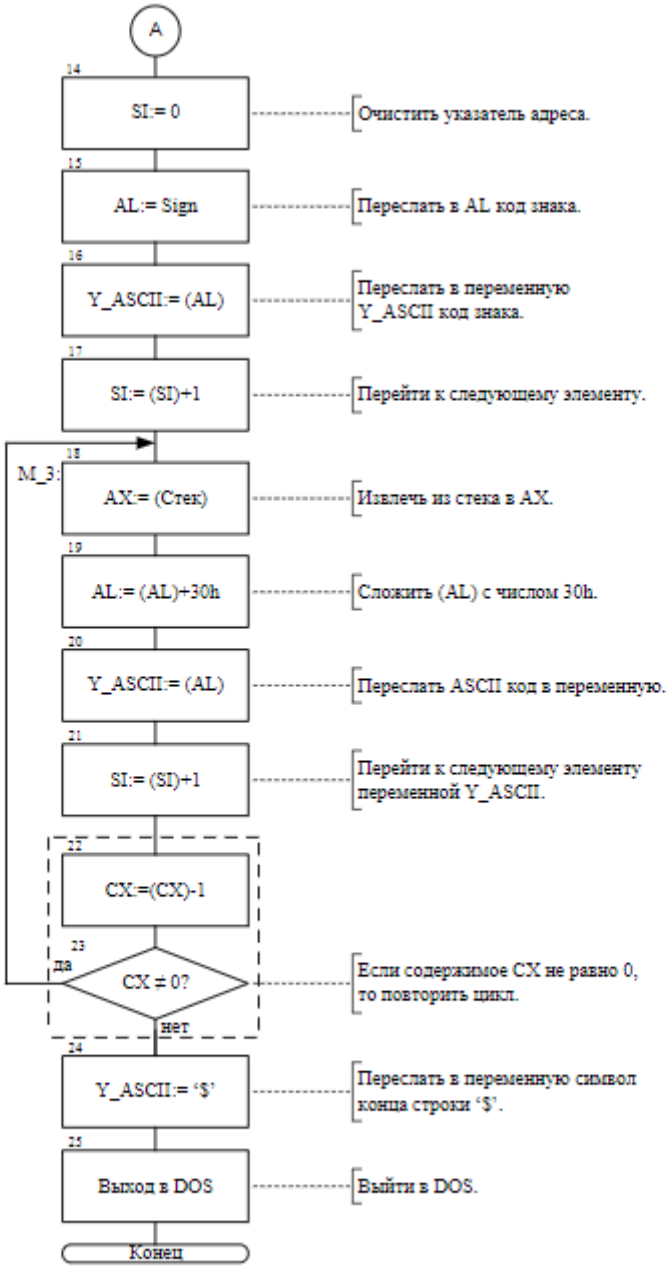


Рис. 6.3 – Продолжение алгоритма преобразования двоичного числа в коды ASCII

В блоках с 7 по 13 выполняется последовательное деление исходного двоичнокодированного числа на **10d** и сохранение остатков деления в **стеке**. Как известно, деление выполняется по команде **DIV src**, где **src** является делителем. Делитель помещается в регистр **BX** (бл.8). **Поскольку размер делимого должен быть в два раза больше размера делителя (регистра BX), оно должно размещаться в паре регистров DX:AX.** Делимым является исходная переменная **Number**, которая имеет размер **WORD** и размещается в аккумуляторе **AX** (бл.2 или бл.6). Это значит, что старшее слово делимого следует установить нулевым, т.е. регистр **DX** – очистить (бл.9). После выполнения деления в бл.10 **в регистре DX образуется целочисленный остаток, а в регистре AX – целое частное**. Остаток сохраняется в стеке (бл.11). В регистре **CX**, который является счетчиком циклов, подсчитывается число последовательных делений (бл.12). Деление продолжается до тех пор, пока частное не станет равным 0 (бл.13).

На третьем этапе алгоритма формируется переменная **Y_ASCII**, которая содержит символы знака и цифр числа. При этом регистр **SI** используется в качестве указателя адреса элементов символьной переменной. В блоках 14…17 в переменную **Y_ASCII** пересылается знак числа. **Сохраненные в стеке остатки деления** (двоичные коды цифр числа) **извлекаются из стека** (бл.18), **преобразуются в ASCII коды** (бл.19) **и пересылаются в символьную переменную Y_ASCII** (бл.20). Эти операции выполняются в цикле до тех пор, пока содержимое счетчика циклов **CX** не станет равным 0 (бл.22, 23). После выхода из цикла, преобразование завершается **пересылкой в переменную Y_ASCII символа конца строки '$'** (бл.24). Программа завершается выходом в DOS (бл.25).

3 Подготовка к работе

3.1. Изучить методические указания и рекомендованную литературу.

3.2. Подготовить ответы на контрольные вопросы.

4 Задание на выполнение работы

4.1 Используя, описанный в разделе 2, алгоритм вычисления полинома разобрать исходный текст программы **POLINOM** (см. ниже). Создать и отладить исполняемый модуль программы **POLINOM**, выполнив этапы ассемблирования и компоновки. Добавить в исходный модуль программы недостающие комментарии.

4.2 Упростить программу, реализовав трижды используемый вывод сообщений на экран в виде **подпрограмм**.

4.3 Отредактировать исходный модуль программы **POLINOM** для своего варианта задания (таблица 6.1). Создать и отладить исполняемый модуль программы **POLY_X** (X – номер варианта), выполнив этапы ассемблирования и компоновки. В случае появления переполнения выяснить причину и устранить ее уменьшив коэффициент a_1.

```
TITLE POLINOM
;Программа вычисления полинома вида Y=a1X^n + a2X^(n-1) + …+ anX +a^(n+1)
;Входные параметры:
;  коэффициенты полинома ai в массиве MAS_A
;  порядок полинома N
;  аргумент полинома X
;  Выходные параметры:
;  вычисленное значение полинома Y
;  сообщения MES_1, MES_2, MES_3
.MODEL SMALL                          ;Модель памяти ближнего типа.
.STACK 100h                           ;Отвести под стек 256 байт.
.DATA                                 ;Открыть сегментданных.
Mas_A  DW  -3, 3, -6, 9, -20          ;Коэффициенты полинома.
N DW 4                                ;Порядок полинома равен 4.
X DW 10                               ;Аргумент полинома равен 10.
Y  DW (?)                             ;Результат вычисления полинома.
Mes_1 DB 'OVERFLOW', 13, 10, '$'      ;Сообщение MES_1.
Mes_2 DB 'RESULT MINUS', 13, 10, '$'   ;Сообщение MES_2.
Mes_3 DB 'RESULT PLUS', 13, 10, '$'   ;Сообщение MES_3.
;-------------------------------------------------------------------------------
.CODE                                 ;Открыть сегмент кодов.
 Start: mov AX, @Data
        mov DS, AX
        lea SI, Mas_A
        mov CX, N
        xor DX, DX
        mov AX,[SI]
M_1:  imul X
        jo Mes1
        inc SI
        inc SI
        add AX, [SI]
        jo Mes1
        loop M_1
        mov Y, AX
```

```
        cmp AX, 0
        jge  Mes3
        mov DX, offset Mes_2            ;Вывод сообщения
        mov AH, 09                      ;на
        int 21h                         ;экран.
Exit :   mov AL, 0
        mov AH, 4Ch
        int 21h
Mes1: mov DX, offset Mes_1              ;Вывод сообщения
        mov AH,09                       ;на
        int 21h                         ;экран.
        jmp Exit
Mes3: mov DX, offset Mes_3              ;Вывод сообщения
        mov AH, 09                      ;на
        int 21h                         ;экран.
        jmp Exit
END  Start
```

Таблица 6.1

Исходные данные

N = 4, X = 10											
№ варианта	a_1	a_2	a_3	a_4	a_5	№ варианта	a_1	a_2	a_3	a_4	a_5
1	-4	9	0	-5	0	9	5	-9	-9	9	4
2	0	-9	8	6	-9	10	3	6	6	7	6
3	2	0	9	6	6	11	-2	8	7	4	9
4	3	2	-9	2	8	12	0	-7	-9	3	-3
5	-3	9	5	1	2	13	-1	6	-4	-9	8
6	-2	8	-5	0	1	14	2	4	3	8	0
7	-3	7	6	-9	9	15	4	0	6	-8	-1
8	3	-7	6	9	-8						