Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж

УТВЕРЖДАЮ
Директор
_____ С.А. Махновский
«23» марта 2017 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ ОП.08 ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

программы подготовки специалистов среднего звена
по специальности СПО
15.02.07 Автоматизация технологических процессов и производств
(по отраслям)
базовой подготовки

Магнитогорск, 2017

ОДОБРЕНО

Предметно-цикловой комиссией
Автоматизации технологических процессов
Председатель: Е.В. Менщикова
Протокол №7 от 14 марта 2017 г.

Методической комиссией

Протокол №4 от 23 марта 2017 г.

Разработчик:
преподаватель ФГБОУ ВО МГТУ им. Г.И. Носова М.Н. Корчагина

Методические указания разработаны на основе рабочей программы учебной дисциплины «Вычислительная техника».

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки студентов составляют практические занятия.

Состав и содержание практических работ направлены на реализацию действующего федерального государственного образовательного стандарта среднего профессионального образования по специальности 15.02.07 Автоматизация технологических процессов и производств (по отраслям)

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений - умений выполнять определенные действия, необходимые в последующем в профессиональной деятельности по общепрофессиональным дисциплинам.

В соответствии с рабочей программой учебной дисциплины «Вычислительная техника» предусмотрено проведение практических работ.

В результате их выполнения, обучающийся должен:

уметь:

– использовать типовые средства вычислительной техники и программного обеспечения.

Содержание практических работ ориентировано на подготовку студентов к освоению профессиональных модулей основной профессиональной образовательной программы по специальности и овладению профессиональными компетенциями:

ПК 4.1. Проводить анализ систем автоматического управления с учетом специфики технологических процессов.

ПК 4.2. Выбирать приборы и средства автоматизации с учетом специфики технологических процессов.

ПК 4.3. Составлять схемы специализированных узлов, блоков, устройств и систем автоматического управления.

ПК 4.4. Рассчитывать параметры типовых схем и устройств.

ПК 4.5. Оценивать и обеспечивать эргономические характеристики схем и систем автоматизации.

А также формированию общих компетенций:

- ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
- ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.
- ОК 4. Осуществлять поиск и использование информации, необ-

ходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

- ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.
- ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.
- ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.
- ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.
- ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

Выполнение студентами практических работ по учебной дисциплине «Вычислительна техника» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;

- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;

- формирование и развитие умений: наблюдать, сравнивать, сопоставлять, анализировать, делать выводы и обобщения, оформлять результаты в виде таблиц, схем, графиков;

- развитие аналитических интеллектуальных умений у будущих специалистов;

- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Продолжительность выполнения практической работы составляет не менее двух академических часов и проводится после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 1.2. Виды информации и способы представления ее в ЭВМ
Практическая работа № 1,2
Перевод чисел из одной системы счисления в другую

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Научиться переводить числа из одной системы счисления в другую.

Выполнив работу, Вы будете:

уметь:

Переводить целые и дробные числа из одной системы счисления в другую.

Материальное обеспечение: Методические указания

Задание:

Осуществить перевод чисел из одной системы счисления в другую.

Краткие теоретические сведения:

Перевод целых чисел из одной системы счисления в другую

1. Последовательно выполнять деление данного числа и получаемых целых частных на основание новой системы счисления до тех пор, пока не получим частное, меньшее делителя.

2. Полученные остатки, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления.

3. Составить число в новой системе счисления, записывая его, начиная с последнего остатка.

Перевод дробных чисел из одной системы счисления в другую

1. Последовательно умножать данное число и получаемые дробные части произведений на основание новой системы до тех пор, пока дробная часть произведения не станет равной нулю или будет достигнута требуемая точность представления числа.

2. Полученные целые части произведений, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления.

3. Составить дробную часть числа в новой системе счисления, начиная с целой части первого произведения.

Пример. Перевести число $0{,}65625_{10}$ в восьмеричную систему счисления.

0,	65625
	* 8
5	25000
	* 8
2	00000

Получаем: $0{,}65625_{10}=0{,}528$

Перевод произвольных чисел

Перевод произвольных чисел, т.е. чисел, содержащих целую и дробную части, осуществляется в два этапа. Отдельно переводится целая часть, отдельно — дробная. В итоговой записи полученного числа целая часть отделяется от дробной запятой (точкой).

Пример. Перевести число 17,2510 в двоичную систему счисления.

Переводим целую часть:	Переводим дробную часть:
17 2 1 8 2 0 4 2 0 2 2 0 1	0, 25 ×2 0 50 ×2 1 00

Получаем: $17{,}25_{10}=1001{,}012$

Перевод чисел из системы счисления с основанием 2 в систему счисления с основанием 2n и обратно

Перевод целых чисел. Если основание q-ичной системы счисления является степенью числа 2, то перевод чисел из q-ичной системы счисления в 2-ичную и обратно можно проводить по более простым правилам. Для того, чтобы целое двоичное число записать в системе счисления с основанием q=2n, нужно:

1. Двоичное число разбить справа налево на группы по n цифр в каждой.

2. Если в последней левой группе окажется меньше n разрядов, то ее надо дополнить слева нулями до нужного числа разрядов.

3. Рассмотреть каждую группу как n-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием q=2n.

Пример. Число 101100001000110010_2 переведем в восьмеричную систему счисления.

Разбиваем число справа налево на триады и под каждой из них записываем соответствующую восьмеричную цифру:

101	100	001	000	110	010
5	4	1	0	6	2

Получаем восьмеричное представление исходного числа: 541062_8.

Пример. Число 1000000000111110000111_2 переведем в шестнадцатеричную систему счисления.

Разбиваем число справа налево на тетрады и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0010	0000	0000	1111	1000	0111
4	0	0	F	8	7

Получаем шестнадцатеричное представление исходного числа: $400F87_{16}$.

Перевод дробных чисел. Для того, чтобы дробное двоичное число записать в системе счисления с основанием q=2n, нужно:

1. Двоичное число разбить слева направо на группы по n цифр в каждой.
2. Если в последней правой группе окажется меньше n разрядов, то ее надо дополнить справа нулями до нужного числа разрядов.
3. Рассмотреть каждую группу как n-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием q=2n.

Пример. Число $0{,}10110001_2$ переведем в восьмеричную систему счисления.

Разбиваем число слева направо на триады и под каждой из них записываем соответствующую восьмеричную цифру:

000,	101	100	010
0,	5	4	2

Получаем восьмеричное представление исходного числа: $0{,}542_8$.

Пример. Число $0{,}100000000011_2$ переведем в шестнадцатеричную систему счисления. Разбиваем число слева направо на тетрады и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0,	1000	0000	0011
0,	8	0	3

Получаем шестнадцатеричное представление исходного числа: $0{,}803_{16}$

Перевод произвольных чисел. Для того, чтобы произвольное двоичное число записать в системе счисления с основанием q=2n, нужно:

1. Целую часть данного двоичного числа разбить справа налево, а дробную — слева направо на группы по n цифр в каждой.
2. Если в последних левой и/или правой группах окажется меньше n разрядов, то их надо дополнить слева и/или справа нулями до нужного числа разрядов;
3. Рассмотреть каждую группу как n-разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием q=2n

Пример. Число $111100101{,}0111_2$ переведем в восьмеричную систему счисления.

Разбиваем целую и дробную части числа на триады и под каждой из них записываем соответствующую восьмеричную цифру:

111	100	101,	011	100
7	4	5,	3	4

Получаем восьмеричное представление исходного числа: $745{,}34_8$.

Пример. Число $11101001000{,}11010010_2$ переведем в шестнадцатеричную систему счисления.

Разбиваем целую и дробную части числа на тетрады и под каждой из них записываем соответствующую шестнадцатеричную цифру:

0111	0100	1000,	1101	0010
7	4	8,	D	2

Получаем шестнадцатеричное представление исходного числа: $748{,}D2_{16}$.

Перевод чисел из систем счисления с основанием q=2n в двоичную систему. Для того, чтобы произвольное число, записанное в системе счисления с основанием q=2n, перевести в двоичную систему счисления, нужно каждую цифру этого числа заменить ее n-значным эквивалентом в двоичной системе счисления.

Пример. Переведем шестнадцатеричное число $4AC35_{16}$ в двоичную систему счисления.

В соответствии с алгоритмом:

4	A	C	3	5
0100	1010	1100	0011	0101

Получаем: 1001010110000110101_2.

Порядок выполнения работы:

Ход работы:

1.Переведите в двоичную систему десятичные числа:

а)123 в) 99 д)1024

б)45 г)456 е)4095.

2. Переведите целые числа из десятичной системы счисления в двоичную:

а)513; в)600; д)602; ж)1000;

б)2304; г)5001; е)7000; з)8192.

3. Переведите десятичные дроби в двоичную систему счисления (ответ записать с шестью двоичными знаками):

а)0,4622; в)0,5198; д)0,5803; ж)0,6124;

б)0,7351; г)0,7982; е)0,8544; з)0,9321.

4. Переведите смешанные десятичные числа в двоичную систему счисления:

а)40,5; б)31,75; в)124,25; г)125,125.

5. Переведите целые числа из десятичной в восьмеричную систему счисления:

а) 8700; б)8888; в)8900; г)9300.

6. Переведите целые числа из десятичной в шестнадцатеричную систему счисления:

а)266; б)1023; в)1280; г)2041.

7. Переведите числа из десятичной системы счисления в восьмеричную:

а) 0,43; б) 37,41; в) 2936; г)481,625.

8. Переведите числа из десятичной системы счисления в шестнадцатеричную:

а) 0,17; б)43,78; в)25,25; г)18,5.

9. Переведите двоичные числа в восьмеричную систему счисления:

а)1010001001011; в)1011001101111; д)110001000100;

б)1010,00100101; г)1110,01010001; е)1000,1111001.

10. Переведите двоичные числа в шестнадцатеричную систему счисления: а)1010001001011; в)1011001101111; д)110001000100;

б)1010,00100101; г)1110,01010001; е)100,1111001.

11. Переведите восьмеричные и шестнадцатеричные числа в двоичную систему счисления:

а)2668; в)12708; д)10,238;

б)26616; г)2a1916; е)10,2316.

12. Осуществите перевод чисел по схеме А10» А16 » А2 » А8:

а) 16547;

б) 21589;

в) 8512;
г) 7756;
д) 5043;
е) 2323.
13. Перевести числа из восьмеричной системы счисления в шестнадцатеричную:
а) 12754;
б) 1515;
в) 7403.
14. Перевести числа из шестнадцатеричной системы счисления в восьмеричную:
а) 1AE2;
б) 1C1C;
в) 34E
г) 45AB
д) 123CC
е) 1010DD
15. Записать числа в восьмеричной системе счисления:
а) 10111010;
б) 11001111000111;
в) $A18C_{16}$;
г) $1375BE_{16}$.
Д) 1789_{10}
е) 2568_{10}
16. Записать числа в шестнадцатеричной системе счисления:
а) 10111010;
б) 11001111000111;
в) 77731_8;
г) 101154_8.
Д) 121514_{10}
е) 2356_{10}
17. Сравните числа:
а) 125_{16} и 111100010101;
б) 757_8 и 1110010101;
в) $A23_{16}$ и 1232_8;
г) $12{,}25_{16}$ и 111,100010101;
д) $63{,}5751_8$ и 11100,10101;
е) $B{,}A_{16}$ и $11{,}3_8$.

Тема 1.2. Виды информации и способы представления ее в ЭВМ

Практическая работа № 3,4

Недесятичная арифметика

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Научиться решать примеры в недесятичной арифметике.

Выполнив работу, Вы будете:

уметь:

Решать примеры в недесятичной арифметике

Материальное обеспечение: Методические указания

Задание:

Решить примеры в недесятичной арифметике

Краткие теоретические сведения:

Арифметические операции в двоичной системе счисления

+	0	1
0	0	1
1	1	10

-	0	1
0	0	11
1	1	0

*	0	1
0	0	0
1	0	1

```
+ 1001
  1010
-------
 10011
```

-110
11

11

***110**
11

+ 110
110

10010

-110	**11**
110	**10**
0	

Выполнить сложение, вычитание, умножение и деление в двоичной системе счисления:

1) 101011+1111=
2) 101111+11=
3) 10110100+11111=
4) 111111+1111=
5) 101010+111111=
6) 111001+111=
7) 101011111+11=
8) 11111+11=
9) 101011111+111=
10) 11111111+1=
11) 1010-11=
12) 1000-11=
13) 11100-11=
14) 100001000-10110011=
15) 11110011-1001011=
16) 110101110-10111111=
17) 10111*11=
18) 10101*101=
19) 101011*1111=
20) 101011*101=
21) 100111*111001=
22) 100101*111011=
23) 111100*100100=
24) 111110*100010=
25) 111010001001/111101=

26) 11111100101/101011=
27)100011011100/110110=
28) 111010001000/111100=

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

Таблица умножения для пятеричной системы:

*	1	2	3	4
1	1	2	3	4
2	2	4	11	13
3	3	11	14	22
4	4	13	22	31

1. **Составить таблицы сложения и умножения для троичной и четверичной системы**.
2. **Выполнить действия в троичной системе счисления:**

1) 12+22=
2) 11+22=
3) 12+11=
4) 221-11=
5) 212-11=
6) 21*2=
7) 121*11=
8) 111*222=

Таблица 5.1

Таблица сложения восьмеричных чисел

Первое слагаемое	Второе слагаемое							
	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Пример сложения двух восьмеричных чисел:

11 — единицы переноса
$(3447)_8$ — первое слагаемое
$(7045)_8$ — второе слагаемое

$(12514)_8$

Сложение начинают с разряда единиц: $(7)_8 + (5)_8 = (14)_8$. Записывают цифру 4 под чертой, а единицу переносят в следующий разряд — разряд восьмерок. Переходят к разряду восьмерок:

$$(1)_8 + (4)_8 + (4)_8 = (5)_8 + (4)_8 = (11)_8.$$

Одну единицу записывают, а другую переносят в следующий разряд. Переходя последовательно от разряда к разряду, определяют сумму $(12514)_8$.

Умножение двоичных и восьмеричных чисел производится аналогично умножению десятичных чисел. При этом пользуются соответствующими таблицами умножения чисел в двоичной (табл. 5.2) и восьмеричной системах счисления.

3. Выполнить сложение двух восьмеричных однозначных чисел:

1) 2356+4567=
2) 356+457=
3) 235+564=
4) 4576+1235=
5) 1452+2356=
6) 4756+7045=
7) 1456+7425=

8) 1564+123=
9) 1234+475=
10) 1237+456=

Вычитание двоичных чисел производится так же, как и десятичных, т. е. последовательно по разрядам от младшего к старшему. Если из меньшей цифры в данном разряде вычитается большая, то производится заем единицы из следующего старшего разряда, т. е. цифра этого старшего разряда становится на единицу меньше.

В вычислительной технике операции вычитания обычно заменяются операциями сложения. Рассмотрим пример такой замены. Вместо того чтобы из числа 85 вычитать число 37, к числу 85 прибавляется число 63 = 100 – 37 (дополнительное к 37) и от результата 148 отнимается единица в старшем разряде. Получается число 48, которое является искомой разностью.

Аналогичным образом можно и в двоичной системе заменить вычитание сложением с использованием *дополнительного кода*. Саму операцию вычитания можно представить как сложение с отрицательным числом.

В вычислительной технике при использовании двоичной системы счисления крайний левый разряд служит для записи знака числа. Для положительного числа в этот разряд записывается 0, а для отрицательного — 1. Записанные таким образом двоичные числа будем называть записанными в *прямом коде*.

Рассмотрим составление дополнительного кода к прямому коду отрицательного числа.

Дополнительный код отрицательных двоичных чисел формируется по следующему правилу. Сначала цифры всех разрядов кроме знакового инвертируют (вместо 0 записывают 1, а вместо 1 — 0) и в младший разряд добавляют еди-

Таблица 5.2

Таблица умножения двоичных чисел

Сомножители	0	1
0	0	0
1	0	1

ницу. Если в младшем разряде уже стоит единица, то при этом приходится изменять цифру в следующем, а, возможно, и в более старших разрядах.

Например, при вычитании из числа 10110 числа 01101 уменьшаемое представляют как положительное число в прямом коде **0**10110, а вычитаемое — как отрицательное число, прямой код которого **1**01101 (полужирным шрифтом выделены цифры знакового разряда). Определяют дополнительный код вычитаемого. Сначала инвертируют цифры всех разрядов, кроме знакового (результат **1**10010), затем прибавляют единицу в младший разряд (**1**10011). Выполняют операцию сложения уменьшаемого (в прямом коде) с вычитаемым (в дополнительном коде):

$$\begin{array}{r} \mathbf{0}10110 \\ + \\ \underline{\mathbf{1}10011} \\ \mathbf{0}01001. \end{array}$$

Число 01001 и есть результат вычитания, полученный в прямом коде. При сложении цифры знаковых разрядов складывают с отбрасыванием возникающего из этого разряда переноса. В данном примере в результате вычитания получилось положительное число, поскольку в знаковом разряде стоит **0**. Это естественно, так как уменьшаемое больше вычитаемого. Если же из меньшего числа вычитать большее, то получается отрицательное число.

Убедимся в этом на примере, из числа 01101 (в прямом коде **0**01101) вычтем 10110. Для этого определим дополнительный код отрицательного числа **1**10110: сначала инвертируем цифры всех разрядов, кроме знакового (**1**01001), потом добавим единицу в младший разряд (**1**01010). Выполним сложение уменьшаемого в прямом коде и вычитаемого в дополнительном коде:

$$\begin{array}{r} \mathbf{0}01101 \\ + \\ \underline{\mathbf{1}01010} \\ \mathbf{1}10111. \end{array}$$

Результат есть отрицательное число (**1** в знаковом разряде) и выражен он в дополнительном коде. Для получения его прямого кода убавим единицу в младшем разряде (**1**10110), после чего инвертируем цифры всех разрядов, кроме знакового (**1**01001).

Правильность вычислений проверим на десятичных числах: $(10110)_2 = (22)_{10}$; $(01101)_2 = (13)_{10}$; $(01001)_2 = (9)_{10}$; 22 – 13 = 9; 13 – – 22 = –9.

Выполнить вычитание двоичных чисел
А) 1010111-111101=

нии складываемых цифр получается 1, что соответствует знаку произведения двух сомножителей с разными знаками. Абсолютная величина произведения определяется перемножением чисел без учета их знаков. Перемножение многоразрядных двоичных чисел производится с помощью табл. 5.2.

При умножении двух двоичных чисел множимое (первый сомножитель) последовательно умножают на каждую цифру множителя (второго сомножителя), начиная либо с младшего, либо со старшего разряда, и для учета веса соответствующей цифры множителя сдвигают либо влево (при начале умножения с младшего разряда множителя), либо вправо (при начале со старшего разряда) на такое число разрядов, на какое соответствующий разряд множителя сдвинут относительно младшего или старшего разряда.

При умножении вручную на бумаге мы привыкли начинать с младшей цифры второго сомножителя. При этом результат умножения на цифру следующего разряда записываем левее предыдущего результата на один разряд, т.е. тем самым производим сдвиг влево. Результаты умножения первого сомножителя на каждую цифру второго сомножителя называют частичными произведениями или промежуточными суммами. Получающиеся в результате умножения и сдвига частичные произведения после суммирования дают полное произведение. Особенность умножения двоичных чисел состоит в том, что частичное произведение может быть либо сдвинутым на соответствующее число разрядов множимым, если соответствующая цифра множителя равна 1, либо нулем, если соответствующая цифра множителя равна 0.

При умножении двоичных многоразрядных чисел с учетом их знаков необходимо выполнить две операции: определить знак произведения и найти его абсолютную величину. Знаковый разряд может быть получен суммированием цифр знаковых разрядов сомножителей без формирования разряда переноса. При несовпаде-

Рассмотрим пример:

```
  10111   — множимое
  ×
   1101   — множитель
  10111   — первое частичное произведение
 00000    — второе частичное произведение
10111     — третье частичное произведение
10111     — четвертое частичное произведение
100101011 — произведение
```

Тот же результат можно получить при умножении, начиная со старших разрядов множителя:

```
 10111
 ×
  1101
 10111
  10111
   00000
    10111
100101011.
```

```
 132
-
 110 — первое вычитание
 022
 220 — сдвиг
-
 110 — первое вычитание
 110
-
 110 — второе вычитание
 000
```

Ответ: 12 (одно вычитание до сдвига и два после).

Замена вычитания сложением остатка с дополнительным кодом вычитаемого сводит операцию деления к последовательности трех простейших операций.

Выполнить деление с помощью операций вычитаний и сдвигов
А) 255:17=
Б) 228:19=
В) 3248:56=

Деление является весьма трудоемкой операцией. В ряде случаев в цифровых устройствах оно заменяется нахождением обратной величины делителя по специальной подпрограмме (на основе какой-либо быстро сходящейся итерационной формулы) и последующим умножением делимого на найденную обратную величину.

Иными словами, во многих машинах операция деления заменяется умножением, так как $a/b = a(1/b)$. По числу b машина автоматически вычисляет число $1/b$, которое затем умножается на a.

Довольно часто результат деления вычисляется не вполне точно, т.е. с некоторым приближением. Ведь деление без остатка не всегда возможно. В привычной нам десятичной системе это тоже часто бывает. Например, если разделить 2 на 3, то в ответе получится 0,666..., т.е. 6 в периоде. На практике принимают результат с округлением: 0,67, или 0,667, или 0,6667. Чем больше знаков после запятой, тем меньше ошибка вычисления.

Тема 1.2. Виды информации и способы представления ее в ЭВМ
Практическая работа № 5,6

Представление чисел в разрядной сетке ЭВМ

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Научиться представлять числа в разрядной сетке ЭВМ.

Выполнив работу, Вы будете:

уметь:

Представлять числа в прямом, обратном и дополнительном коде.

Материальное обеспечение: Методические указания

Задание:

Представить числа в разрядной сетке ЭВМ согласно заданию в инструкции.

Краткие теоретические сведения:

Прямой код любого двоичного числа представлен правильной двоичной дробью

Нуль и единица слева от запятой в записи кодов чисел означает положительное (плюс) и отрицательное (минус) число соответственно.

Обратный код

Для перевода правильной отрицательной двоичной дроби в обратный код необходимо в знаковом разряде поставить единицу, а все цифры разрядов числа инвертировать, т. е. единицы заменить нулями, а нули единицами.

Задания для самостоятельного выполнения

1. Заполнить таблицу, записав отрицательные десятичные числа в прямом, обратном и дополнительном кодах в 16-разрядном представлении:

Десятичные числа	Прямой код	Обратный код	Дополнительный код
-10			
-100			
-1000			
-10000			

2. Заполнить таблицу, записав десятичные числа в заданном компью-
терном представлении:

Десятичные числа	Компьютерное представление	
	целые неотрицательные числа	целые числа со знаком
255		
-255		
32768		
-32768		

3. Заполнить таблицу, записав максимальные и минимальные значения чисел в заданном компьютерном представлении:

Компьютерное представление	Максимальное значение	Минимальное значение
целые неотрицательные числа		
целые числа со знаком		
большое целое число со знаком		

4. Выполнить арифметическое действие 2010 - 6010 в 16-разрядном компьютерном представлении.

5. Записать следующие числа в форме с плавающей запятой и нормализованной мантиссой:

а) 217,93410; б) 7532110; в) 10,010110; г) 20045010.

6. Определить максимальное число и его точность для формата чисел двойной точности у если для хранения порядка и его знака отводится 11 разрядов, а для хранения мантиссы и ее знака 53 разряда.

Произвести сложение, вычитание, умножение и деление чисел 0,1-2 и ОД • 2" в формате с плавающей запятой.__

Сложение чисел. Сложение чисел с фиксированной запятой осуществляется в одном из машинных кодов -обратном или дополнительном. Причем, операции в этих кодах выполняются обычно над двоичными, числами, являющимися правильными дробями. Последовательность выполнения операции сложения следующая:

- исходные числа записываются в принятом для данной машины коде;
- производится поразрядное сложение кодов чисел, включая и знаковые разряды;
- единица переноса добавляется к младшему разряду суммы, если операция выполнялась над числами в обратном коде, или не учитывается (отбрасывается) в дополнительном коде;
- производится анализ на переполнение разрядной сетки. В случае переполнения вырабатывается сигнал на прерывание программы.

Переполнение разрядной сетки устраняется путем изменения масштабных коэффициентов.
Операция вычитания в ЭВМ заменяется операцией сложения в модифицированном обратном или дополнительном коде.
Рассмотрим сложение положительных и отрицательных чисел.
Пример. Сложить числа x=0,101001 и y=0,011011 в модифицированном коде

Решение
00,101001
+
00,011011
01,000100
Сочетание 01 в знаковых разрядах свидетельствует о переполнении. Полученная сумма есть число, по модулю превышающее единицу.

7. Сложить числа x=0,101111 и y=1,110010 в модифицированных обратном и дополнительном кодах. Результат сложения представить в прямом коде.
8. Сложить числа x= 1,1010 и y= 1,1101

Различные значения цифр (10) в знаковых разрядах сумм, модифицированных обратного и дополнительного кодов означают переполнение разрядной сетки машины.

Тема 1.3. Логические основы ЭВМ

Практическая работа № 7,8

Построение таблиц истинности

Формируемые компетенции:

- ПК 4.2. Выбирать приборы и средства автоматизации с учетом специфики технологических процессов.
- ПК 4.3. Составлять схемы специализированных узлов, блоков, устройств и систем автоматического управления.
- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Научиться строить таблицы истинности.

Выполнив работу, Вы будете:

уметь:

. Строить таблицы истинности

Материальное обеспечение: Методические указания

Задание:

Построить таблицы истинности

Краткие теоретические сведения:

Логическая операция КОНЪЮНКЦИЯ (логическое умножение):

• в естественном языке соответствует союзу и;

• в алгебре высказываний обозначение &;

• в языках программирования обозначение And.

Конъюнкция — это логическая операция, ставящая в соответствие каждым двум простым высказываниям составное высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

Таблица истинности

A	**B**	*A&B*
0	**1**	**0**
0	**0**	**0**
1	**1**	**1**
1	**0**	**0**

Логическая операция ДИЗЪЮНКЦИЯ (логическое сложение):

• в естественном языке соответствует союзу или;

• обозначение v ;

• в языках программирования обозначение Or.

90 Глава 3

Дизъюнкция — это логическая операция, которая каждым двум простым высказываниям ставит в соответствие составное высказывание, являющееся ложным тогда и только тогда, когда оба исходных высказывания ложны и истинным, когда хотя бы одно из двух образующих его высказываний истинно.

Таблица истинности

A	**B**	*AvB*
0	**1**	**1**

0	**0**	**0**
1	**1**	**1**
1	**0**	**1**

Логическая операция ИНВЕРСИЯ (отрицание):
• в естественном языке соответствует словам неверно, что... и частице не; $\overline{}$
• обозначение A;
• в языках программирования обозначение Not.
Отрицание — это логическая операция, которая каждому простому высказыванию ставит в соответствие составное высказывание, заключающееся в том, что исходное высказывание отрицается.

Таблица истинности

A	$\overline{A}$
0	**1**
1	**0**

Логическая операция ИМПЛИКАЦИЯ (логическое следование):

• в естественном языке соответствует обороту если ..., то ...;

• обозначение => .

Импликация — это логическая операция, ставящая в соответствие каждым двум простым высказываниям составное высказывание, являющееся ложным тогда и только тогда, когда условие (первое высказывание) истинно, а следствие (второе высказывание) ложно.

A	B	A =>Б
0	0	1
0	1	1
1	0	0
1	1	1

Логическая операция ЭКВИВАЛЕНЦИЯ (равнозначность):

• в естественном языке соответствует оборотам речи тогда и только тогда; в том и только в том случае;

• обозначения <=>, ~ .

Эквиваленция — это логическая операция, ставящая в соответствие каждым двум простым высказываниям составное высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания одновременно истинны или одновременно ложны.

Таблица истинности эквиваленции:

A	B	A⇔B
0	0	1
0	1	0
1	0	0
1	1	1

Логические операции имеют следующий приоритет: действия в скобках, инверсия, &, v, =>, о.

1. Найдите значения логических выражений:

а) (lvl)v(lvO);

6)((lv0)vl)vl;

в) (Ovl)v(lvO);

г) (0&1)&1;

д) 1&(1&1)&1;

е) ((lv0)&(l&l))&(0vl);

ж) ((l&0)v(l&0))vl;
з) ((l&l)v0)&(0vl);
и) ((0&0)v0)&(lvl).

2. Для формулы *A&(BvB&C)* построить таблицу истинности алгебраически и с использованием электронных таблиц.

Количество логических переменных 3, следовательно, количество строк в таблице истинности должно быть 2^3=8 Количество логических операций в формуле 5, следовательно количество столбцов в таблице истинности должно быть 3+5=8.

A	B	C	$\overline{B}$	$\overline{C}$	$\overline{B}$ & $\overline{C}$	B V($\overline{B}$ & $\overline{C}$)	A&(BV$\overline{B}$ & $\overline{C}$)
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

3. Построить таблицы истинности для следующих формул:

а)Av(Bv$\overline{B}$_=>$\overline{C}$);

б)A&(B&$\overline{B}$=>$\overline{C}$);

в) Av(Bv$\overline{B}$)&Av(B => C).

4. Выбрать составное высказывание, имеющее ту же таблицу истинности, что и не (не А и не(В и С)).

1) А и В или С и А; 3) А и (В или С);
2) (А или В) и (А или С); 4) А или (не В или не С).

Логические операции имеют следующий приоритет: действия в скобках, инверсия, конъюнкция, дизъюнкция, импликация и эвивалентность.

1.Пусть логическая функция задается следующей формулой: F(x,y,z)=$(x \sim y)\&z \to x$. Вычислить значение функции при наборе аргументов (1,0,1), т.е x=1, y=0, z=1.

2. Пусть логическая функция трех переменных задана формулой F(x,y,z)=$(x \cup y) \to x\&z$

x	y	Z	$\overline{x}$	$\overline{x} \cup y$	x&z	$(x \cup y) \to x\&z$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

3. Пусть заданы две функции F(x,y,z)=$\overline{(x \cup y)} \to x\&z$ и $G(x,y,z) = x\&(z \cup y)$. Докажите, что таблицы истинности этих функций совпадают.

Основные законы булевой алгебры.

1. Ассоциативность конъюнкции и дизъюнкции (сочетательный закон):
   x(yz)=(xy)z=xyz
   x∪(y∪z)=(x∪y) ∪z=x∪y∪z
2. Коммутативность конъюнкции и дизъюнкции (переместительный закон):
   xy=yx
   x∪y=y∪z
3. Дистрибутивность конъюнкции относительно дизъюнкции (распределительный закон)
   x(y∪z)=xy∪xz
4. Дистрибутивность дизъюнкции относительно конъюнкции (распределительный закон)
   xyz=(x∪y)(x∪z)
5. Закон идемпотентности (одинаковости):
   xx=x; x∪x=x

   Заполнить таблицу истинности для проверки законов булевой алгебры

x	y	z	y∪z	x(y∪z)	xy	xz	xy∪xz
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

6. Закон двойного отрицания:

$\overline{\overline{x}}=x$

7. Правило де Моргана, или закон отрицания (инверсии):

$\overline{xy}=\overline{x}\cup\overline{y}$

$\overline{x\cup y}=\overline{x}\,\overline{y}$

8. Свойства констант:

x&1=x

x∪1=1

x&0=0

x∪0=x

Дополнительные законы алгебры логики

1. Закон поглощения

x∪xy=x

x(x∪y)=x

2. Законы склеивания

$xy\cup\overline{x}y=y$

$(x\cup y)(x\cup\overline{y})=y$

3. Закон Блейка – Порецкого

$x\cup\overline{x}y=x\cup y$

4. Закон свертки:

$xy\cup\overline{x}z\cup yz=xy\cup\overline{xz}$

Задание: Построить таблицы истинности.

А) (А импликация В) эквивалентность А конъюнкция В

Б) А импликация (В импликация АВ)

В) (А импликация В) импликация ((А импликация (отрицание В)) импликация (отрицание А))
Г) (А импликация (С импликация В)) импликация (В дизъюнкция С)

Тема 1.3. Логические основы ЭВМ

Практическая работа № 9,10

Построение логических схем по таблице истинности

Формируемые компетенции:

- ПК 4.2. Выбирать приборы и средства автоматизации с учетом специфики технологических процессов.
- ПК 4.3. Составлять схемы специализированных узлов, блоков, устройств и систем автоматического управления.
- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Научиться строить логические схемы по таблицам истинности.

Выполнив работу, Вы будете:

уметь:

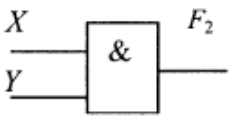
Строить логические схемы по таблицам истинности.

Материальное обеспечение: Методические указания

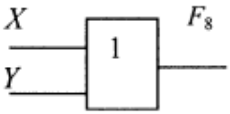
Задание:

Построить логические схемы по таблицам истинности

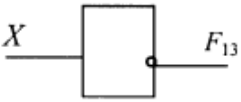
Краткие теоретические сведения:



Конъюнктор

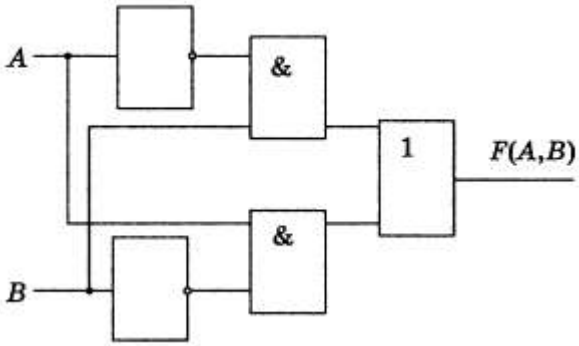


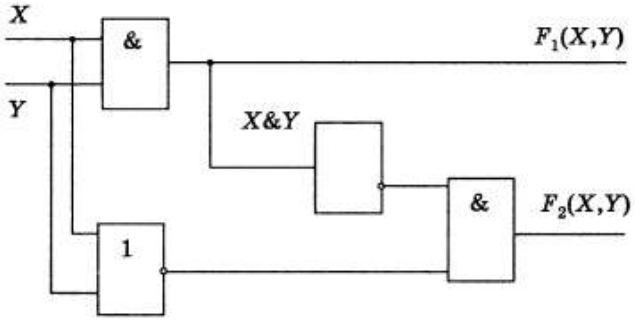
Дизъюнктор



Инвертор

1. По предложенным схемам построить таблицы истинности


A
B
&
&
1
F(A,B)



Логические функции двух переменных

Аргументы		Логические функции															
A	B	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

2.

Существуют 16 логических функций от двух переменных Постройте их логические схемы с помощью базовых логических элементов: конъюнктора, дизъюнктора и инвертора.

Тема 1.3. Логические основы ЭВМ

Практическая работа № 11

Комбинационные схемы

Формируемые компетенции:

- ПК 4.2. Выбирать приборы и средства автоматизации с учетом специфики технологических процессов.
- ПК 4.3. Составлять схемы специализированных узлов, блоков, устройств и систем автоматического управления.
- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Выполнив работу, Вы будете:

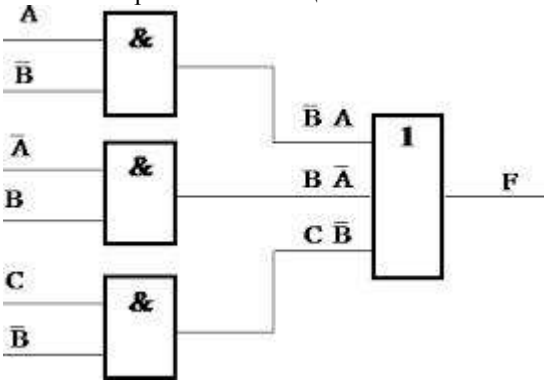
уметь:

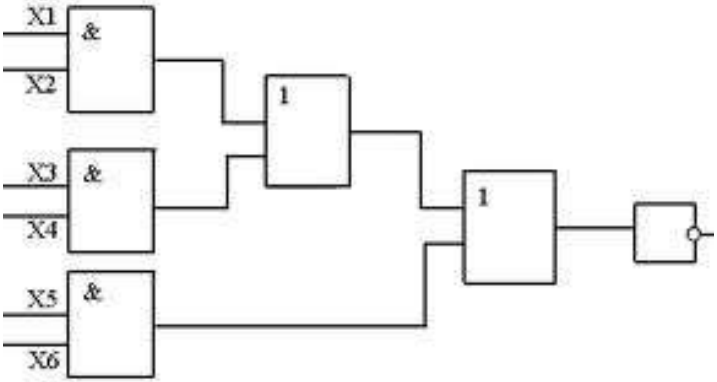
Строить комбинационные схемы в программе Multisim

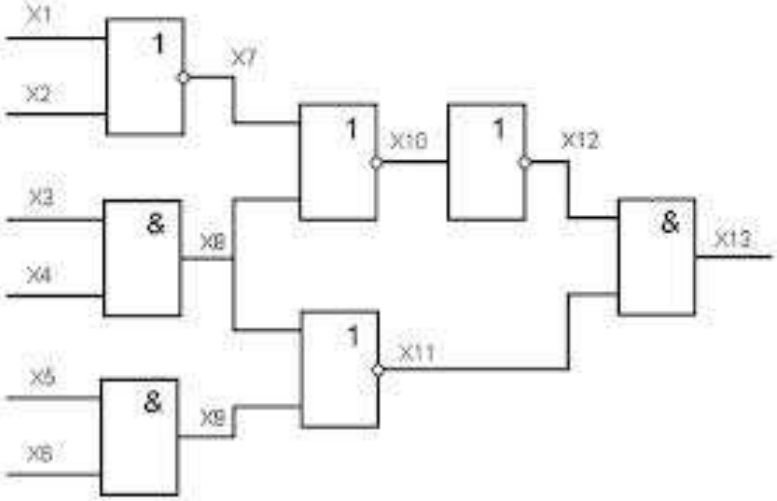
Материальное обеспечение: Методические указания

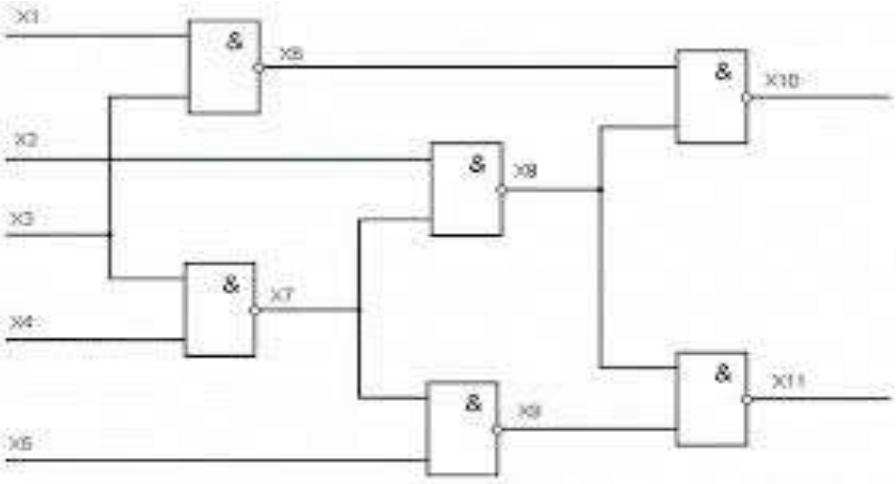
Задание:

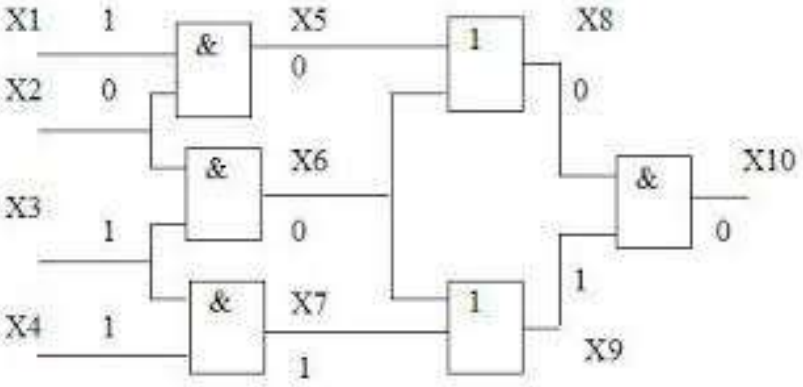
Построить комбинационные схемы и таблицы исинности к ним.

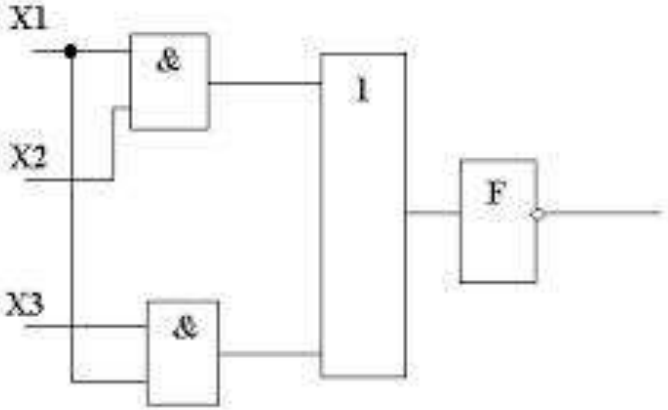


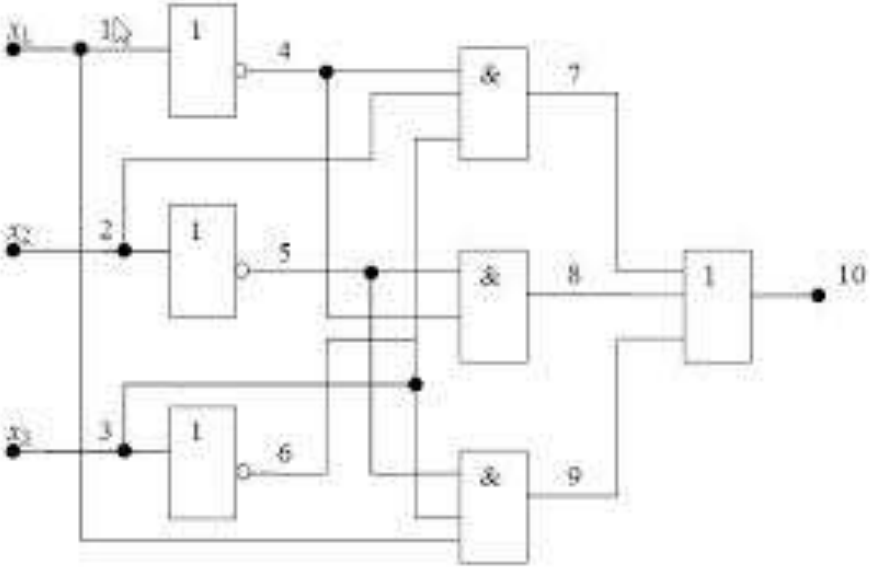

X1
X2
X3
X4
X5
X6
&
&
&
1
1


X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
1
1
1
1
&
&
&


X1
X2
X3
X4
X5
&
X6
X7
X8
X9
X10
X11


X1
1
X2
0
X3
1
X4
1
&
X5
0
X6
0
X7
1
1
X8
0
1
X9
&
X10
0


X1
X2
X3
&
&
1
F


1
2
3
1
1
1
4
5
6
&
&
&
7
8
9
1
10

Тема 2.1 Типовые комбинационные цифровые устройства

Практическая работа № 12,13

Работа шифратора

Работа дешифратора

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть работу дешифратора.

Материальное обеспечение: Методические указания

Дешифраторы

Комбинационной называется логическая схема, реализующая однозначное соответствие между значениями входных и выходных сигналов. Дешифратор – логическая комбинационная схема, имеющая n информационных входов и 2^n выходов. Каждой комбинации логических уровней на входах будет соответствовать активный уровень на одном из 2^n выходов. Как любая логическая схема, дешифратор может быть задан таблицей истинности. Таблица истинности дешифратора 3х8 (табл. 1) состоит из трех столбцов, соответствующих входным сигналам X_0, X_1, X_2, и восьми столбцов, соответствующих выходным сигналам Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7. В первых слева трех столбцах расположены возможные комбинации входных сигналов, а в последних восьми – соответствующие им комбинации выходных сигналов.

Таблица 1

X_2	X_1	X_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Схема имеет восемь выходов, на одном из которых потенциал высокий, на остальных низкий. Номер единственного выхода, имеющего высокий потенциал, соответствует двоичному числу, формируемому состояниями входных сигналов. Этот принцип формирования выходного сигнала можно описать следующим образом: $Y_i = 0$, если $i = k$; $Y_i = 1$, если $i \neq k$, здесь i – номер разряда; $k = 2^2 X_2 + 2^1 X_1 + 2^0 X_0$.

Выражения для каждого выхода дешифратора:

$$\overline{Y}_0 = \overline{X}_2 \overline{X}_1 \overline{X}_0, \qquad Y_4 = X_2 \overline{X}_1 \overline{X}_0,$$
$$\overline{Y}_1 = \overline{X}_2 \overline{X}_1 X_0, \qquad Y_5 = X_2 \overline{X}_1 X_0,$$
$$\overline{Y}_2 = \overline{X}_2 X_1 \overline{X}_0, \qquad Y_6 = \overline{X}_2 X_1 X_0,$$
$$Y_3 = X_2 X_1 \overline{X}_0, \qquad Y_7 = X_2 X_1 X_0,$$

где «¯» – инвертирование.

Таким образом, схема дешифратора должна содержать три схемы «НЕ» и восемь схем «3И» (рис. 1).

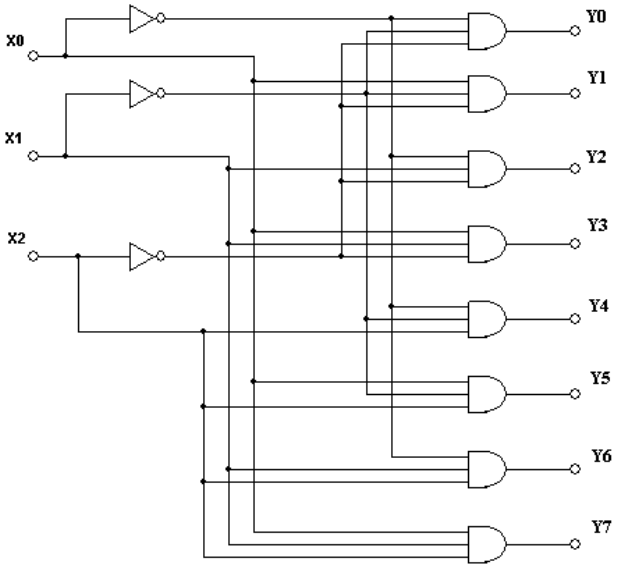


Рис. 1. Схема дешифратора 3x8

Исследование дешифратора

1. В программе multisim собрать схему дешифратора 3x8 (рис.2). Для этого поместить на схему восемь логических схем «3И», восемь све-

тодиодов, пять логических схем «НЕ», три переключателя на два направления и три вольтметра.

2. Присвоить переключателям управляющие клавиши.

3. С помощью переключателей подать на вход дешифратора все возможные комбинации сигналов и записать для каждого входного сигнала выходной сигнал (восьмиразрядную комбинацию), т.е. заполнить таблицу истинности.

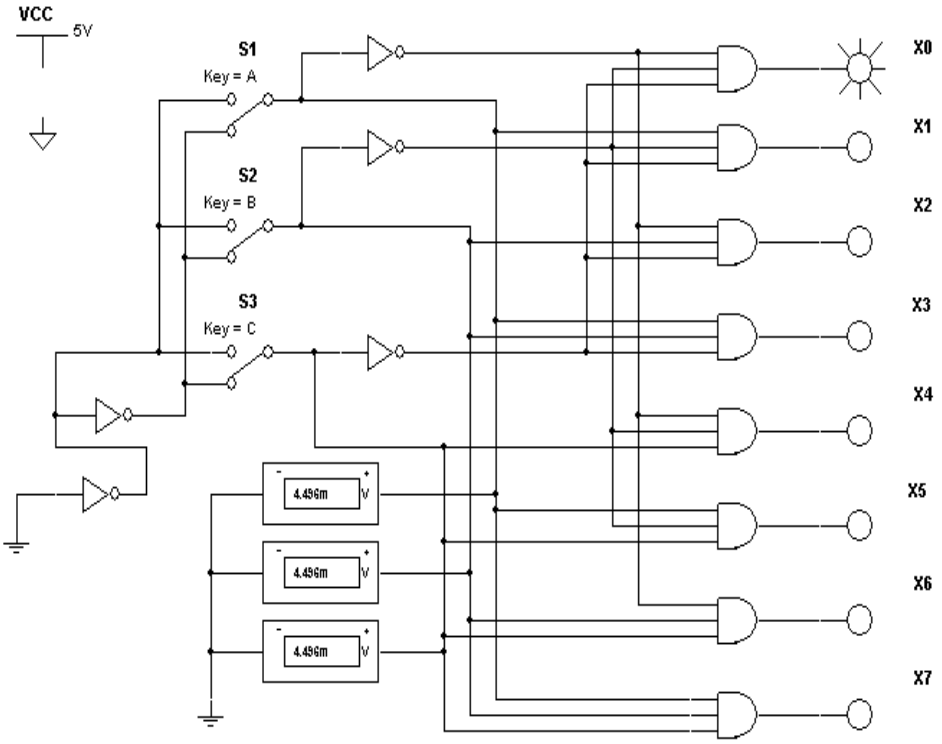


Рис. 2 . Дешифратор 3х8

Содержание отчета

Отчет должен включать:

1) название пункта работы,

2) исследуемую схему,

3) результат моделирования для дешифратора – таблица истинности.

Тема 2.1 Типовые комбинационные цифровые устройства

Практическая работа № 14

Работа мультиплексора

МУЛЬТИПЛЕКСОР

Мультиплексор (MS) – это функциональный узел, осуществляющий подключение (*коммутацию*) одного из нескольких входов к выходу *y*. На выход такого устройства передаётся логический уровень того информационного разряда, номер которого в двоичном коде задан на адресных входах x_1 и x_2. Условное изображение мультиплексора на четыре входа и возможный вариант его структурной схемы показаны на рис 1. *а* и *б*.

При $x_1 = 0$ и $x_2 = 0$, $y = a$; при $x_1 = 0$ и $x_2 = 1$, $y = b$; при $x_1 = 1$ и $x_2 = 0$, $y = c$ и при $x_1 = 1$ и $x_2 = 1$, $y = d$.

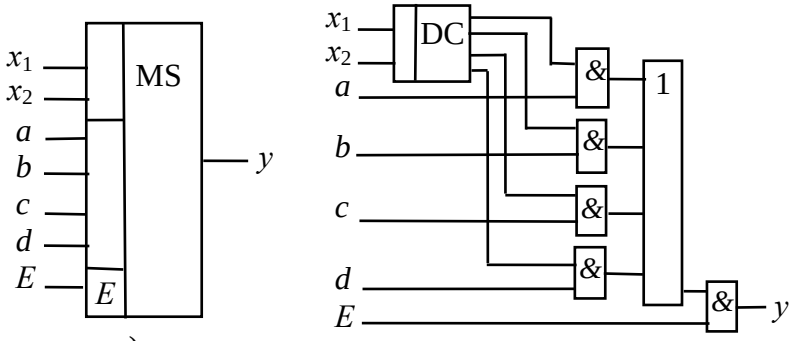


Рис. 1

Функционирование мультиплексора описывается выражением

$$y = a\bar{x}_1\bar{x}_2 + b\bar{x}_1x_2 + cx_1\bar{x}_2 + dx_1x_2.$$

Вход *E* – разрешающий: при $E = 1$ мультиплексор работает как обычно, при $E = 0$ выход узла находится в неактивном состоянии, мультиплексор заперт. Серийные узлы выпускаются с числом адресных входов $n = 2, 3$ и 4 при возможном числе 2^n коммутируемых входов. При необходимости коммутировать большее количество входов используют несколько мультиплексоров. Мультиплексоры находят широкое применение в устройствах отображения информации в различных устройствах управления.

Так как мультиплексор может пропустить на выход сигнал с любого информационного входа, адрес которого установлен на соответствующих адресных входах, то на основе мультиплексоров реализуют логические функции, подавая на информационные входы логические 1 или 0 в соответствии с таблицей переключений, а на адресные входы – аргументы функции.

Задание №1. На рабочем поле среды MULTISIM собрать схему (Рис.2) для исследования работы мультиплексора. Подать на адресные входы соответствующий код, для получения сигнала на выходе. Проанализировать осциллограмму на осциллографе.

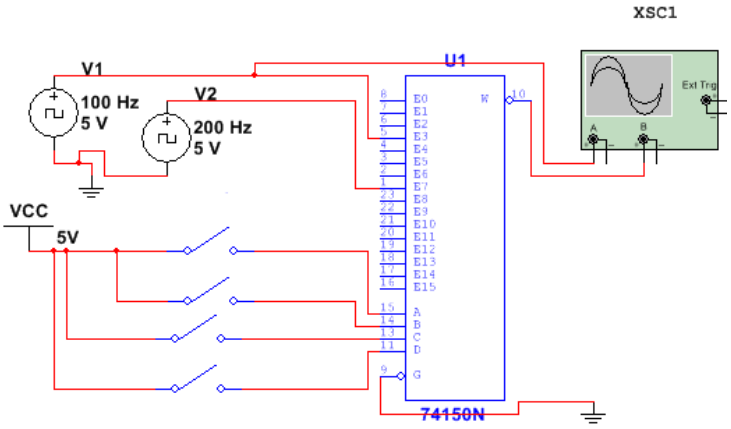
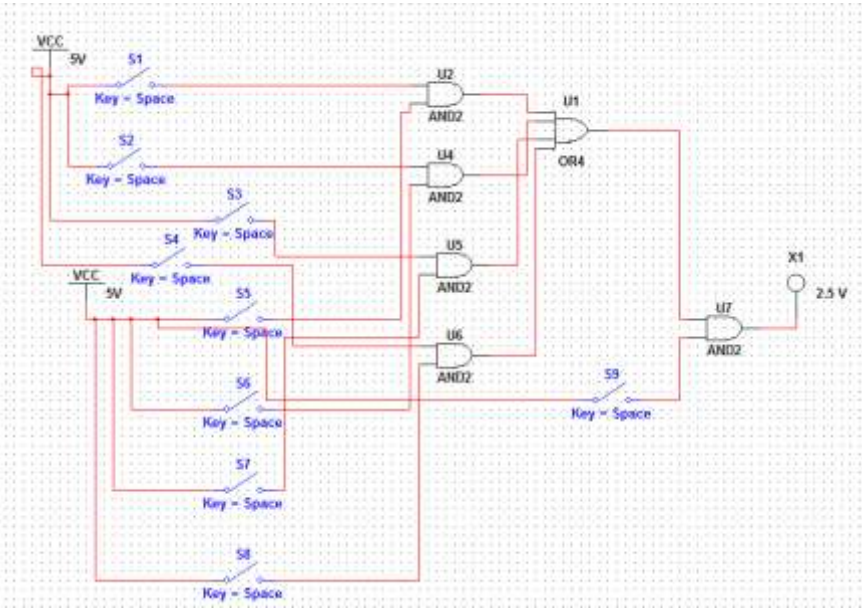


Рис 2.

Задание №2

На рабочем поле среды MULTISIM собрать схему (Рис.3). Замыкая ключи найти те сигналы, при которых на выходе получаем 1. В тетради построить таблицу и заполнить ее данными.



Тема 2.1 Типовые комбинационные цифровые устройства

Практическая работа № 15

Работа сумматора

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть работу сумматора.

Арифметические сумматоры – составная часть арифметико-логических устройств (АЛУ) микропроцессоров (МП). Арифметический сумматор состоит из двух устройств: полусумматора и *n* полных сумматоров. Полный сумматор имеет три входа: *A*, *B* – входы суммируемых операндов, C_i – вход переноса из предыдущего разряда сумматора и два выхода: *S* – выход полного сумматора и C_0 – выход переноса. Полусумматор отличается от полного тем, что у него нет входа переноса из предыдущего разряда. Полусумматор используется в качестве первого разряда арифметического сумматора, а в качестве остальных разрядов – полные сумматоры (рис. 1). Полусумматор – одна из простейших комбинационных логических схем.

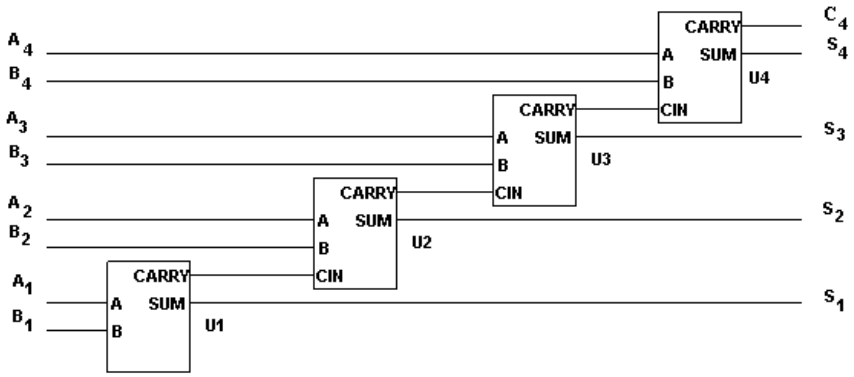


Рис. 1 . Четырехразрядный арифметический сумматор

Последовательность выполнение работы

1. Собрать (нарисовать) схему четырехразрядного арифметического сумматора (рис. 2). Поместить на схему три 16-ричных индикатора и генератор слова.
2. Открыть генератор слова и задать суммируемые числа. Четыре младших разряда каждого генерируемого слова составляют первое слагаемое (операнд). Следующие четыре разряда составляют второе слагаемое (операнд).
3. Запустить процесс моделирования и следить за показаниями индикаторов. Записать суммируемые числа и результат суммирования.

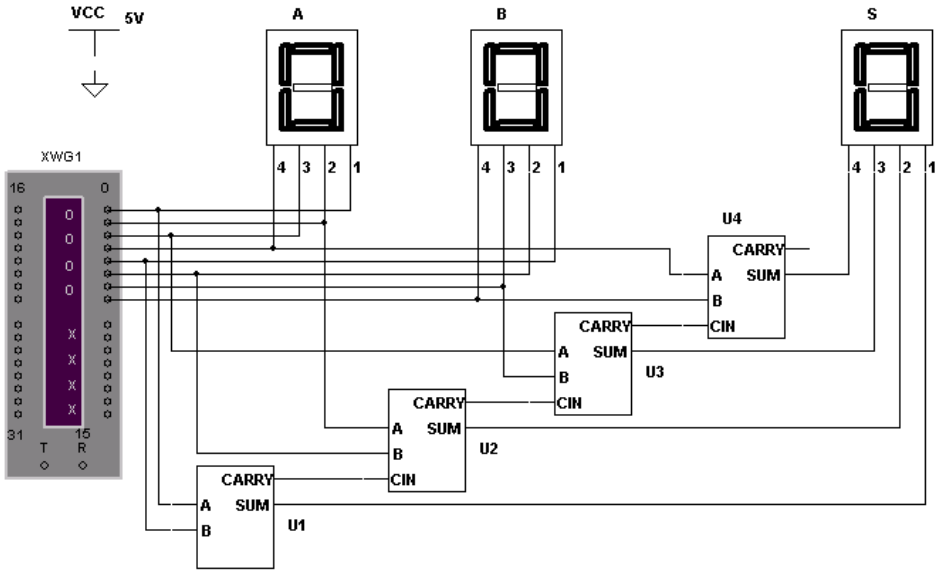
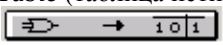
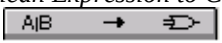


Рис. 2. Схема исследования четырехразрядного сумматора

4. Собрать схему, изображенную на рис. 3, и с помощью логического анализатора, последовательно нажимая кнопки *Circuit to Truth Table* (таблица истинности цепи)  , *Truth Table to Boolean Expression* (булево выражение по таблице истинности)  , *Boolean Expression to Circuit* (создание схемы по булеву выражению)  , получ
   - таблицу истинности полусумматора,
   - логические выражения для выходов *S* и *C*,
   - схемную реализацию логических выражений для выходов *S* и *C*.

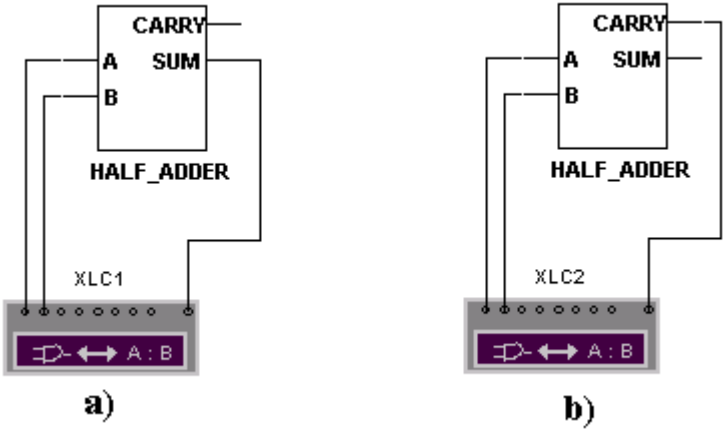


Рис. 3. Схема исследования полусумматора: *а* – выход *S*, *б* – выход *C*

5. Собрать схему, изображенную на рис. 4, и с помощью логического анализатора получить таблицу истинности полного сумматора, логические выражения для выходов *S* и *C* и схемную реализацию логических выражений (см. п. 4).

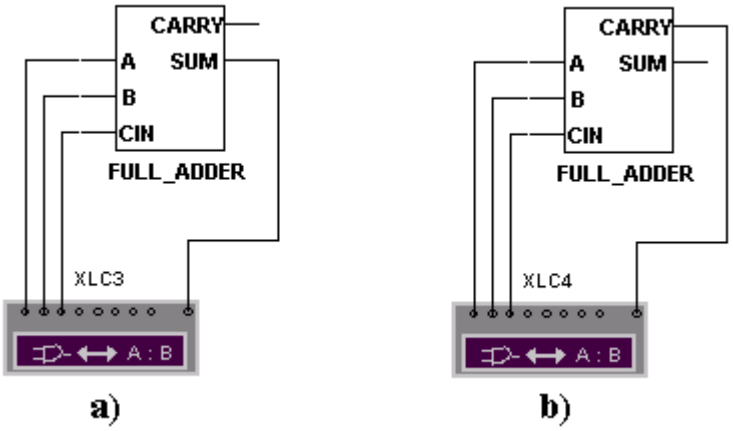


Рис. 4. Схема исследования полного сумматора: *а* – выход *S*, *б* – выход *C*

Описание используемых контрольно-измерительных приборов

Генератор слова (Word Generator) 

Генератор слова (или кодовый генератор) предназначен для генерации 32-разрядных двоичных слов, которые набираются пользователем в 16-ричном коде в строке *Hex* или в двоичном коде в строке *Binary* на панели *Edit* (рис. 53).

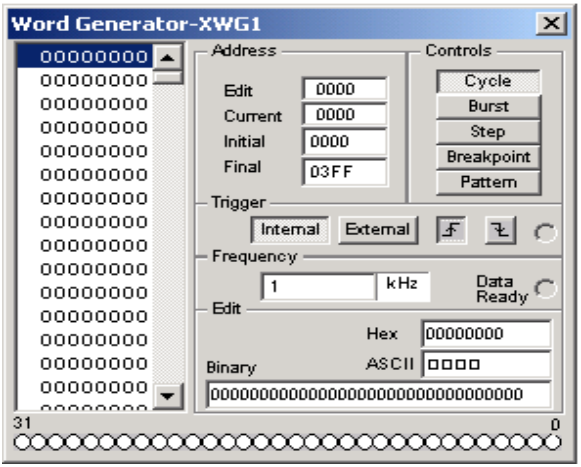


Рис. 5. Лицевая панель генератора слова

В окне, расположенном слева на лицевой панели генератора слова, отображаются 8-разрядные 16-ричные числа от 00000000 до FFFFFFFF (от 0 до 4294967265). Каждая горизонтальная строка представляет 32-разрядное двоичное число. Эти числа поступают в параллельном виде на выходные клеммы генератора, расположенные в нижней части лицевой панели.

Чтобы изменить значение любого бита кодового слова, надо выбрать число (щелкнуть по нему, при этом оно меняет цвет), которое необходимо изменить, и ввести новое значение в строках *HEX*,*ASCII* или *Binary* на панели *Edit*. Измененное кодовое слово отображается на выходных клеммах генератора, расположенных в нижней части лицевой панели.

На панели *Address* расположены четыре окна. Каждое кодовое слово из списка имеет адрес, выраженный 4-разрядным

16-ричным числом.

В окне *Edit* отображается адрес выбранного в таблице слова, в окне *Current* – адрес выдаваемого кодового слова.

В окне *Initial* устанавливается адрес первого кодового слова множества слов, поступающих на выход генератора, в окне *Final* – адрес последнего кодового слова множества слов, поступающих на выход генератора.

Для того чтобы создать множество кодовых слов, выдаваемых генератором слова, нужно ввести адрес первого и последнего слова в окно *Initial* и *Final* соответственно.

На панели *Controls* устанавливается режим выдачи кодовых слов.

Чтобы выдать 32-разрядное слово на выход прибора, надо щелкнуть по одной из кнопок *Step, Burstor Cycle*. Номер этого слова отобразится в окне *Current* на панели *Address*.

Если необходимо выдать только одно слово, следует щелкнуть по кнопке *Step*, если все кодовые слова множества, то щелкнуть по кнопке *Burst*.

Если щелкнуть по кнопке *Cycle*, то будут выдаваться все кодовые слова множества непрерывно циклически. Остановить выдачу слов можно, повторно щелкнув по кнопке *Cycle*.

Если нужно остановить и возобновить выдачу слов с определенного слова, нужно щелкнуть по кнопке *Breakpoint*.

Чтобы установить контрольную точку (Breakpoint), нужно выбрать в списке кодовое слово, на котором следует остановить вывод слов, и затем щелкнуть по кнопке *Breakpoint*. У этого слова появится метка в виде звездочки.

Чтобы удалить контрольную точку, нужно выбрать существующую контрольную точку, затем щелкнуть по кнопке *Breakpoint*.

Можно установить несколько контрольных точек. Контрольные точки могут использоваться как при непрерывной (*Cycle*), так и при однократной (*Burst*) выдаче множества слов.

С помощью кнопки *Pattern* можно создавать новые или использовать

ранее записанные множества кодовых слов.

На панели *Triggering* расположены четыре кнопки, с помощью которых можно установить источник запускающего сигнала (внутренний (*Internal*) или внешний (*External*)) и фазу запускающего сигнала (по переднему или заднему фронту).

На панели *Frequency* устанавливается тактовая частота генератора слова в герцах, кило- или мегагерцах. Кодовые слова поступают на выход генератора с каждым тактом генератора. Рядом расположена клемма, на которую выдается сигнал готовности выдавать данные.

Логический преобразователь (Logic Converter) 

На лицевой панели преобразователя (рис. 6) расположены клеммы-индикаторы входов *A, B, C, …,H* и клемма выхода *Out*, окно для отображения таблицы истинности исследуемой схемы, строка для отображения ее булева выражения и панель *Conversions*.

На панели *Conversions* расположены шесть кнопок, используемых для получения:

-  – таблицы истинности исследуемого устройства,
-  – булева выражения, реализуемого исследуемым устройством,
-  – минимизированного булева выражения,

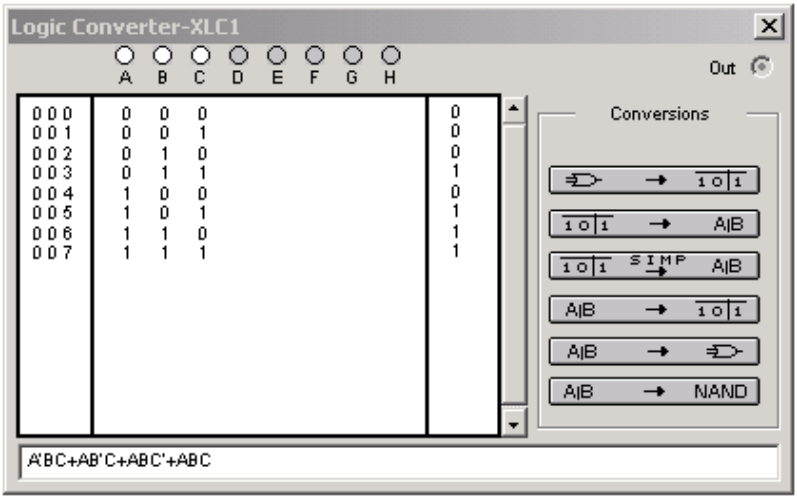


Рис. 6. Лицевая панель логического преобразователя

-  – таблицы истинности по булевому выражению;
-  – схемы устройства по логическому выражению на логических элементах без ограничения их типа,
-  – создания схемы устройства только на логических элементах «И-НЕ».

Содержание отчета

Отчет должен включать:

1) название пункта работы,
2) исследуемую схему,
3) результат моделирования.

Тема 2.1 Типовые комбинационные цифровые устройства

Практическая работа № 16

Работа компаратора

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть работу компаратора.

Материальное обеспечение: Методические указания

Задание:

Цели: Изучить работу и принцип действия цифрового компаратора.

Ход работы

1. Построить схему компаратора на логических элементах. Подключить к схеме логический анализатор. Проверить график работы компаратора.

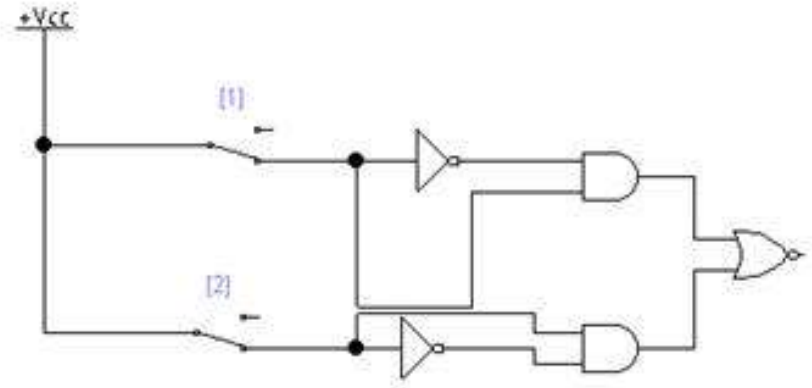


Рисунок 1 – Схема компаратора на логических элементах.

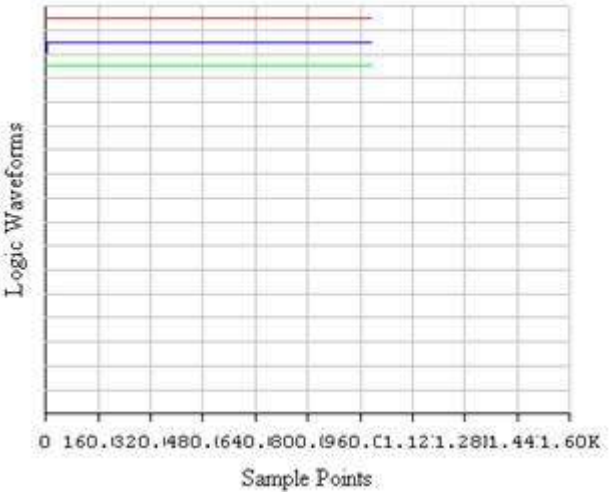


Рисунок 2 – Результаты с анализатора.

Цифровые компараторы выполняют сравнение двух чисел заданных в двоичном коде. Компараторы определяют либо равенство двух двоичных чисел с одинаковым количеством разрядов, либо вид неравенства (A>B, A<B). Цифровые компараторы имеют два входа и три выхода.

Схема одноразрядного компаратора представляет собой структуру логического элемента (исключающее «или»).

При A=B F=0;

При A B C=1;

При A B D=1;

2. В тетради построить схемы (рис 1 и рис 3). Записать теорию о компараторе.

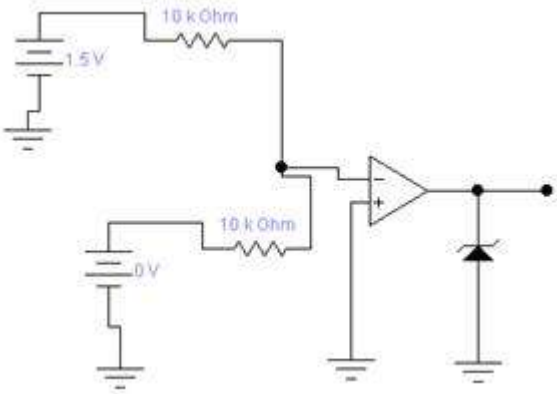


Рисунок 3 – Схема компаратора на операционном усилителе.
Компаратор сравнивает два напряжения и указывает большее из них. Если напряжение на А больше, чем на В, на $U_{ВЫХ}$ вырабатывается логическая единица. Если напряжение на В больше, то на $U_{ВЫХ}$ вырабатывается логический ноль.

3. Нарисовать схему исследования функции «И» (рис. 4).

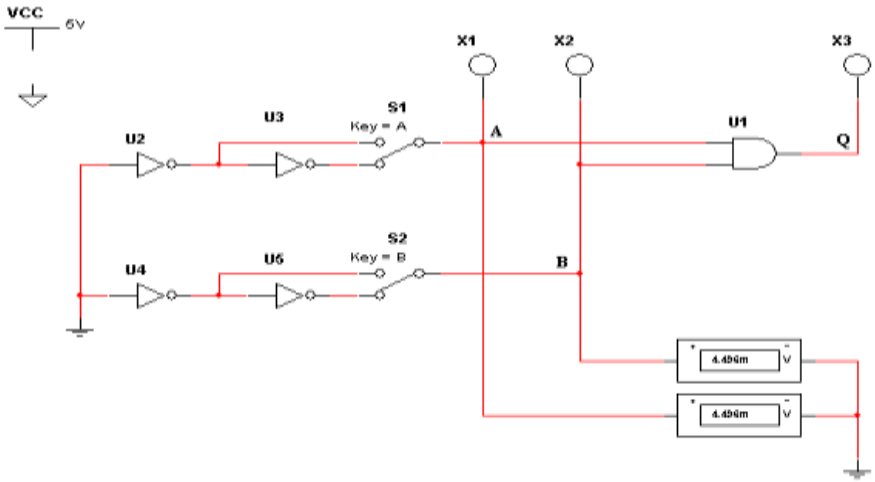


Рис. 4. Схема исследования функции «2И»

Схема содержит исследуемую функцию «И» (*U1*), два двухпозиционных переключателя (*S1, S2*), управляемые клавишами *A* и *B* (заглавные буквы латинского алфавита), источники сигналов логической единицы (*U2*,*U4*), логического нуля (*U3*,*U5*), три светодиода (*X1, X2, X3*), два вольтметра и источник постоянного напряжения 5В (*VCC*).

4. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

5. Подать на входы схемы «И» все возможные комбинации уровней сигналов *A* и *B* с помощью переключателей *S1* и *S2*. И для каждой комбинации зафиксировать показания вольтметров и уровни входных сигналов *A* и *B* и уровень выходного сигнала *Q* (логическая единица – соответствующий светодиод X_i светится, логический ноль – соответствующий светодиод X_i не светится). Результаты измерений занести в таблицу истинности (табл. 1).

Таблица 1

Входы		*Выход*
A	**B**	**Q**
0	0	
0	1	
1	0	
1	1	

Исследование логической функции «И-НЕ»

6. Нарисовать схему исследования функции «2И-НЕ» (рис. 5).

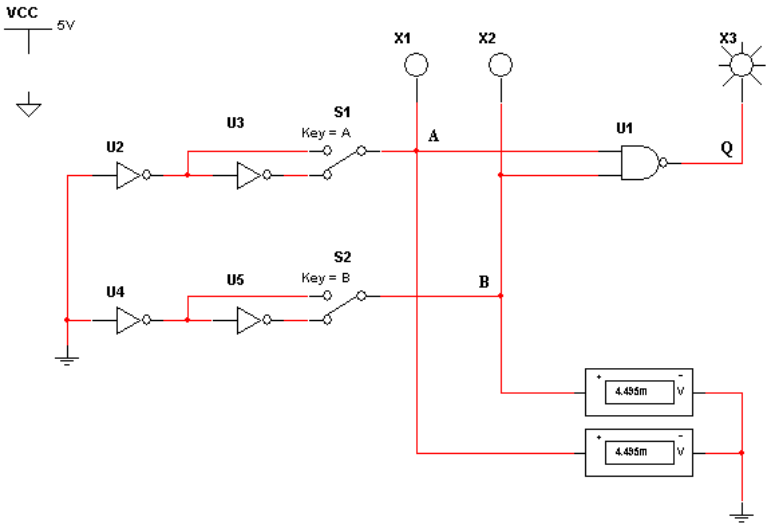


Рис. 5. Схема исследования функции «2И-НЕ»

7. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

8. Подать на входы схемы «И-НЕ» все возможные комбинации уровней сигналов *A* и *B* с помощью переключателей *S1* и *S2*. И для каждой комбинации зафиксировать показания вольтметров, уровни входных сигналов *A* и *B* и уровень выходного сигнала *Q* (логическая единица – соответствующий светодиод X_i светится, логический ноль – соответствующий светодиод X_i не светится). Результаты измерений занести в таблицу истинности (табл. 3).

Таблица 2

Входы		*Выход*
A	**B**	**Q**
0	0	

0	1	
1	0	
1	1	

Исследование логической функции «ИЛИ»

9. Нарисовать схему исследования функции «2ИЛИ» (рис. 6).

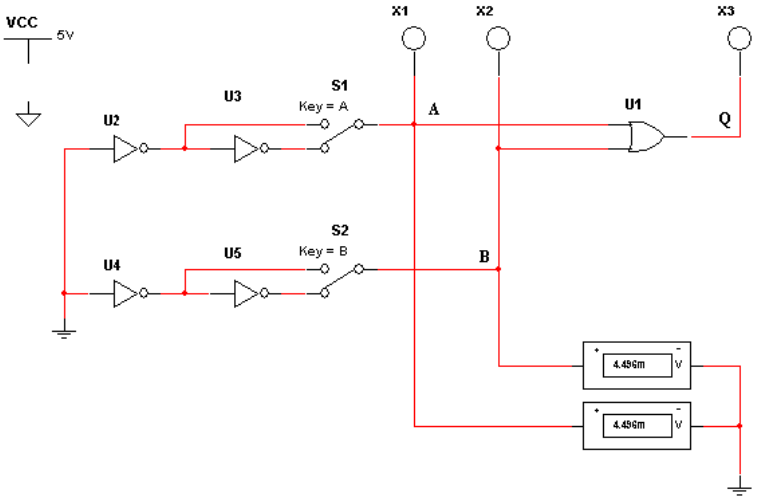


Рис. 6. Схема исследования функции «2ИЛИ»

10. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

11. Подать на входы схемы «ИЛИ» все возможные комбинации уровней сигналов *A* и *B* с помощью переключателей *S1* и *S2*. И для каждой комбинации зафиксировать показания вольтметров, уровни входных сигналов *A* и *B* и уровень выходного сигнала *Q* (логическая единица – соответствующий светодиод X_i светится, логический ноль – соответствующий светодиод X_i не светится). Результаты измерений занести в таблицу истинности (табл. 4).

Таблица 3

Входы		*Выход*
A	**B**	**Q**
0	0	
0	1	
1	0	
1	1	

Проработать задания по методическим указаниям

Тема 2.2. Последовательные цифровые устройства

Практическая работа № 17,18,19

Работа RS-триггера, D-триггеры

Работа Т-триггера

Работа JK-триггера

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть работу RS-триггера, D-триггеров ,Т-триггера, JK-триггера.

Материальное обеспечение: Методические указания

1. Собрать схемы рис 2., рис 3.
2. Замыкая ключи проанализировать работу RS – триггера.
3. Построить в тетради схему рис 1, таблицы истинности
4. Собрать схему рис 4.
5. Замыкая ключи проанализировать работу D – триггера
6. В тетради построить схему рис 4
7. Выполнить задание для закрепления

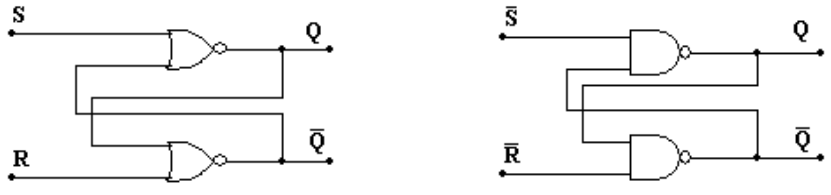
Триггерами называются устройства, обладающие двумя устойчивыми состояниями ($Q = 1$ и $Q = 0$) и способные находиться в одном из них сколь угодно долго и переходить из одного состояния в другое под воздействием внешних сигналов. В каком из этих состояний окажется триггер, зависит от сигналов на входах триггера и от его предыдущего состояния, т.е. он имеет память. Таким образом, триггер – элементарная ячейка памяти.

Тип триггера определяется алгоритмом его работы, в зависимости от которого триггер может иметь установочные, информационные и управляющие входы. Установочные входы обуславливают состояние триггера независимо от состояния других входов. Входы управления разрешают запись данных, подающихся на информационные входы. Наиболее распространенными являются триггеры *RS*-, *JK*-, *D*- и *T*-типов.

RS-триггер – простейший автомат с памятью, который может находиться в двух состояниях. Триггер имеет два установочных входа: установки *S* (*set* – установка) и сброса *R* (*reset* – сброс), на которые подаются входные сигналы от внешних источников. При подаче на установки ак-

тивного логического уровня триггер устанавливается в единицу ($Q = 1$, $Q' = 0$, здесь штрих означает инвертирование), при подаче активного уровня на вход сброса триггер устанавливается в ноль ($Q = 0$, $Q' = 1$). Если на оба установочных входа подать пассивный логический уровень, то триггер сохраняет предыдущее состояние выходов: $Q = 1$ или $Q = 0$. Каждое состояние устойчиво и поддерживается за счет действия обратных связей. Подача активного уровня одновременно на оба установочных входа запрещена, так как триггер не может быть установлен в ноль и единицу.

RS-триггер может быть выполнен на элементах «ИЛИ-НЕ» или «И-НЕ» (рис. 1).



а *б*

Рис. 1. *RS*-триггер: *а* – на элементах «ИЛИ-НЕ», *б* – на элементах «И-НЕ»

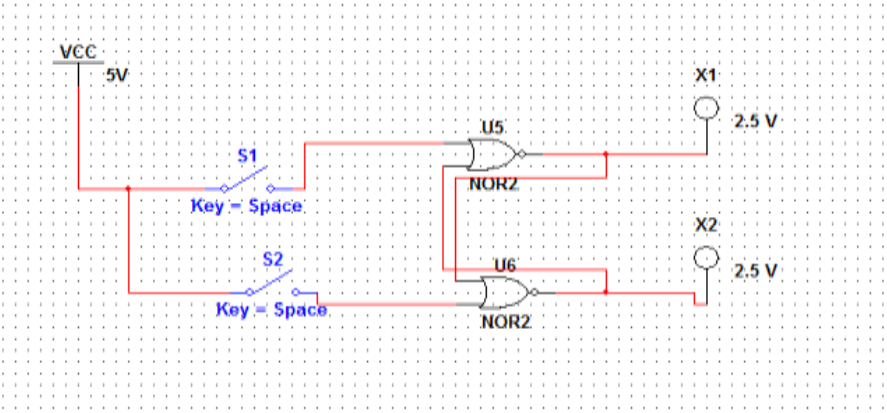


Рис 2 RS – триггер на элементах ИЛИ-НЕ

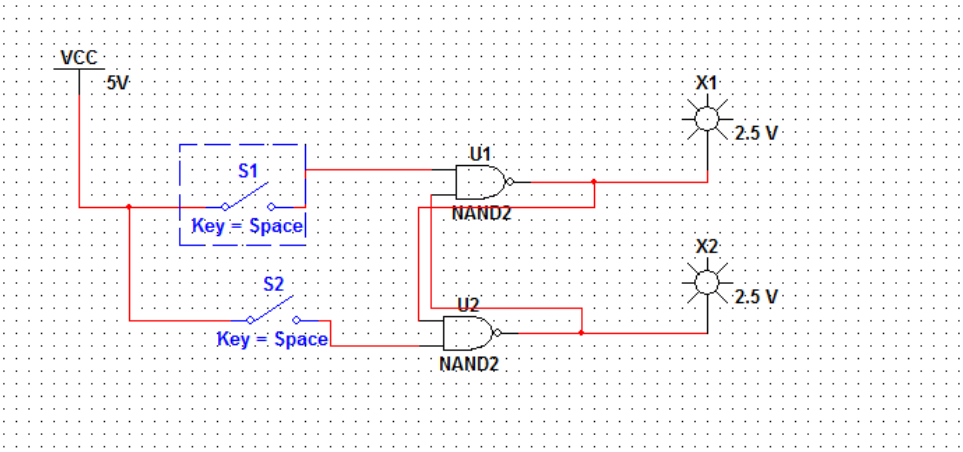


Рис 3 RS – триггер на элементах И-НЕ

Для *RS*-триггеров, выполненных на элементах «ИЛИ-НЕ», активным уровнем на управляющих входах является уровень логической единицы, а на элементах «И-НЕ» – уровень логического нуля.

RS-триггер – основной узел построения последовательных схем. Условия переходов триггеров из одного состояния в другое можно описать табличным, аналитическим или графическим способами. Табличное описание работы *RS*-триггера на элементах «ИЛИ-НЕ» и «И-НЕ» представлено в табл. 1 и 2 соответственно, где Q_t – предшествующее состояние выхода; Q_{t+1} – новое состояние, устанавливающееся после перехода; - – неопределенное состояние.

Таблица 1

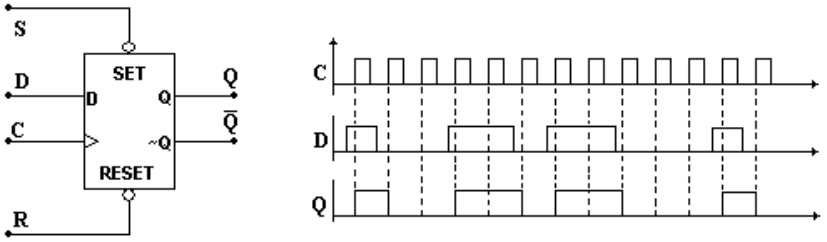
R	S	Q_{t+1}
0	0	Q_t
1	0	0
0	1	1
1	1	–

Таблица 2

R	S	Q_{t+1}
0	0	–
1	0	1
0	1	0
1	1	Q_t

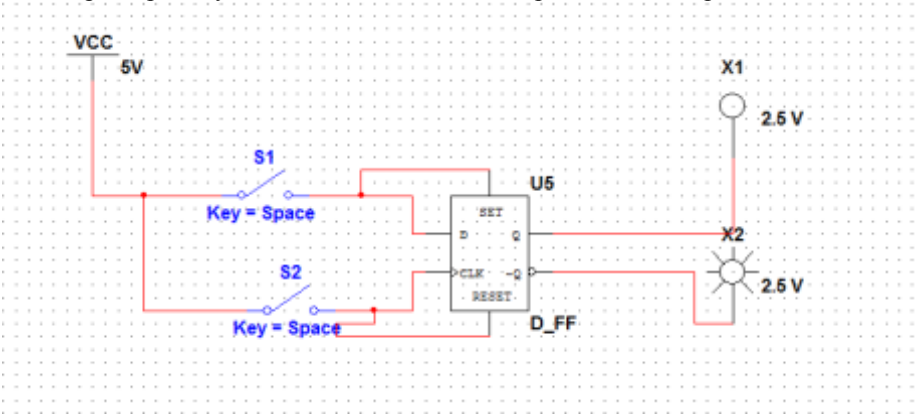
D-триггер имеет один информационный вход *D* (*data* – данные) и один счетный вход *C*. Информация с входа *D* записывается в триггер по положительному перепаду импульса на счетном входе и сохраняется до следующего положительного перепада. Кроме счетного *C* и информационного *D* входов, у триггера есть два асинхронных установочных входа *R* и *S*. Установочные входы приоритетные. Активный уровень сигнала на входе *S* устанавливает триггер в состояние единица (*Q*=1), а на входе *R* в состояние ноль (Q=0), независимо от сигналов на остальных входах.

Условное обозначение *D*-триггера с диаграммами входных и выходных сигналов приведено на рис. 4.



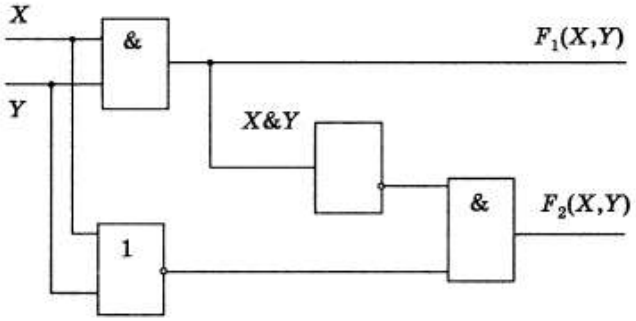
а *б*

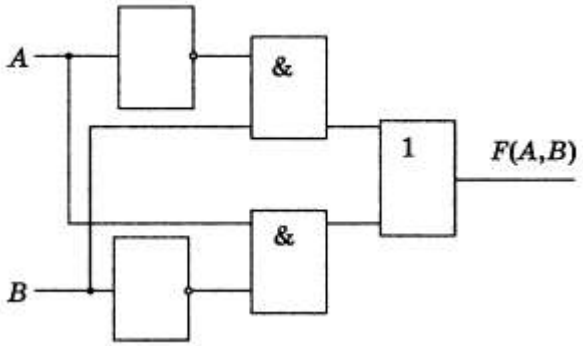
Рис. 4. *D*-триггер: *а* – условное обозначение; *б* – временные диаграммы



Задание для закрепления

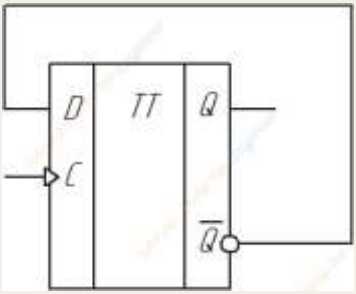
Собрать схемы в Multisim. Проверить работу схем. Построить таблицы истинности к схемам.





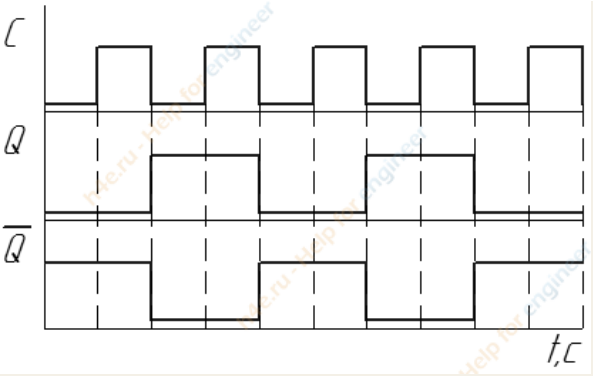
Триггер Т –типа (счетный триггер)

Т-триггеры можно построить с помощью любого двухступенчатого триггера. Наличие двух ступеней позволяет избавиться от запрещенных состояний. Ранее мы рассматривали принцип работы D-триггера, именно по этому построение счетного t-триггера будем осуществлять на его базе. Он состоит из входа С (ранее синхронизирующий) и выхода Q. Чтобы произвести синтез необходимого нам устройства, нужно инверсный выход соединить со входом:



Счетным Т-триггер называют потому что он считает количество импульсов, которое поступает к нему на вход. Правда счет ведется лишь до одного. При повторной подаче сигнала на вход – значение выхода сбрасывается. Это свойство дало возможность использовать устройство, как делитель частоты. То есть с выхода будем снимать импульсы вдвое меньшей частоты, чем было на входе.

Для построения счетного устройства мы использовали д-триггер с работой по заднему фронту. Соответственно и полученное будет работать по тому же принципу, временная диаграмма Т-триггера имеет следующий вид:



Собранный T-trigger на логических элементах представлен ниже, рассматривая его, легко понять принцип работы. Синий провод означает нулевой уровень напряжения, красный – единица. Работает устройство при подаче импульсов с определенной частотой на вход С. Начинает происходить подсчет входящего сигнала, и по заднему фронту, выход меняет значение:

1. Собрать схему рис. 1.

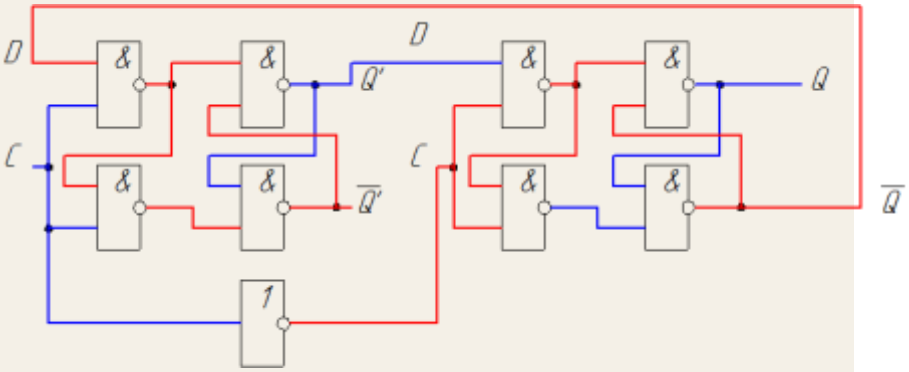
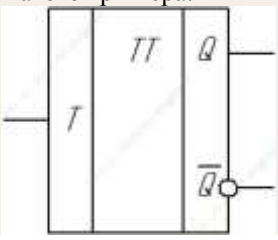
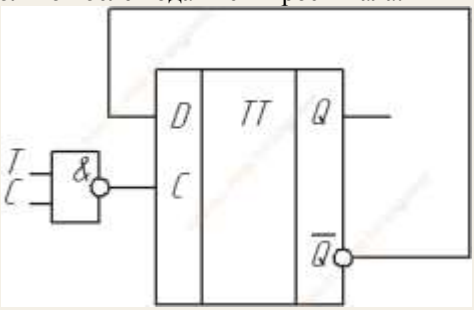


Рис. 1

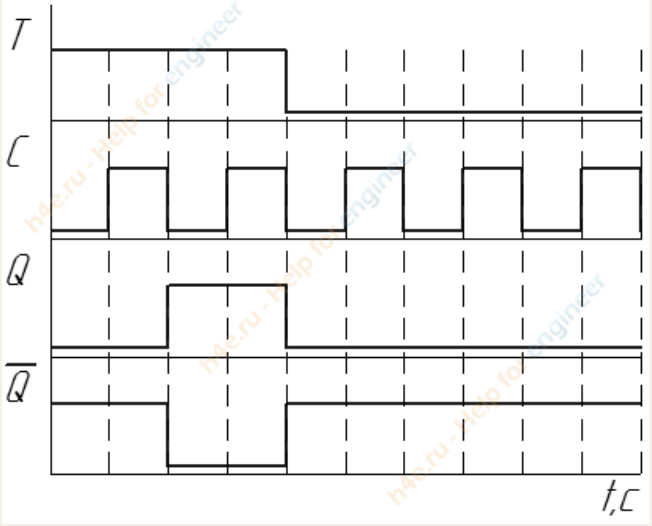
Обозначение ничем не отличается от ранее рассмотренных, где ТТ означает наличие двух ступенчатого триггера:



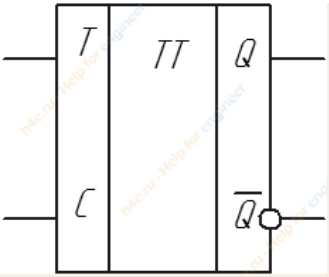
Все это мы говорили об асинхронном т-триггере, это означает что его работа не контролируется никаким дополнительным сигналом, а он начинает выполнять свои функции сразу при подачи С. Небольшая модификация позволяет получить синхронный t-триггер, теперь он включится в работу только после подачи синхросигнала:



Временная диаграмма синхронного Т триггера приобретает чуть иной характер, появляется прямая зависимость выхода от синхронизирующего входа:



Внешнее обозначение так же по простой логике чуть изменяется:



Т-триггер или счетный триггер – устройство, осуществляющее счетный режим. Такие схемы можно построить на основе JK-триггеров или D-триггеров.

2. Собрать схему рис. 2 и проанализировать осциллограмму.

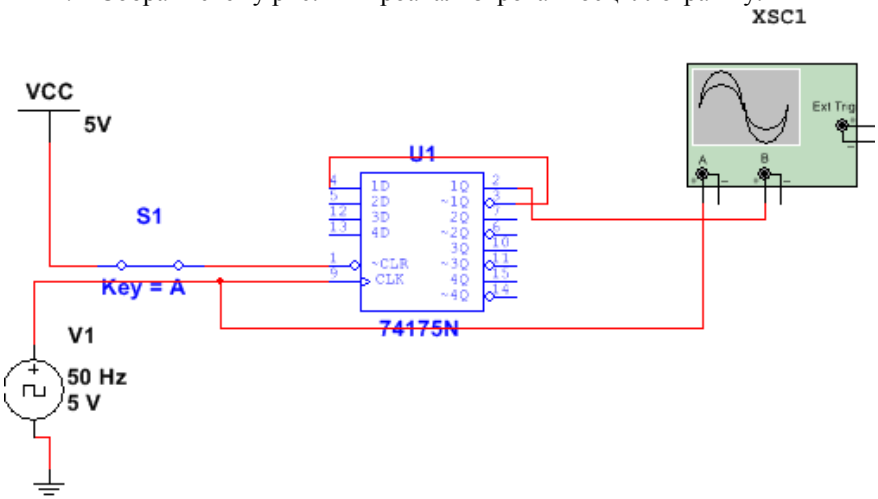
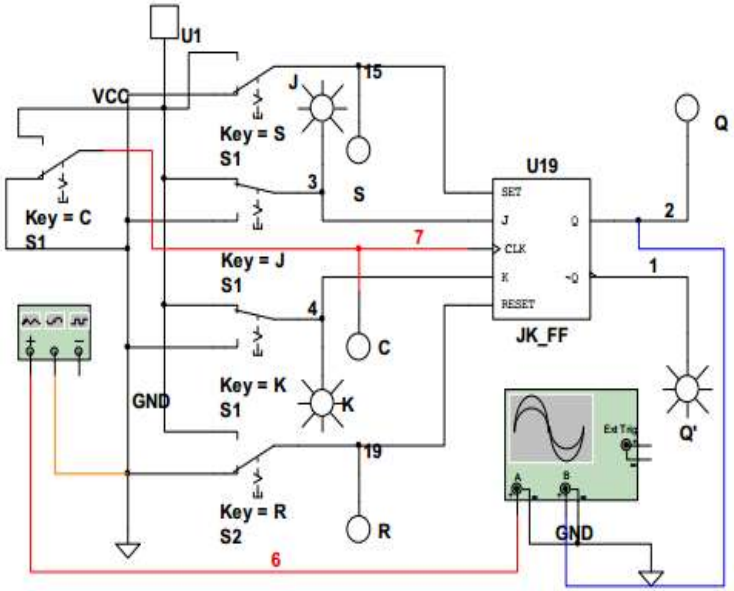
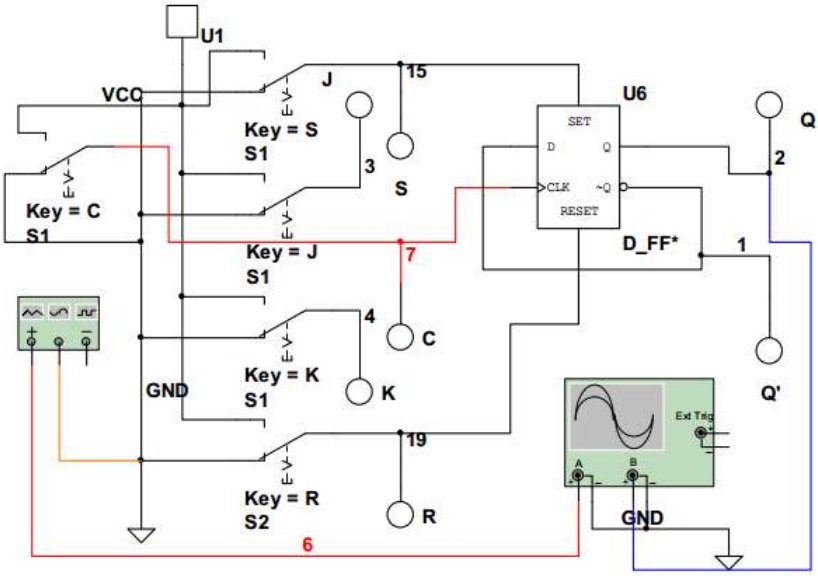


Рис. 2

3. Исследовать Т-триггер на основе JK – триггера. Построить таблицу истинности.



4. Исследовать Т-триггер на основе D – триггера. Построить таблицу истинности.



Тема 2.2. Последовательные цифровые устройства

Практическая работа № 20,21

Работа счетчиков

Работа регистра

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть работу счетчика, регистра

Материальное обеспечение: Методические указания

Задание:

Исследование регистра

Наиболее распространенным узлом цифровой техники и устройств автоматики являются регистры. Регистры строятся на базе синхронизированных одно- и двухступенчатых RS , D и JK-триггеров.

Регистры с параллельным приемом и выдачей информации часто служат для хранения информации и называются регистрами памяти или хранения.

1. В мультисиме построить схему четырехразрядного регистра сдвига с автоматической записью единицы в первый разряд регистра (рис. 1).

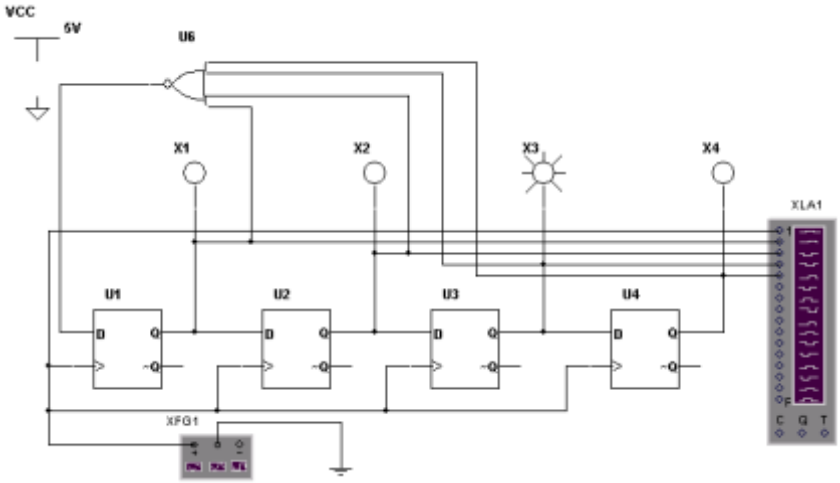


Рис. 1. Четырехразрядный регистр сдвига с автоматической записью единицы

Схема содержит четыре D-триггера, четыре светодиода, одну логическую схему 4ИЛИ-НЕ, функциональный генератор и логический анализатор. Логическая схема 4ИЛИ-НЕ служит для автоматической записи 1 в регистр. На выходе этой схемы единица будет только тогда, когда все разряды регистра будут находиться в нулевом состоянии.

2. Открыть окно функционального генератора и установить вид генерируемых сигналов – прямоугольные импульсы, генерируемую частоту - 1000 Гц, амплитуду генерируемых импульсов –5 вольт.
3. Открыть окно логического анализатора, дважды щелкнув по иконке логического анализатора.

4.

Запустить процесс моделирования, нажав кнопку  на панели инструментов и в появившемся меню выбрать команду **Run**.

5.

Построить в тетради схему рис.1, подписать все элементы, находящиеся на схеме.

Исследование суммирующего счетчика

1.

Нарисовать схему четырехразрядного счетчика, считающего в прямом направлении (рис. 2).

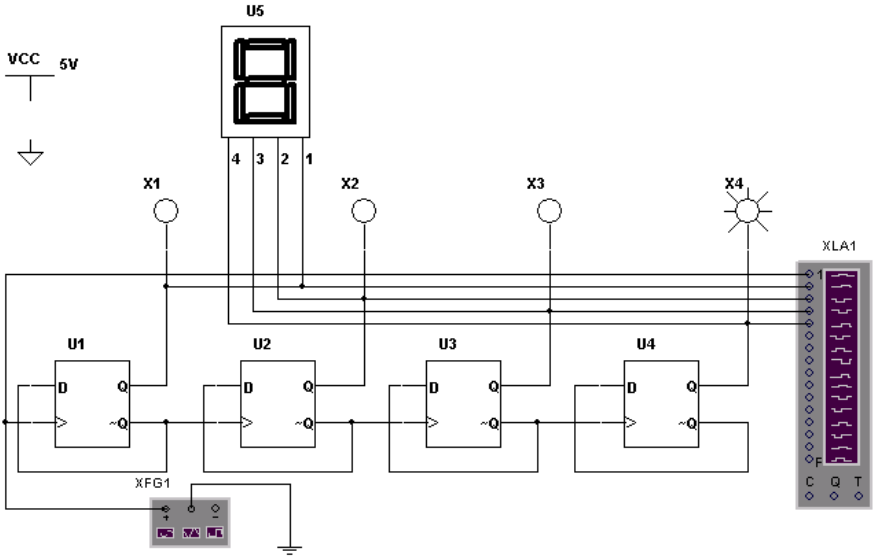


Рис. 2 . Четырехразрядный суммирующий счетчик

Поместить на схему четыре D-триггера, четыре светодиода, функциональный генератор, логический анализатор и 16-ричный индикатор.

2.

Открыть окно логического анализатора, щелкнув по иконке логического анализатора.

3. Запустить процесс моделирования, нажав кнопку  на панели инструментов и в появившемся меню выбрать команду Run.
4. Построить в тетради схему рис.1, подписать все элементы, находящиеся на схеме.
5. Наблюдать за показаниями 16-ричного индикатора и сравнить его показания с соответствующими состояниями светодиодов.

Исследование вычитающего счетчика

1. В мультисиме построить схему четырехразрядного счетчика, считающего в обратном направлении (рис. 3).

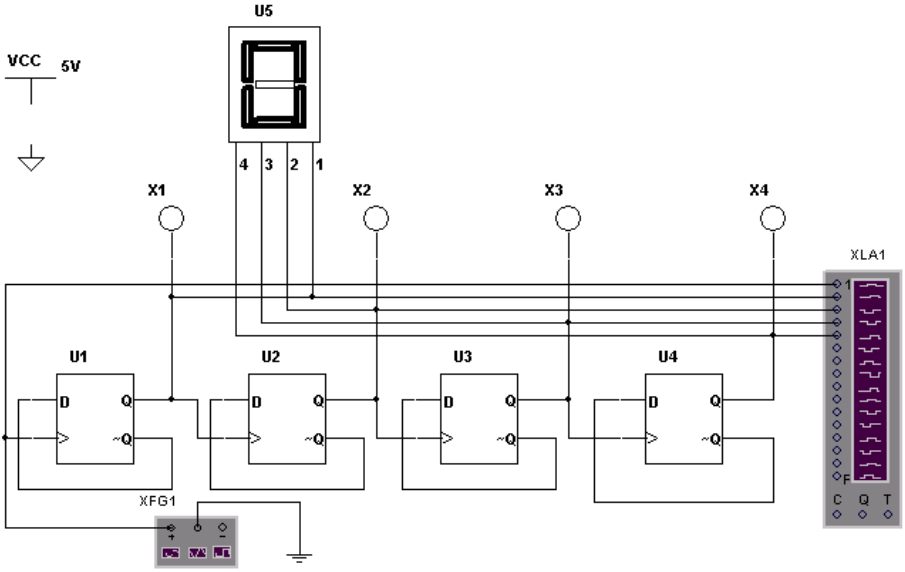


Рис. 3. Четырехразрядный вычитающий счетчик

2. Открыть окно логического анализатора, щелкнув по иконке логического анализатора.

3. Запустить процесс моделирования, нажав кнопку  на панели инструментов и в появившемся меню выбрать команду Run.
4. Построить в тетради схему рис.1, подписать все элементы, находящиеся на схеме.
5. Наблюдать за показаниями 16-ричного индикатора и сравнить его показания с соответствующими состояниями светодиодов.

Тема 3.1 Основы микропроцессорных систем

Практическая работа № 22,23

Структура устройства управления

Типы микропроцессоров

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть устройства управления и типы микропроцессоров

Материальное обеспечение: Методические указания

Задание:

Составить конспект в тетради по типам микропроцессоров и устройствам управления.

Микропроцессором называется программно-управляемое устройство, предназначенное для обработки цифровой информации и управления этим процессом, реализованное в виде одной или нескольких БИС (СБИС).

Для МП на БИС или СБИС характерны:

• простота производства (по единой технологии);
• низкая стоимость (при массовом производстве);
• малые габариты (пластина площадью несколько квадратных сантиметров или кубик со стороной несколько миллиметров);
• высокая надежность;
• малое потребление энергии;

Микропроцессор выполняет следующие функции:

• чтение и дешифрацию команд из основной памяти;
• чтение данных из ОП и регистров адаптеров внешних устройств;
• прием и обработку запросов и команд от адаптеров на обслуживание ВУ;
• обработку данных и их запись в ОП и регистры адаптеров ВУ;
• выработку управляющих сигналов для всех прочих узлов и блоков ПК.

Основные характеристики МП.

1.Под *разрядностью* МП понимается стандартная длина слова, с которым оперируют составные части МП.
По разрядности МП бывают с фиксированной и с изменяемой разрядностью слова. При фиксированной разрядности наиболее распространены МП с длиной слова 8 и 16 бит. Во втором случае возможно построение 8-, 16-, 24-, 32-разрядных МП из секций разрядностью 2,4 и 8.
2. *Производительность* МП определяется временем решения ряда тестовых задач и зависит от быстродействия выполнения простых операций, разрядности, числа регистров общего назначения, структуры схем ввода-вывода и др. факторов.
3. *Система команд* является отличительным признаком для любого МП. Она отражает функциональные возможности устройства. Система команд МП может содержать как малое число команд (восемь), так и большое число (до 200) основных команд. Состав команд не является нормализованным. Система команд МП объединяет арифметические, логические, передачи данных, вызова подпрограмм, возврата из подпрограмм, команды ветвления, сдвигов и др.
4. *Объем адресуемой памяти* характеризует информационные возможности МП и к настоящему времени достигает сотен Мбайт, что ранее было доступно только универсальным ЭВМ.
Первый микропроцессор был выпущен в 1971 г . фирмой Intel (США) – МП 4004. В настоящее время выпускается несколько сотен различных микропроцессоров, но наиболее популярными и распространенными являются микропроцессоры фирмы Intel и Intel -подобные.
Все микропроцессоры можно разделить на три группы:
* МП типа CISC (Complex Instruction Set Computing) с полным набором команд ;
* МП типа RISC (Reduced Instruction Set Computing) с сокращенным набором команд ;
* МП типа MISC (Minimum Instruction Set Computing) с минимальным набором команд и весьма высоким быстродействием (в настоящее время эти модели находятся в стадии разработки).
Микропроцессоры типа CISC
Особенности:
- предусматривается возможность работы в вычислительной сети;
- имеется возможность многозадачной работы (многопрограммность) и сопутствующая ей защита памяти;
- микропроцессоры могут работать в двух режимах: реальном (Real mode) и защищенном (Protected mode). В реальном режиме имитируется (эмулируется) работа МП 8086, естественно, однозадачная. В защищенном режиме возможна многозадачная работа с непосредственным досту-

пом к расширенной памяти и с защитой памяти, отведенной задачам, от посторонних обращений;

Микропроцессоры типа RISC

Микропроцессоры типа RISC содержат набор только простых, чаще всего встречающихся в программах команд. При необходимости выполнения более сложных команд в микропроцессоре производится их автоматическая сборка из простых. В этих МП на выполнение каждой простой команды за счет их наложения и параллельного выполнения тратится 1 машинный такт (на выполнение даже самой короткой команды из системы CISC обычно тратится 4 такта).

Функционально МП состоит из двух частей:

* операционной, содержащей устройство управления, арифметико-логическое устройство и микропроцессорную память (за исключением нескольких адресных регистров);
* интерфейсной, содержащей адресные регистры МПП, блок регистров команд, схемы управления шиной и портами.

Современные микропроцессоры имеют несколько групп регистров в микропроцессорной части, работающих с различной степенью опережения, что позволяет выполнять операции в конвейерном режиме. Такая организация МП дает возможность значительно повысить его эффективное быстродействие.

Тема 3.4 Основы программирования на языке низкого уровня

Практическая работа № 24

Основы программирования на языке низкого уровня

Формируемые компетенции:

- ПК 4.4. Рассчитывать параметры типовых схем и устройств.
- ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

Цель работы:

Рассмотреть основы программирования на языке низкого уровня

Материальное обеспечение: Методические указания

Задание:

Составить конспект в тетради по основам программирования на языке низкого уровня.

Язык ассемблера — тип языка программирования низкого уровня, см. подробнее о происхождении и использовании термина.

Команды языка ассемблера один в один соответствуют командам процессора и, фактически, представляют собой удобную символьную форму записи (мнемокод) команд и аргументов. Также, язык ассемблера обеспе-

чивает связывание частей программы и данных через метки, выполняемое при ассемблировании (для каждой метки высчитывается адрес, после чего каждое вхождение метки заменяется на этот адрес).

Каждая модель процессора, в принципе, имеет свой набор команд и соответствующий ему язык (или диалект) ассемблера.

Обычно программы или участки кода пишутся на языке ассемблера в случаях, когда разработчику критически важно оптимизировать такие параметры, как быстродействие (например, при создании драйверов) и размер кода (загрузочные сектора, программное обеспечение для микроконтроллеров и процессоров с ограниченными ресурсами, вирусы, навесные защиты).

Связывание ассемблерного кода с другими языками

Большинство современных компиляторов позволяют комбинировать в одной программе код, написанный на разных языках программирования. Это позволяет быстро писать сложные программы используя высокоуровневый язык, не теряя быстродействия в критических ко времени задачах, используя для них части написанные на языке ассемблера. Комбинирование достигается несколькими приемами:

Вставка фрагментов на языке ассемблера в текст программы (специальными директивами языка) или написание процедур на языке ассемблера. Способ хороший для несложных преобразований данных, но полноценного ассемблерного кода — с данными и подпрограммами, включая подпрограммы с множеством входов и выходов, не поддерживаемых высокоуровневыми языками, с помощью него сделать нельзя.

Модульная компиляция. Большинство современных компиляторов работают в два этапа. На первом этапе каждый файл программы компилируется в объектный модуль. А на втором объектные модули линкуются (связываются) в готовую программу. Прелесть модульной компиляции состоит в том что каждый объектный модуль будущей программы может быть полноценно написан на своем языке программирования и скомпилирован своим компилятором (ассемблером).

Синтаксис

Единого стандарта для синтаксиса языков ассемблера не существует, конкретный разработчик волен установить свои собственные синтаксические правила. Однако существуют традиционные подходы, которых при-

держиваются языки ассемблера для наиболее распространённых процессорных архитектур, своего рода стандарт de facto. Так основными стандартами являются стандарты — Intel и AT&T.

Каждая инструкция записывается в отдельной строке.

Полный формат каждой строки инструкций следующий:

label: code ; comment

где label — название метки; code — собственно, инструкция языка ассемблера; comment — комментарий.

При этом один или два компонента строки могут отсутствовать, то есть строка может состоять, к примеру, только из комментария, или содержать только метку или инструкцию.

Объекты, над которыми производятся действия, это регистры процессора и участки оперативной памяти. Обозначения для них также являются частью синтаксиса.

Ассемблерная инструкция, состоит из мнемоники команды и списка аргументов через запятую (один, два или три в зависимости от инструкции). Мнемоникой команды служит трёх- или четырёхбуквенными сокращениями их аналогов, обычно на английском языке, например:

jmp — продолжать выполнение с нового адреса памяти (от англ. jump - прыжок)
mov — переместить данные (от англ. move - передвинуть)
sub — получить разность двух значений (от англ. subtract - вычесть)
xchg — обменять значения в регистрах/ячейках памяти (от англ. exchange - обмен)

От ассемблера к ассемблеру меняется синтаксис аргументов, но мнемоники, обычно, остаются одинаковыми (такими какие используются в оригинальной спецификации процессора), за исключением двух случаев: Если ассемблер использует кроссплатформенный AT&T-синтаксис, то оригинальные мнемоники приводятся к синтаксису AT&T.

Если изначально существовало два стандарта записи мнемоник (система команд была наследована от процессора другого производителя).

Например процессор Zilog Z80 наследовал систему команд Intel i8080, расширил ее и поменял мнемоники (и обозначения регистров) на свой лад. Например сменил интеловские «mov» на «ld» (команда перемещения данных). Процессоры Motorola Fireball наследовали систему команд Z80, несколько её урезав. Вместе с тем, Motorola официально вернулась к мнемоникам Intel. И в данный момент половина ассемблеров для Fireball работает с интеловскими мнемониками, а половина с мнемониками Zilog.

Текст программ может быть дополнен директивами ассемблера (параметры, влияющие на процесс ассемблирования и свойства выходного файла).

Каждый ассемблер имеет собственные директивы.

Для упрощения и ускорения написания программ на языке ассемблера служат макросы.

Достоинства языка ассемблера

Максимально оптимальное использование средств процессора, использование меньшего количества команд и обращений в память, и как следствие — большая скорость и меньший размер программы

Использование расширенных наборов инструкций процессора (MMX, SSE, SSE2, SSE3)

Доступ к портам ввода-вывода и особым регистрам процессора (в большинстве ОС эта возможность доступна только на уровне модулей ядра и драйверов)

Возможность использования самомодифицирующегося (в том числе перемещаемого) кода (под многими платформами эта возможность недоступна, так как запись в страницы кода запрещена, в том числе и аппаратно, однако в большинстве общедоступных систем из-за их врожденных недостатков имеется возможность исполнения кода содержащегося в сегменте (секции) данных, куда запись разрешена)

Максимальная «подгонка» для нужной платформы

NB: Последние технологии безопасности, внедряемые в операционные системы и компиляторы, не позволяют делать самомодифицирующего кода, так как исключают одновременную возможность исполнения программы и запись в одном и том же участке памяти (технология W^X).

Технология W^X используется в OpenBSD (где и появилась), в других BSD-системах, в Linux; в Microsoft Windows (начиная с Windows XP SP2)

используется схожая технология DEP.

Недостатки

Большие объемы кода, большое число дополнительных мелких задач, меньшее количество доступных для использования библиотек, по сравнению с языками высокого уровня

Трудоёмкость чтения и поиска ошибок (хотя здесь многое зависит от комментариев и стиля программирования)

Зачастую компилятор языка высокого уровня, благодаря современным алгоритмам оптимизации, даёт более эффективную программу (по соотношению качество/время разработки).

Непереносимость на другие платформы (кроме совместимых).

Ассемблер более сложен для совместных проектов.

Пример программы на языке ассемблера

Пример программы для операционной системы DOS на процессоре семейства Intel x86, выдающей на экран приветствие (написан на TASM):

```
mov bx,1 ; указание направления вывода (на экран)
mov cx,13 ; указание количества символов строки
mov dx,offset msg ; поместить в регистр DX смещение строки
mov ah,40h ; выбор функции вывода строки
int 21h ; вызов прерывания DOS "Набор процедур" для вывода строки
int 20h ; вызов прерывания DOS (завершение программы)

msg DB 'Hello, World!$'
```

msg — метка (идентификатор), упрощающая доступ к данным.

Происхождение и критика термина «язык ассемблера»

Данный тип языков получил свое название от названия транслятора (компилятора) с этих языков — ассемблера (англ. assembler — сборщик). Название последнего обусловлено тем, что на древних компьютерах не существовало языков более высокого уровня, и единственной альтернативой созданию программ с помощью ассемблера было программирование непосредственно в кодах.

Язык ассемблера в русском языке часто называют «ассемблером» (а что-то связанное с ним — «ассемблерный»), что, согласно английскому переводу слова, неправильно, но вписывается в правила русского языка.

Однако, сам ассемблер (программу) тоже называют просто «ассемблером», а не «компилятором языка ассемблера» и т.п.

Использование термина «язык ассемблера» также может вызвать ошибочное мнение о существовании единого языка низкого уровня, или хотя бы стандарта на такие языки. При именовании языка, на котором написана конкретная программа, желательно уточнять, для какой архитектуры она предназначена и на каком диалекте языка написана.