

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Поволжский государственный университет телекоммуникаций и информатики»

Т.А. Коваленко, А.Г. Солодов

**«ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЯЗЫКИ
ПРОГРАММИРОВАНИЯ»**

Часть 2

Учебное пособие

Самара 2022

УДК - 004.772
ББК 3.32.97
К-56 Коваленко Т.А

Рекомендовано к изданию методическим советом ПГУТИ Протокол №48 от 12.04.2022

Рецензенты:

к.т.н. начальник отдела сбора информации ГНКЦ ЦСКБ «ПРОГРЕСС» Халилов Р.Р.

Коваленко Т.А

К-56 Вычислительная техника и языки программирования. Часть 2: учебное пособие/ Т.А. Коваленко, А.Г. Солодов – Самара: ПГУТИ, 2022. – 168 с.

Учебное пособие предназначено для студентов второго курса специальности 11.03.02 дневной и заочной формы обучения. В нем рассмотрены вопросы, относящиеся к области цифровой и вычислительной техники, достаточной для последующего изучения специальных дисциплин.

Пособие представлено в двух частях теоретической и практической. В теоретической части дается представление об основных вопросах цифровой техники. Вторая часть состоит из лабораторных и практических работ, которые направлены на получение практических навыков синтеза цифровых устройств, знания элементов архитектуры современных цифровых сигнальных процессоров и практического их использования.

Пособие позволяет рассмотреть не только теоретические вопросы, но и выполнить самостоятельно лабораторные работы и практические задания.

Согласно приказу МОН РФ «19» сентября 2017 г. № 930 и в соответствии с решением Ученого Совета ФГБОУ ВО ПГУТИ от «31» августа 2021г. (протокол № 1) учебное пособие направлено на усвоение принципов работы современных информационных технологий и использован их для решения задач профессиональной деятельности (ОПК-4).

Материал, представленный в учебном пособии, является актуальным. Он изложен доступным для студентов языком.

Учебное пособие является необходимым и полезным в учебном процессе.

ISBN

© Коваленко Т.А., Солодов А.Г., 2022

© Поволжский государственный университет телекоммуникаций и информатики, 2022

Содержание

Введение.....	6
Теоретическая часть.....	7
Лекция 1 Логические основы цифровых устройств	8
1.1. Двоичная система счисления.....	8
1.2. Понятие функции алгебры логики.....	10
1.3. Основные законы и тождества алгебры логики	14
Лекция 2 Комбинированные цифровые устройства	16
2.1. Понятие и последовательность синтеза	16
2.2. Способы задания КЦУ	19
Лекция 3 Методы минимизации ФАЛ	22
3.1 Вывод минимальной ФАЛ.....	22
3.2 Базисы и минимальные базисы	25
3.3. Построение структурной схемы.....	26
Лекция 4 Типовые КЦУ. Дешифраторы и шифраторы.....	31
4.1 Дешифратор.....	32
4.2 Шифратор	35
Лекция 5 Типовые КЦУ. Мультиплексор, демультиплексор	39
и преобразователь кода.....	39
5.1 Мультиплексор.....	39
5.2 Демультиплексор	42
5.3 Преобразователь кода.....	43
Лекция 6 Последовательные цифровые устройства.....	46
6.1. Понятие и способ задания ПЦУ	46
6.2 Понятие и классификация триггеров.....	49
6.3. Типовые триггеры.....	50
6.4 Синтез ПЦУ	56

Лекция 7 Принципы управления микропроцессора	59
7.1 Классификация микропроцессоров.....	60
7.2. Декомпозиция МП.	62
7.3 Принцип аппаратного управления ("жёсткой" логики).....	63
7.4. Принцип микропрограммного управления (гибкой логики).	64
7.5. Способы формирования сигналов управления в управляющих автоматах с "гибкой" логикой.	67
Лекция 8 Элементы архитектуры ЦСП TMS320C6х.....	69
8.1. Данные	69
8.2. Методы адресации операндов	72
Лекция 9 Структура ЦСП TMS320C6х	74
Лекция 10 Структура командной строки ассемблера, вопросы особенности адресации и команд.....	79
10.1 Основные команды процессора 'С6х для целых чисел	81
10.2 Особенности методов базирования и индексации в С6х.....	82
10.3 Команды ввода исходных данных	85
10.4 Арифметические команды	86
Лекция 11 Организация стандартных алгоритмических структур	91
Лабораторные работы	96
Лабораторная работа 1	97
Исследование логических элементов.....	97
Лабораторная работа 2.....	99
Моделирование работы комбинационных цифровых устройств.....	99
Лабораторная работа 3	102
Шифраторы и дешифраторы.....	102
Лабораторная работа 4.....	106
Мультиплексоры и демультимплексоры.....	106
Лабораторная работа 5	111
Триггеры.....	111

Лабораторная работа 6	116
Знакомство с симулятором TMS320C6201	116
Лабораторная работа 7	123
Пересылка данных	123
Лабораторная 8	130
Арифметические операции	130
Лабораторная работа 9	134
Ветвление с простым условием	134
Практические занятия	143
Практическая работа 1	143
Элементарные ФАЛ. Законы и тождества алгебры логики	143
Практическая работа 2	145
Минимизация ФАЛ методом карт Вейча-Карно и	145
минимальные базисы	145
Практическая работа 3	147
Синтез дешифратора и шифратора	147
Практическая работа 4	148
Триггеры	148
Практическая работа 5	149
Синтез двоичных счетчиков	149
Приложение	150
Система команд TMS320C6x для чисел с фиксированной запятой	150
Предметный указатель	165
Список используемой литературы	166

Введение

Целью дисциплины «Вычислительная техника и языки программирования» является обеспечение базовой подготовки студентов очного и заочного отделений специальности 11.03.02 в области цифровой и вычислительной техники, достаточной для последующего изучения специальных дисциплин.

Изучение дисциплины «Вычислительная техника и языки программирования» предполагает самостоятельную работу с литературой, прослушивание лекций, лабораторные и практические занятия. Данный курс направлен на приобретение знания основных теоретических положений цифровой техники и практических навыков синтеза цифровых устройств, знания элементов архитектуры современных цифровых сигнальных процессоров и практического их использования, а также знания современных сетевых технологий.

В пособие рассматриваются вопросы архитектуры микропроцессоров и программного обеспечения микропроцессорных систем. Оно содержит необходимые сведения об архитектуре процессоров фирмы Texas Instruments, информацию о структуре программы на ассемблере и основных директивах, а также сведения о форматах машинных команд и правилах их записи в ассемблере.

Пособие позволяет рассмотреть не только теоретические вопросы, но и выполнить самостоятельно лабораторные работы.

Использование данного учебного пособия является хорошим подспорьем для студентов технических специальностей.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части рассматриваются основные теоретические положения цифровой техники. Дается описание комбинированных, последовательных цифровых устройств. Понятия и способ их задания. Вопросы об архитектуре процессоров фирмы Texas Instruments, информация о структуре программы на ассемблере и основных директивах, а также сведения о форматах машинных команд и правилах их записи в ассемблере. Кроме этого обсуждаются некоторые приемы программирования на ассемблере. Приведены также форматы и примеры использования некоторых команд

Цель теоретической части: Дать представление в области цифровой и вычислительной техники, достаточной для последующего изучения специальных дисциплин, тем самым обеспечивая базовую подготовку студентов очного и заочного отделений специальностей 11.03.02,

Лекция 1

Логические основы цифровых устройств

В лекции рассматриваются понятия, двоичная система исчисления, переход от одной позиционной системы в другую, функции алгебры. Даются основные определения и законы, относящиеся к функции алгебры логики. Представлены элементы функции алгебры логики одного и двух аргументов.

Цель лекции: Ознакомиться с основными определениями систем исчисления, перевода из одной системы в другую, основами алгебры логики. Построение функций алгебры логики, основные элементы лежащие в их основе. Изучить законы функций алгебры логики.

1.1. Двоичная система счисления

Разряд – позиция, занимаемая отдельным символом данной системы счисления в изображении числа.

<u>n-1</u> ... 3 2 1 0	номер разряда
<u>X</u> ... X <u>X X X</u>	число
↑ ↑	
старший младший	

Разряды нумеруются справа налево, начиная с 0.

Вес разряда – количественное значение одной единицы, помещенной в данный разряд.

Численно вес разряда определяется через основание системы счисления (ОСС) и номер разряда: $\text{ОСС}^{\text{НОМЕР РАЗРЯДА}}$.

ОСС = количеству символов ее алфавита.

Примеры:

	Десятичная система					Двоичная система				
номер разряда	...	3	2	1	0	...	3	2	1	0
число	...	X	<u>X</u>	<u>X</u>	<u>X</u>	...	X	<u>X</u>	<u>X</u>	<u>X</u>
вес разряда	...	10^3	10^2	10^1	10^0	...	2^3	2^2	2^1	2^0

Разрядная сетка – количество разрядов, отведенное для представления чисел в данной цифровой системе или устройстве.

Переход от одной позиционной системы счисления к другой

1. Определение десятичного эквивалента:

$$N_{10} = x_{n-1} \cdot \text{ОСС}^{n-1} + x_{n-2} \cdot \text{ОСС}^{n-2} + \dots + x_0 \cdot \text{ОСС}^0, \text{ где}$$

x_i – символ из алфавита некоторой системы счисления, размещенный в i -ом разряде соответствующего числа.

Пример для 10010_2 : $1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 18_{10}$

2. Определение двоичного эквивалента десятичного числа:

А). Поразрядное взвешивание:

шаг 1. Определить наибольшее n , при котором $2^n \leq N_{10}$.

Найденное n = номеру старшего разряда искомого двоичного числа, в который помещается 1.

Например, для $N_{10} = 29$ $n = 4$, т.к. $2^5 = 32 > 29$.

шаг 2. Уменьшить n на 1.

шаг 3. Если $n = 0$ искомое число найдено. В противном случае продолжить.

шаг 4. Если $2^{n-1} \leq N_{10} - 2^n$, в разряд $(n-1)$ поместить 1. В противном случае 0.

В нашем примере $2^3 < 29 - 2^4$. Следовательно, в 3-й разряд также помещаем 1.

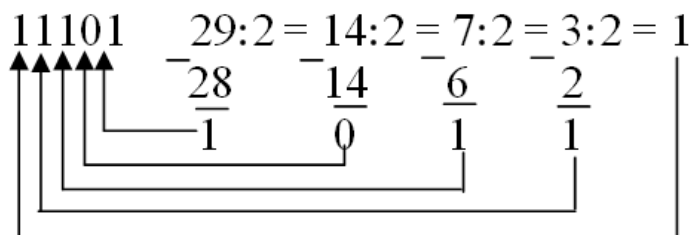
шаг 5. Вернуться к шагу 2.

Следуя приведенному алгоритму, $29_{10} = 11101_2$.

Данный алгоритм удобен для двоичной системы счисления. В случае произвольной системы счисления предпочтительным является следующий алгоритм.

Б). Последовательное деление на ОСС, пока оно больше очередного частного.

Например, для ОСС = 2:



1.2. Понятие функции алгебры логики

Функция, однозначно определяющая соответствие каждой из всех возможных совокупностей значений аргументов нулю (ложь) или единице (истина), называется функцией алгебры логики (ФАЛ).

Логические переменные – как функция, так и аргументы, могут принимать только два значения. Обычно их обозначают символами 0 и 1, что в вычислительной технике соответствует низкому и высокому уровню напряжения сигнала соответственно.

Для технической реализации ФАЛ используют электрические схемы, называемые логическими элементами. Логический элемент (ЛЭ) выполняет логические операции над одной или более логическими переменными. При этом переменная-функция соответствует выходу ЛЭ, а переменные-аргументы – разрядам двоичных наборов, поступающим на соответствующие входы ЛЭ. Фактически по внутреннему устройству любой ЛЭ это набор транзисторов соединенных по соответствующему принципу.

Закон функционирования ЛЭ представляют в виде таблицы истинности (Таблица 1.1).

Таблица 1.1

Закон функционирования ЛЭ

Номер двоичного набора	Аргументы			Функция
	x_{n-1}	...	x_0	
0	0	...	0	
...
$2^n - 1$	1	...	1	

В строках первого раздела этой таблицы по порядку записываются десятичные номера входных двоичных наборов. В строках второго раздела записываются собственно входные наборы, представляющие собой n -разрядное двоичное отображение соответствующего десятичного номера. В строках третьего раздела записываются соответствующие входным наборам значения функции.

В общем случае ФАЛ строятся на основе элементарных ФАЛ. Элементарной называется ФАЛ одного или двух аргументов, в логическом выражении которой содержится не более одной логической операции.

К элементарным ФАЛ одного аргумента относятся:

1. Константа нуля. Реализуется генератором нуля, который на схемах обозначается соединением соответствующего входа ЛЭ с «землей» (общим проводом источника питания).

2. Константа единицы. Реализуется генератором единицы, который на схемах обозначается соединением соответствующего входа ЛЭ с полюсом источника питания.

3. Повторение. Реализуется логическим элементом повторителем. Его условное графическое обозначение и таблица истинности показаны на рис.1.1. Записывается ФАЛ как $y = x$. Зачастую возникает вопрос, зачем нужен данный логический элемент? Повторитель: представляет собой просто усилительный каскад. В логическом смысле усиление является аналоговой функцией, а не цифровой, но буферы часто требуются и в цифровых схемах. Например, биты адреса при выходе из процессора

оказываются довольно слабыми по нагрузочной способности, и их нельзя подавать в шину адреса без усиления.

4. Инверсия или логическое отрицание. Реализуется логическим элементом НЕ (инвертором). Его условное графическое обозначение и таблица истинности показаны на рис. 1.2. Записывается ФАЛ как $y = \bar{x}$.

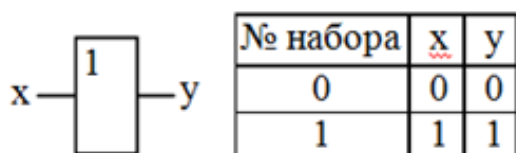


Рис.1.1 – Повторитель

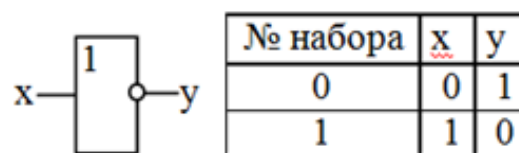


Рис.1.2 – Инвертор

Основными из элементарных ФАЛ двух аргументов являются:

1. Дизъюнкция (логическое сложение). Реализуется логическим элементом ИЛИ. Его условное графическое обозначение и таблица истинности показаны на рис.1. 3. Записывается ФАЛ как $y = a \vee b$.

2. Конъюнкция (логическое умножение). Реализуется логическим элементом И. Его условное графическое обозначение и таблица истинности показаны на рис.1.4. Записывается ФАЛ как $y = a \wedge b$.

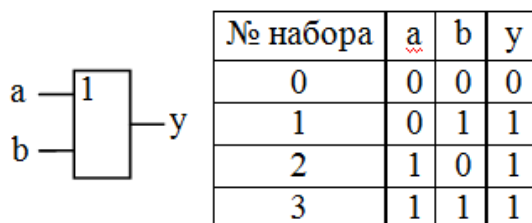


Рис. 1.3 – ИЛИ

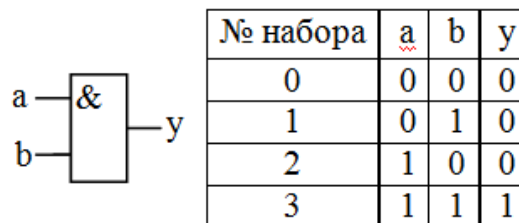


Рис.1.4 – И

3. Стрелка Пирса. Реализуется логическим элементом ИЛИ-НЕ. Его условное графическое обозначение и таблица истинности показаны на рис.1.5. Записывается ФАЛ как $y = a \downarrow b$ и может быть представлена в развернутой форме: $y = \overline{a \vee b}$.

4. Штрих Шеффера. Реализуется логическим элементом И-НЕ. Его условное графическое обозначение и таблица истинности показаны на

рис.1.6. Записывается ФАЛ как $y = a \mid b$ и может быть представлена в развернутой форме: $y = \overline{a \wedge b}$.

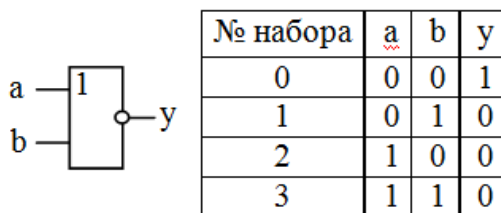


Рис.1.5 – ИЛИ-НЕ

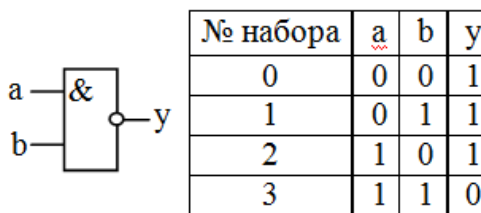


Рис.1.6 – И-НЕ

5. Исключающее ИЛИ (сложение по модулю два, XOR - eXception OR). Реализуется сумматором по модулю два. Его условное графическое обозначение и таблица истинности показаны на рис. 1.7. Записывается ФАЛ как $y = a \oplus b$ и может быть представлена в развернутой форме: $y = \overline{(a \wedge b)} \vee (a \wedge \overline{b})$.

6. Эквивалентность (равнозначность). Реализуется одноименным ЛЭ. Его условное графическое обозначение и таблица истинности показаны на рис.1.8. Записывается ФАЛ как $y = a \sim b$ и может быть представлена в развернутой форме: $y = \overline{a \oplus b} = \overline{(a \wedge \overline{b})} \vee (a \wedge b)$.

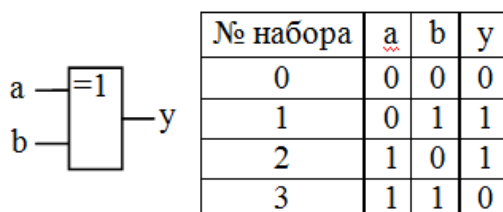


Рис.1.7 – Исключающее ИЛИ

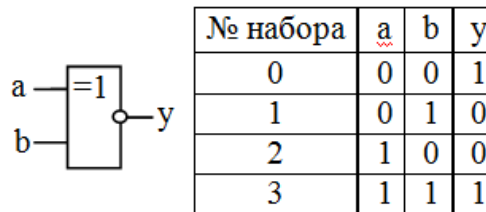


Рис. 1.8 – Эквивалентность

Функции дизъюнкция и штрих Шеффера с одной стороны, конъюнкция и стрелка Пирса с другой, являются частными случаями функций константы (постоянной) нуля и единицы, соответственно.

Функция константы единицы (нуля) от n аргументов обращается в единицу (ноль) лишь на каком-либо одном наборе аргументов и обращается в ноль (единицу) на остальных наборах.

Все рассмотренные функции могут быть расширены на произвольное число аргументов.

1.3. Основные законы и тождества алгебры логики

Тождества:

$$\begin{array}{llll} x \vee x = x, & x \vee \bar{x} = 1, & x \vee 1 = 1, & x \vee 0 = x, \\ x \wedge x = x, & x \wedge \bar{x} = 0, & x \wedge 1 = x, & x \wedge 0 = 0, \\ x \oplus x = 0, & x \oplus \bar{x} = 1, & x \oplus 1 = \bar{x}, & x \oplus 0 = x. \end{array}$$

Законы:

– двойной инверсии – $\bar{\bar{a}} = a$;

– сочетательный – $a \wedge (b \wedge c) = (a \wedge b) \wedge c$, $a \vee (b \vee c) = (a \vee b) \vee c$,

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c;$$

– переместительный – $a \wedge b = b \wedge a$, $a \vee b = b \vee a$, $a \oplus b = b \oplus a$;

– распределительный – $a \wedge (b \vee c) = a \wedge b \vee a \wedge c$,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c),$$

$$a \wedge (b \oplus c) = a \wedge b \oplus a \wedge c;$$

– двойственности (правила де Моргана) – $\overline{a \vee b} = \bar{a} \wedge \bar{b}$, $\overline{a \wedge b} = \bar{a} \vee \bar{b}$;

– поглощения – $a \vee a \wedge c = a$, $a \wedge (a \vee c) = a$;

– склеивания – $a \wedge \bar{c} \vee a \wedge c = a$, $(a \vee \bar{c}) \wedge (a \vee c) = a$.

Все указанные законы и тождества справедливы для любого числа аргументов, причем аргументом может быть как простая переменная, так и функция.

Законы и тождества алгебры логики используются для преобразования ФАЛ в процессе синтеза цифровых устройств (ЦУ). В частности, целью преобразования может быть упрощение ФАЛ:

$$y = \bar{a} \cdot (b \vee \bar{c}) \vee b \wedge \bar{c} = a \vee (\bar{b} \vee \bar{c}) \vee b \wedge \bar{c} = a \vee b \wedge \bar{c} \vee b \wedge \bar{c} = a \vee b \oplus c$$

Кроме того, в тождествах отражены правила замены логических элементов одного типа логическими элементами другого типа.

Минимизация ФАЛ при построении ЦУ на аппаратном уровне приводит к уменьшению количества используемых элементов, а при программной

реализации к уменьшению команд, а соответственно уменьшению энергопотребления и увеличению быстродействия.

Краткие итоги

Рассмотрены понятия, двоичной системы исчисления и перевода из одной системы в другую. Дано определение функции алгебры логики. Основные элементы ФАЛ: дизъюнкция, конъюнкция, стрелка Пирса, штрих Шеффера, исключающее ИЛИ и эквивалентность. Изучены основные законы тождества алгебры и логики.

Вопросы для самопроверки

1. Дайте определение ФАЛ.
2. Сколько значений могут принимать логические переменные?
3. Перечислите логические операции, относящиеся к элементарным ФАЛ одного аргумента.
4. Перечислите логические операции, относящиеся к элементарным ФАЛ двух аргументов.
5. Дайте определение конъюнкции.
6. Дайте определение дизъюнкции.

Лекция 2

Комбинированные цифровые устройства

В лекции рассматриваются цифровые устройства. Отражение двоичных наборов физических значений разрядов. Дается терминология, которая применяется в комбинированных цифровых устройствах (КЦУ). Описывается последовательность синтеза, способы задания КЦУ.

Цель лекции: Ознакомиться с основными определениями, применяемыми в КЦУ. Уяснить когда КЦУ называется полностью или не полностью (частично) определённым. Усвоить понятия последовательности синтеза и способы задания КЦУ.

2.1. Понятие и последовательность синтеза

Любое цифровое устройство можно рассматривать как преобразователь входных n -разрядных двоичных наборов в выходные m -разрядные двоичные наборы. Поскольку n определяется числом информационных входов цифрового устройства, длина (число разрядов) входных двоичных наборов постоянна и не зависит от их значения. Тогда общее число входных двоичных наборов не превышает 2^n и, следовательно, диапазон десятичных чисел, которые могут быть обработаны таким цифровым устройством составляет от 0 до $M_{\max} = 2^n - 1$. Отсюда легко решается обратная задача: число разрядов, необходимое для представления в цифровом устройстве абсолютных значений десятичных чисел от M_{\min} до M_{\max} определяется из соотношения: $n = \lceil \log_2(M_{\max} + 1) \rceil$, где $\lceil x \rceil$ – наименьшее целое, не меньшее x (округление в большую сторону).

Физически значения разрядов (логических переменных) двоичных наборов отображаются электрическими сигналами в одной из двух форм – потенциальной или импульсной.

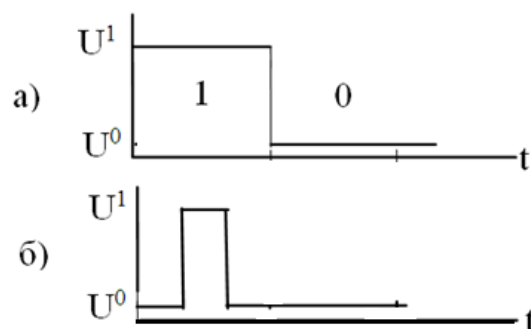


Рис. 2.1 – Физическое представление бит информации

В потенциальной форме (рис. 2.1,а) единичному значению двоичного разряда соответствует высокий уровень напряжения U^1 , а нулевому – низкий уровень U^0 , близкий к «земле». В импульсной форме (рис. 2.1,б) единичному значению двоичного разряда соответствует относительно короткий импульс, а нулевому – отсутствие импульса.

Как видно, в любой форме сигналы могут принимать только два значения и потому называются двоичными сигналами. В этой терминологии и определим КЦУ: комбинационным называется цифровое устройство, у которого выходные двоичные сигналы в любой момент времени зависят только от тех двоичных сигналов, которые поступают на вход устройства в тот же момент времени. Таким образом, сигналы на выходе КЦУ изменяются практически сразу после изменения входных сигналов.

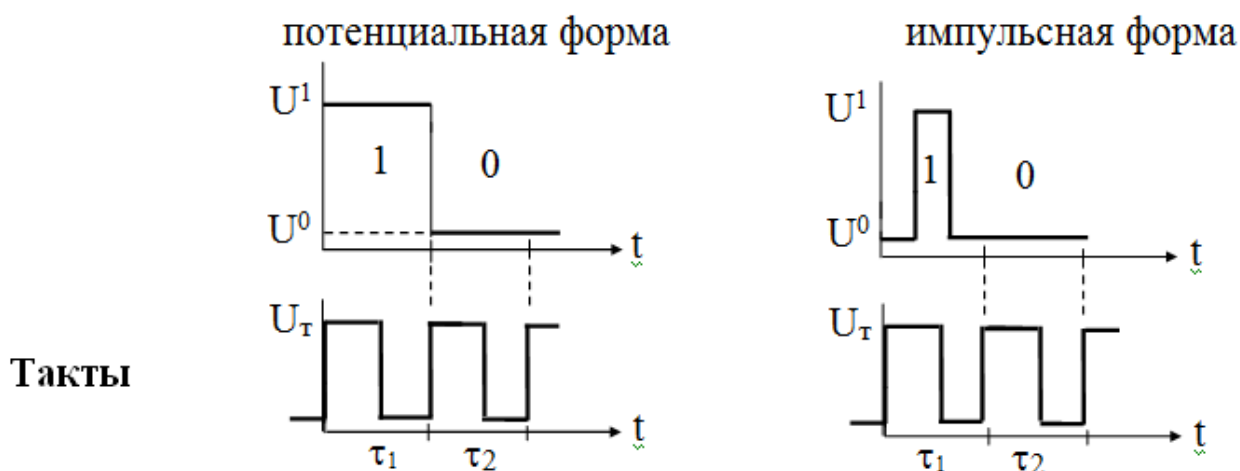


Рис. 2.2 – Отображение двоичных сигналов

В реальных цифровых устройствах каждые входной и соответствующий ему выходной двоичные наборы отображаются двоичными сигналами лишь в течение определенного интервала времени, называемого тактом (рис. 2.2). При этом говорят, что устройство тактируется, то есть входные и выходные сигналы могут изменяться лишь с началом (окончанием) каждого последующего такта. Сами такты задаются генератором тактовой частоты.

КЦУ называется полностью определённым, если каждому из всех его возможных входных двоичных наборов поставлен в соответствие строго определённый двоичный набор на выходе. Если хотя бы для одного входного набора значение одного или более разрядов соответствующего выходного набора безразлично, КЦУ называется не полностью или частично определённым. На практике частично определённые КЦУ соответствуют ситуациям, когда некоторые двоичные наборы либо никогда не появляются на входе, либо никогда не оказываются востребованными на выходе.

Синтез любого КЦУ проводится в следующей последовательности:

1. Задается закон функционирования.
2. Для каждого из m выходов выводится минимальная ФАЛ, то есть ФАЛ с минимальным числом членов и минимальным числом аргументов в каждом члене.
3. При необходимости каждая минимальная ФАЛ записывается в заданном минимальном базисе.
4. В соответствии с системой минимальных ФАЛ строится структурная схема устройства.

2.2. Способы задания КЦУ

Табличный.

Правила работы задаются таблицей истинности. При этом в случае частично определенного КЦУ строки раздела «Выходной набор» таблицы, соответствующие безразличным входным наборам, заполняются символом тильда (рис. 2.3). Однако при большом значении n таблица истинности становится громоздкой и теряет наглядность.

№ набора	Входной набор			Выходной набор		
	x_{n-1}	...	x_0	y_{m-1}	...	y_0
0	0	...	0	~	...	~
...		
2^n-1	1	...	1	1	...	0

Рис. 2.3 – Пример таблицы истинности при безразличном нулевом наборе

Скобочная запись таблицы истинности.

В случае полностью определенного КЦУ для каждого разряда выходного набора в круглых (квадратных) скобках через запятую перечисляются десятичные номера входных наборов, на которых значение этого разряда обращается в 0 (1).

Полностью определенное Частично определенное

№ вх. набора	x_2	x_1	x_0	y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

№ вх. набора	x_2	x_1	x_0	y
0	0	0	0	~
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	~
6	1	1	0	0
7	1	1	1	1

Рис. 2.4 – Скобочная запись таблицы истинности

На рис. 2.4 показана, запись $y_i(n = 3) = [0, 1, 4, 7]$ означает, что i -й

разряд выходных наборов имеет значение 1 на 0, 1, 4 и 7 n-разрядных входных наборах. Следовательно, на остальных ($2^3 - 4 = 4$) не указанных наборах, этот разряд имеет значение 0.

В случае частично определенного КЦУ используются оба вида скобок, что позволяет не указывать безразличные входные наборы.

На рис. 2.4, запись $y_i(n = 3) = [1, 4, 7, (2, 3, 6)]$ означает, что i-й разряд выходных наборов принимает значение 1 на первом, четвертом и седьмом входных наборах, значение 0 – на втором, третьем и шестом, и его значение безразлично на остальных ($2^3 - 6 = 2$) не указанных входных наборах.

Аналитический.

Правила работы КЦУ задаются системой ФАЛ, каждая из которых соответствует определенному выходу КЦУ и записывается на основании таблицы истинности или ее скобочного представления в любой из двух совершенных форм – дизъюнктивной (СДНФ) или конъюнктивной (СКНФ).

Правила записи ФАЛ в совершенной форме:

–ФАЛ в СДНФ (СКНФ) представляется дизъюнкцией (конъюнкцией) своих членов, каждый из которых соответствует единственному и строго определенному входному набору, обращающего данную функцию в 1 (0) или соответствующего безразличному ее значению;

–каждый член функции в СДНФ (СКНФ) образуется конъюнкцией (дизъюнкцией) всех аргументов, которые берутся с инверсией при нулевом (единичном) значении в данном входном наборе и без инверсии – при единичном (нулевом). Таким образом, каждый член функции в СДНФ (СКНФ) является функцией конституанты 1 (0) (таб.2.1).

Таблица 2.1

Примеры записи ФАЛ в совершенной форме

№ вх. набора	x ₂	x ₁	x ₀	y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

№ вх. набора	x ₂	x ₁	x ₀	y
0	0	0	0	~
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	~
6	1	1	0	0
7	1	1	1	1

$$y_{\text{сднф}} = \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} \vee \overline{x_2} \cdot \overline{x_1} \cdot x_0 \vee x_2 \cdot \overline{x_1} \cdot \overline{x_0} \dots \quad y_{\text{скнф}} = (x_2 \vee x_1 \vee x_0) \wedge (\overline{x_2} \vee \overline{x_1} \vee \overline{x_0})..$$

набор 0 набор 1 набор 4набор 0набор 2

Краткие итоги

Рассмотрены понятия КЦУ. Даны понятие, и последовательность синтеза. Изучены как двоичные наборы отображаются двоичными сигналами и способы задания КЦУ.

Вопросы для самопроверки

1. Дайте определение цифрового устройства.
2. Дайте определение КЦУ.
3. Что означает отображение электрическими сигналами двоичных наборов в потенциальной форме?
4. Что означает отображение электрическими сигналами двоичных наборов в импульсной форме?
5. Какова последовательность синтеза любого КЦУ?
6. Перечислите способы задания КЦУ.
7. Опишите табличный способ задания КЦУ.
8. Опишите аналитический способ задания КЦУ.
9. В чем заключается скобочная запись таблицы истинности для задания КЦУ?
10. Перечислите правила записи ФАЛ в совершенной форме.

Лекция 3

Методы минимизации ФАЛ

В лекции рассматриваются выводы минимальной ФАЛ методом графическим с помощью карт Вейча-Карно, и алгебраическим – метод Квайна. Понятие минимального базиса как минимальный набор ФАЛ, позволяющий представить любую функцию от произвольного числа аргументов.

Цель лекции: Ознакомиться с основными методами минимизации ФАЛ. Уяснить чем отличается графический от алгебраического метода. Усвоить понятия минимального базиса. Рассмотреть построение схемы цифрового устройства(ЦУ)

3.1 Вывод минимальной ФАЛ

Существуют графические и алгебраические методы минимизации ФАЛ. Из графических наибольшее распространение получил метод карт Вейча-Карно, а из алгебраических – метод Квайна.

Карта Вейча-Карно представляет собой таблицу истинности специальной формы с числом клеток 2^n (рис. 3.1), где n – длина входных наборов КЦУ.

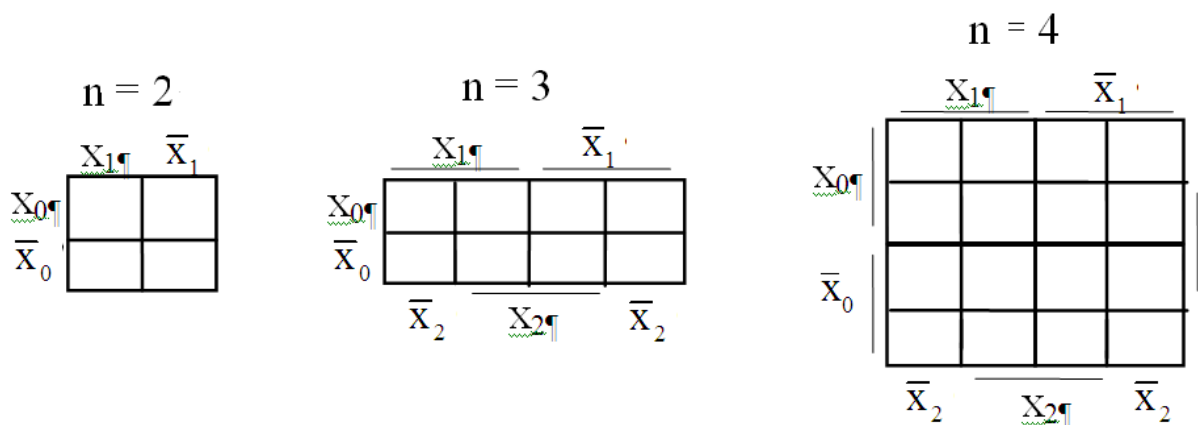


Рис 3.1 – Формат карт Вейча-Карно

Минимизация методом карт Вейча-Карно проводится в следующей

последовательности.

1. Каждая клетка карты соответствует определенному члену исходной ФАЛ. Местоположение клетки определяется пересечением строк и столбцов карты, одноименных с аргументами данного члена ФАЛ. Найденная клетка отмечается 1 в случае СДНФ, 0 в случае СКНФ и тильдой, если данный член ФАЛ соответствует безразличному значению функции.

2. Отмеченные и только отмеченные клетки объединяются в замкнутые области. При этом:

- каждая область должна представлять собой прямоугольник с числом клеток 2^k , где $k = 0, 1, 2, \dots$. Следовательно, во-первых, область может содержать одну клетку ($k = 0$), две ($k = 1$), четыре ($k = 2$), восемь ($k = 3$) и т.д., но не 3, 5, 6, 7 и т.д. клеток. Во-вторых, нельзя объединять клетки, расположенные по диагонали;

- число клеток в области должно быть максимально возможным, поскольку только тогда число аргументов в соответствующем члене выводимой функции будет минимальным;

- одни и те же клетки могут входить в разные области, т.е. области могут пересекаться;

- при условии участия всех отмеченных клеток в процедуре формирования областей следует стремиться к минимальному числу областей, поскольку только тогда число членов выводимой функции будет минимальным. С этой целью допускается сворачивание карты в цилиндр относительно горизонтальной или вертикальной осей с соединением противоположных граней, также возможно объединение 4-х углов.

Учет безразличных входных наборов (клеток, помеченных тильдой) повышает эффективность минимизации. Однако если эти клетки не способствуют расширению областей, их учитывать не следует.

3. В соответствии с выделенными областями записывается

минимальная ФАЛ в дизъюнктивной (если исходной была СДНФ) или конъюнктивной (если исходной была СКНФ) нормальной форме. При этом каждая область определяет отдельный член минимальной функции, который составляется лишь из тех аргументов, которые в данную область входят либо только с инверсией, либо только без инверсии (Рис. 3.2).



Рис. 3.2 – Вывод минимальной ФАЛ методом карт Вейча-Кано

Минимизация методом Квайна заключается в следующем.

1. Члены исходной ФАЛ, где учтены и безразличные входные наборы, отличающиеся только в одной переменной, группируются в пары.

Две ФАЛ отличаются в одной переменной, если эта переменная в одну из них входит с инверсией, а в другую без инверсии. Других различий нет.

При группировке руководствуются следующими правилами:

– пары могут пересекаться, т.е. один и тот же член функции может входить в различные пары;

– при условии участия в процедуре группирования всех членов ФАЛ, число пар должно быть минимальным.

2. К сформированным парам в случае СДНФ применяется операция склеивания, а в случае СКНФ – операция поглощения.

3. По результатам преобразования записывается новая ФАЛ, к которой применяются законы и тождества алгебры логики или повторно все пункты минимизации.

Пример для ФАЛ $y_{/n=3} = [0, 1, 7, (2, 6)]$.

Шаг 1. Запись ФАЛ в любой из совершенных форм.

Шаг 2. Группирование:

$$y = \overbrace{\bar{x}_2 \bar{x}_1 \bar{x}_0 \vee \bar{x}_2 \bar{x}_1 x_0}^{1 \text{ пара}} \vee \overbrace{x_2 x_1 x_0 \vee \bar{x}_2 x_1 x_0}^{2 \text{ пара}} \vee \overbrace{x_2 \bar{x}_1 \bar{x}_0 \vee x_2 \bar{x}_1 x_0}^{3 \text{ пара}}$$

Шаг 3. Применить закон склеивания (в случае СДНФ) или поглощения (в случае СКНФ):

$$1 \text{ пара: } \bar{x}_2 \bar{x}_1 \bar{x}_0 \vee \bar{x}_2 \bar{x}_1 x_0 = \bar{x}_2 \bar{x}_1$$

$$2 \text{ пара: } x_2 x_1 x_0 \vee \bar{x}_2 x_1 x_0 = x_1 x_0$$

$$3 \text{ пара: } x_2 \bar{x}_1 \bar{x}_0 \vee x_2 \bar{x}_1 x_0 = x_2 \bar{x}_1$$

Шаг 4. Записать новую ФАЛ и при возможности группирования возврат к шагу 2:

$$(шаг 2) \quad y = \overbrace{\bar{x}_2 \bar{x}_1 \vee x_1 x_0}^{1 \text{ пара}} \vee [x_2 \bar{x}_1];$$

$$(шаг 3) \quad 1 \text{ пара: } \bar{x}_2 \bar{x}_1 \vee x_2 \bar{x}_1 = \bar{x}_1;$$

$$(шаг 4) \text{ новая ФАЛ: } y = \bar{x}_1 \vee x_1 x_0.$$

В противном случае продолжить.

Шаг 5. Если возможно, применить законы и тождества алгебры логики:

$$y = \overline{\bar{x}_1 \vee x_1 x_0} = \overline{x_1 x_1 x_0} = \overline{x_1 (\bar{x}_1 \vee x_0)} = \overline{x_1 \bar{x}_0} = \bar{x}_1 \vee x_0.$$

Шаг 6. Объявить конечный результат.

При этом полезным бывает следующее соотношение $\bar{x} \cdot z \vee x = x \cdot z$.

3.2 Базисы и минимальные базисы

Полным базисом называется система ФАЛ, позволяющая представить любую функцию от произвольного числа аргументов. Полный

базис допускает использование логических элементов самых различных типов: И, ИЛИ, И-НЕ, ИЛИ-НЕ, НЕ и т.д.

Минимальным базисом называется минимальный набор ФАЛ, позволяющий представить любую функцию от произвольного числа аргументов. Минимальный базис допускает использование логических элементов только одного строго определенного типа. Из минимальных базисов наибольшее практическое применение получили базисы И-НЕ и ИЛИ-НЕ.

ФАЛ в базисе ИЛИ-НЕ может содержать только операции стрелка Пирса, а в базисе И-НЕ – только операции штрих Шеффера. Вместе с тем допустимо использование операции инверсии, а, следовательно, в технической реализации – логического элемента НЕ.

Для представления минимальной ФАЛ в любом из минимальных базисов используются законы двойной инверсии и двойственности. Например, ФАЛ $y = x_2 \cdot x_3 \vee x_1 \cdot x_0$ запишется в базисе И-НЕ следующим образом: $y = \overline{\overline{x_2 \cdot x_3 \vee x_1 \cdot x_0}} = \overline{\overline{x_2 \cdot x_3} \wedge \overline{x_1 \cdot x_0}}$.

3.3. Построение структурной схемы

Структурная схема ЦУ представляет собой совокупность условных графических изображений логических элементов, типовых цифровых устройств и связей между ними.

Элементной базой при технической реализации цифровых устройств являются интегральные схемы различной степени интеграции.

На структурных схемах компоненты микросхем нумеруются слева направо и сверху вниз, то есть построчно. При этом используются две латинские буквы DD, за которыми следуют номер микросхемы в устройстве и через точку номер компоненты в микросхеме. Если микросхема содержит один компонент (логический элемент или типовое устройство), то используется одна буква D и номер компоненты в

нумерации отсутствует.

Последовательность построения структурной схемы устройства, реализующего данную ФАЛ или систему ФАЛ, определяется приоритетностью логических операций: старшей является инверсия, затем следует конъюнкция и, наконец, операции типа дизъюнкции – дизъюнкция, исключающее ИЛИ и эквивалентность.

При разработке структурной схемы необходимо учитывать такой параметр микросхем, как коэффициент разветвления по выходу $K_{\text{раз}}$, определяющий допустимое число входов интегральных компонент, одновременно подключаемых к выходу компоненты данной микросхемы при сохранении ее работоспособности в заданных условиях эксплуатации. Значение $K_{\text{раз}}$ определяется типом выходного каскада компонент микросхемы. Помимо этого тип выходного каскада определяет способ организации мультиплексной линии.

Мультиплексирование – поочередное подключение к линии передачи на время T выхода каждого из N логических элементов или цифровых устройств.

Выходные каскады выполняются в одном из трех основных вариантов.

1. Обычный каскад (рис. 3.3, а). $K_{\text{раз}} = 10$.

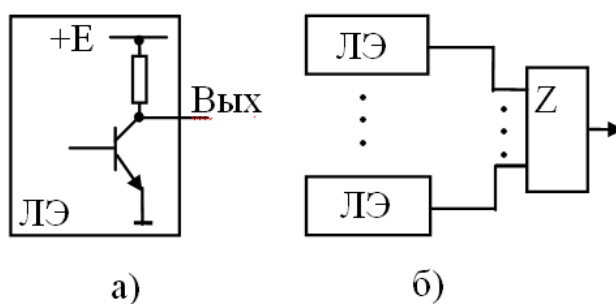


Рис. 3.3 – Обычный выходной каскад

Выход имеет два устойчивых состояния – 0 (транзистор открыт) и 1 (транзистор закрыт).

При организации мультиплексной линии (рис. 3.2,б) требуется дополнительный ЛЭ Z, тип которого (И либо ИЛИ) зависит от значения

выходного сигнала логических элементов в состоянии закрытого ключа.

2. Каскад с открытым коллектором (рис. 3.4,а). $K_{\text{раз.}} = 16$.

Для обеспечения устойчивого состояния 1 выход через внешний резистор подключается к источнику питания.

Повышенное значение $K_{\text{раз}}$ объясняется возможностью регулирования выходного тока путем изменения сопротивления резистора.

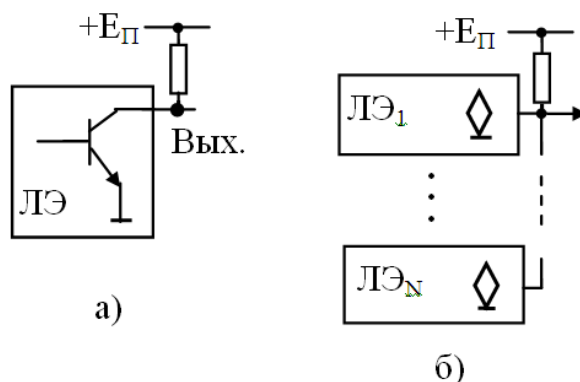


Рис 3.4 – Каскад с открытым коллектором

Исполнение выходных каскадов с открытым коллектором отмечается на корпусе микросхемы символом в виде подчеркнутого ромба (рис. 3.4,б).

Мультиплексная линия образуется соединением выходов всех N логических элементов на одном общем внешнем резисторе (рис. 3.4,б). Такое включение эквивалентно использованию дополнительного логического элемента ИЛИ, поэтому его называют "монтажным (проводным) ИЛИ".

3. Каскад с тремя состояниями (рис. 3.5,а). $K_{\text{раз.}} = 10$.

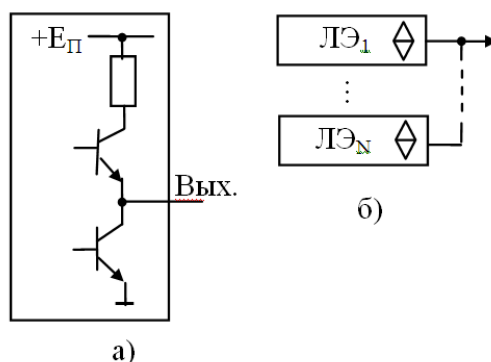


Рис 3.5 – Каскад с тремя состояниями

Здесь возможны три ситуации:

–напряжение логического нуля на выходе соответствует открытому нижнему транзистору;

–напряжение логической единицы на выходе соответствует закрытому нижнему и открытому верхнему транзистору;

–при обоих закрытых транзисторах выход отключен от цепей питания. Это и есть третье (безразличное) состояние.

Исполнение выходных каскадов с тремя состояниями отмечается на корпусе микросхемы символом в виде ромба с поперечной линией (рис. 3.5,б).

Мультиплексная линия образуется «монтажным ИЛИ» без дополнительного резистора (рис. 3.5,б).

Порядок построения схемы:

1. Реализуется инверсия отдельной переменной, а затем группы переменных (штрих Шеффера, стрелка Пирса) в порядке возрастания их численности.
2. Реализуется операция конъюнкции.
3. Реализуются операции типа дизъюнкции (дизъюнкция, сумма по модулю два, эквивалентность) (рис. 3.6).

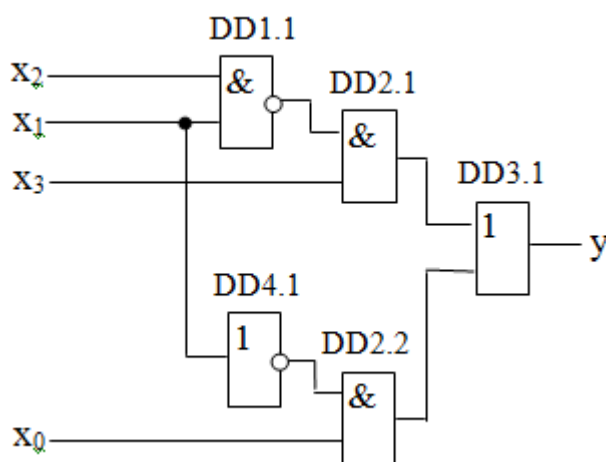


Рис. 3.6 – Пример реализации ФАЛ

Краткие итоги

Изучены основные методы минимизации ФАЛ. Рассмотрены понятия минимального базиса, а так же построение схемы ЦУ. Даны понятия интегральной схемы, мультиплексированию. Изучены три вида выходных каскадов: обычный, с открытым коллектором и тремя состояниями.

Вопросы для самопроверки

1. Опишите метод минимизации ФАЛ с помощью Карты Вейча-Карно.
2. Опишите метод минимизации Квайна.
3. Что называется полным базисом?
4. Что называется минимальным базисом?
5. Что такое структурная схема ЦУ?
6. Дайте определение интегральной схемы.
7. Дайте определение понятию мультиплексирование.
8. Опишите обычный каскад.
9. Опишите каскад с открытым коллектором.
10. Опишите каскад с тремя состояниями.

Лекция 4

Типовые КЦУ. Дешифраторы и шифраторы.

В лекции рассматриваются типовые КЦУ дешифраторы и шифраторы. Их принцип работы и построения типовых КЦУ.

Цель лекции: Ознакомиться с основными принципами работы типовых КЦУ дешифраторов, шифраторов. Рассмотреть синтез дешифратора и шифратора. Структуру построения дешифратора.

На входы типовых КЦУ могут подаваться два вида сигналов – информационные сигналы и сигналы управления. Информационные сигналы отображают обрабатываемую информацию, а сигналы управления выполняют одну или несколько из следующих функций:

- переключение входных информационных сигналов на определенные выходы устройства;
- задание порядка формирования выходных сигналов;
- синхронизация работы устройства. Синхросигнал задает временной интервал между любыми двумя соседними моментами срабатывания устройства и отображается тактовыми импульсами (тактами);
- стробирование устройства. Сигнал стробирования задает временной интервал, в течение которого выходы устройства разблокированы и на них передается результат обработки информационных сигналов.

Разрешающее значение управляющего сигнала называется активным. На условном графическом обозначении типовых КЦУ всегда указывается именно активный управляющий сигнал: нулевой знаком инверсии \neg |, единичный – его отсутствием | |. Аналогично для некоторых КЦУ указываются и активные значения выходных сигналов.

К основным из типовых КЦУ относятся дешифраторы, шифраторы, мультиплексоры, демультиплексоры и преобразователи кода.

4.1 Дешифратор

Дешифратором называется КЦУ, преобразующее n -разрядное двоичное число на входе в активный сигнал на одном из m выходов.

Основное назначение дешифраторов – формирование управляющих сигналов.

Дешифратор с n входами и $m = 2^n$ выходами называется полным, а в случае $m < 2^n$ – неполным (частично определенное КЦУ).

На рис. 4.1 приведено условное графическое обозначение дешифраторов на примере полного дешифратора 2×4 ($n = 2$, $m = 4$) со стробированием (вход V) и инверсными выходами (активным нулевым выходным сигналом). Информационные входы обозначаются весовыми коэффициентами двоичных разрядов, что устанавливает однозначное соответствие между номером входа дешифратора и номером разряда двоичного набора. Для выходов используется сквозная нумерация.

Таблица истинности полных дешифраторов (Рис. 4.1 (б), ограниченная дешифратором Рис. 4.1 (а)).

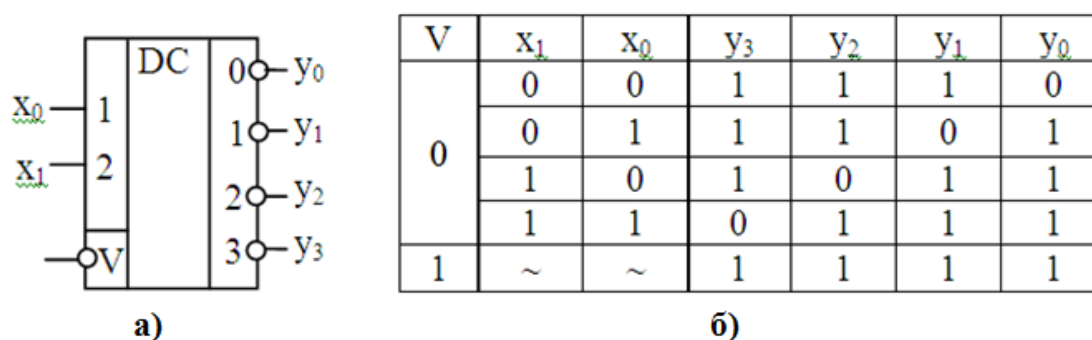


Рис. 4.1 – Дешифратор и его таблица истинности

Из таблицы видно, что номер выхода, на котором появляется активный сигнал, является десятичным эквивалентом текущего двоичного набора на информационных входах. В маркировке микросхем

дешифраторов используются буквы ИД.

На базе любого дешифратора $n \times m$ можно строить дешифратор большего размера, путем каскадной схемы соединения.

Пример: дешифратор 4×16 на базе дешифратора 2×4 (рис.4.2).

Количество микросхем первой ступени: $K_1 = \lceil N/M \rceil$ (в примере $K_1 = 16/4 = 4$). Их информационные входы соединены параллельно и являются информационными входами младших разрядов искомого дешифратора.

По правилу работы дешифратора разрешенной должна быть только одна из K_1 микросхем. Этот принцип реализуется с помощью второй ступени дешифраторов, управляющих дешифраторами первой ступени по входам стробирования. Число микросхем этой ступени: $K_2 = \lceil K_1/M \rceil$ (в примере $K_2 = 4/4 = 1$). Их информационные входы также соединяются параллельно и являются информационными входами следующих разрядов искомого дешифратора.

Аналогично организуются третья и т.д. ступени, каждая из которых управляет предыдущей ступенью.

Наконец, информационные входы дешифратора последней ступени являются информационными входами старших разрядов искомого дешифратора.

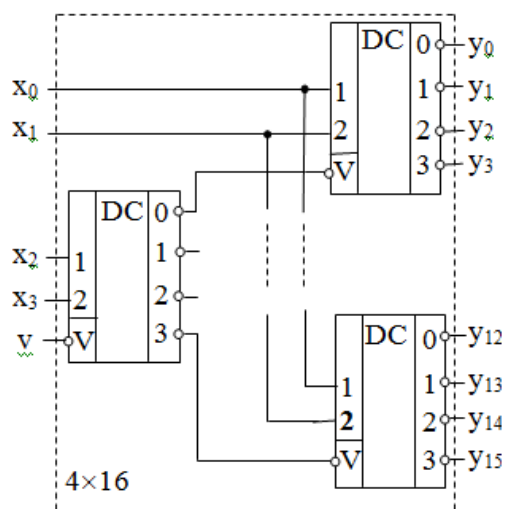


Рис. 4.2 – Дешифратор 4×16 на базе микросхем дешифратора 2×4

Таким образом, информационные выходы дешифратора второй ступени используются в качестве управляющих для дешифраторов первой ступени.

Синтез полного дешифратора

Пример: синтез дешифратора 2×4 с инверсными выходами в полном базисе (рис.4.3).

v	x ₁	x ₀	y ₃	y ₂	y ₁	y ₀
0	0	0	1	1	1	0
	0	1	1	1	0	1
	1	0	1	0	1	1
	1	1	0	1	1	1
1	~	~	1	1	1	1

$$y_0 = v \vee x_1 \vee x_0$$

$$y_1 = v \vee x_1 \vee \bar{x}_0$$

$$y_2 = v \vee \bar{x}_1 \vee x_0$$

$$y_3 = v \vee \bar{x}_1 \vee \bar{x}_0$$

Рис.4.3 – Синтез дешифратора 2×4 с инверсными выходами

Синтез неполного дешифратора

Пример: синтез дешифратора 3×5 с прямыми выходами в полном базисе (рис.4.4)

	<u>x₁</u>	<u>\bar{x}_1</u>		<u>x₁</u>	<u>\bar{x}_1</u>			
<u>x₀</u>			~					
<u>\bar{x}_0</u>	~		~	1				
	<u>\bar{x}_2</u>	<u>\bar{x}_2</u>	<u>\bar{x}_2</u>					
	$y_0 = \bar{x}_0 \bar{x}_1 \bar{v}$							
	<u>x₁</u>	<u>\bar{x}_1</u>		<u>x₁</u>	<u>\bar{x}_1</u>			
<u>x₀</u>			~	1				
<u>\bar{x}_0</u>	~		~					
	<u>\bar{x}_2</u>	<u>\bar{x}_2</u>	<u>\bar{x}_2</u>					
	$y_1 = x_0 \bar{x}_1 \bar{v}$							

v	x ₂	x ₁	x ₀	y ₇	y ₆	y ₃	y ₁	y ₀
0	0	0	0	0	0	0	0	1
	0	0	1	0	0	0	1	0
	0	1	0	~	~	~	~	~
	0	1	1	0	0	1	0	0
	1	0	0	~	~	~	~	~
	1	0	1	~	~	~	~	~
	1	1	0	0	1	0	0	0
	1	1	1	1	0	0	0	0
1	~	~	~	0	0	0	0	0

Рис.4.4 – Синтез дешифратора 3×5 с прямыми выходами в полном базисе

4.2 Шифратор

Шифратором называется КЦУ, преобразующее активный сигнал на одном (неприоритетный шифратор) или нескольких (приоритетный шифратор) из n информационных входов в m -разрядное двоичное число на выходе.

Шифраторы используются в устройствах ввода информации в цифровые системы и устройства.

Шифратор с m выходами и $n = 2^m$ входами называется полным, в противном случае – неполным (частично определенное КЦУ).

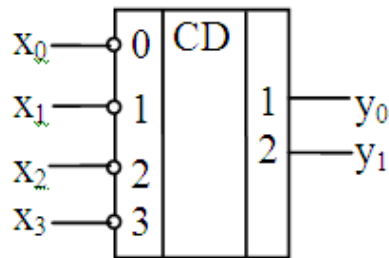


Рис. 4.5 – Полный шифратор

На рис. 4.5 приведено условное графическое обозначение шифраторов на примере полного шифратора 4×2 ($n = 4$, $m = 2$) с инверсными входами (с активным нулевым сигналом на входе). Для входов используется сквозная нумерация, а выходы обозначаются весовыми коэффициентами двоичных разрядов, что позволяет правильно определить двоичное число на выходе шифратора.

X_3	X_2	X_1	X_0	y_1	y_0
1	1	1	0	0	0
1	1	0	1(\sim)	0	1
1	0	1(\sim)	1(\sim)	1	0
0	1(\sim)	1(\sim)	1(\sim)	1	1
1	1	1	1	1	1

Рис. 4.6 – Таблица истинности полного шифратора

Таблица истинности неприоритетного и приоритетного (значения

для входов указаны в скобках) полных шифраторов, ограниченная шифратором 4×2 показана на рис. 4.6. Из таблицы следует, что двоичное число на выходе неприоритетного шифратора соответствует десятичному номеру активного входа. Для приоритетного шифратора допускается наличие активного сигнала одновременно на нескольких входах. В этом случае двоичное число на выходе соответствует наибольшему по номеру активному входу. Из таблицы также видно, что шифраторам свойственна неопределенность: одно и то же двоичное число на выходе образуется как при активном старшем входе, так и пассивных всех входах. Если к тому же шифратор дополняется входом стробирования, образуется еще одна неопределенность: в режиме блокирования все выходы шифратора тоже будут установлены в единичное значение. Для идентификации этих ситуаций в интегральных шифраторах предусматриваются два служебных выхода.

В маркировке микросхем шифраторов используются буквы ИВ.

Рассмотрим пример построения шифратора для преобразования десятичного одноразрядного кода (десятичных чисел от 0 до 9) в двоичный код. При этом предполагается, что сигнал, соответствующий логической единице, в каждый момент времени подаётся только на один вход. Условное обозначение шифратора и таблица соответствия кода (рис.4.7)

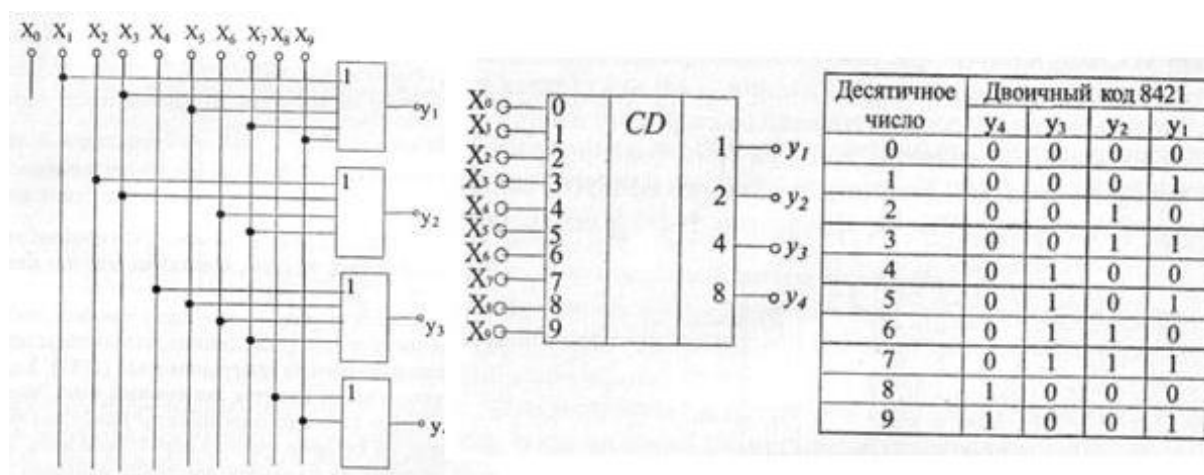


Рис.4.7 – Шифратор с таблицей истинности

На практике часто используют шифратор с приоритетом. В таких шифраторах код двоичного числа соответствует наивысшему номеру входа, на который подан сигнал «1», т. е. на приоритетный шифратор допускается подавать сигналы на несколько входов, а он выставляет на выходе код числа, соответствующего старшему входу.

Рассмотрим в качестве примера шифратор с приоритетом (приоритетный шифратор) К555ИВ3 серии микросхем К555 (ТТЛШ) (рис.4.8).

Шифратор имеет 9 инверсных входов, обозначенных через $\overline{PR1}, \dots, \overline{PR9}$. Аббревиатура PR обозначает «приоритет». Шифратор имеет четыре инверсных выхода $\overline{B1}, \dots, \overline{B8}$. Аббревиатура В означает «шина» (от англ. bus). Цифры определяют значение активного уровня (нуля) в соответствующем разряде двоичного числа. Например, $\overline{B8}$ обозначает, что ноль на этом выходе соответствует числу 8. Очевидно, что это неполный шифратор.

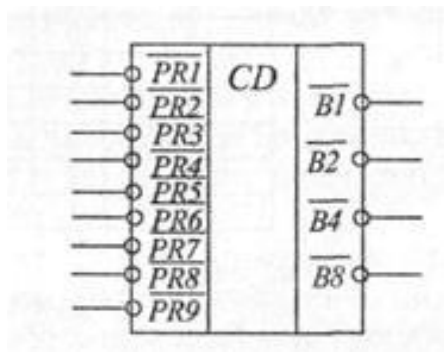


Рис.4.8 – Шифратор

Если на всех входах — логическая единица, то на всех выходах также логическая единица, что соответствует числу 0 в так называемом инверсном коде (1111). Если хотя бы на одном входе имеется логический ноль, то состояние выходных сигналов определяется наибольшим номером входа, на котором имеется логический ноль, и не зависит от сигналов на входах, имеющих меньший номер.

Краткие итоги:

Изучили основные принципы работы типовых КЦУ дешифраторов, шифраторов. Рассмотрели понятие управляющего сигнала, таблицы истинности типовых КЦУ. Выяснили в чем различие шифраторов и дешифраторов.

Вопросы для самопроверки

1. Какие сигналы подаются на входы типовых КЦУ?
2. Дайте определение информационного сигнала.
3. Дайте определение сигнала управления.
4. Дайте определение дешифратора.
5. Каково основное назначение дешифраторов?
6. Сколько выходов имеет полный дешифратор с тремя входами?
7. Дайте определение шифратору.
8. Каково основное назначение шифратора?
9. Чем отличается работа шифратора от дешифратора?

Лекция 5

Типовые КЦУ. Мультиплексор, демультиплексор и преобразователь кода.

В лекции рассматриваются типовые КЦУ мультиплексор, демультиплексор и преобразователь кода. Их принцип работы и построения.

Цель лекции: Ознакомиться с основными принципами работы типовых КЦУ мультиплексора, демультиплексора и преобразователя кода. Рассмотреть синтез мультиплексора, демультиплексора и преобразователя кода. Структуру построения мультиплексора.

5.1 Мультиплексор

Мультиплексором называется КЦУ, обеспечивающее подключение к единственному выходу одного из n информационных входов, выбор которого производится m -разрядным двоичным числом, поступающим на управляющие (адресные, селективные) входы. Очевидно, число селективных входов $m = \lceil \log_2(n) \rceil$.

Мультиплексоры используются для организации мультиплексной линии или перехода от параллельной передачи двоичных наборов (всех разрядов в одном такте) к последовательной (поразрядной).

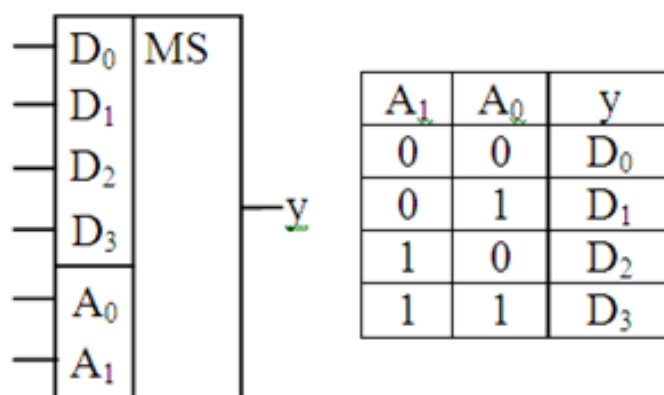


Рис. 5.1 – Мультиплексор

На рис. 5.1 приведены таблица истинности и условное графическое

обозначение мультиплексоров на примере мультиплексора 4×1 ($n = 4$, один выход). Буквой D обозначены информационные входы, а буквой A – адресные. Индекс при букве A обозначает номер разряда соответствующего двоичного числа. Используется и сквозная нумерация входов цифрами от 0 до $n+m-1$, но селективные (адресные) входы всегда узнаются по полочке, отделяющей их от информационных входов.

В маркировке микросхем мультиплексоров используются буквы КП.

На базе любого мультиплексора $n \times 1$ можно построить мультиплексор большего размера, путем каскадной схемы соединения.

Пример: мультиплексор 16×1 на базе микросхем мультиплексора 4×1 :

Количество микросхем первой ступени: $K_1 = \lceil N/M \rceil$. Их селективные входы соединены параллельно и являются селективными входами младших разрядов искомого мультиплексора.

Последующие ступени используются для постепенного уменьшения числа выходов до одного. Так, число микросхем второй ступени: $K_2 = \lceil K_1/M \rceil$ (в примере $K_2 = 4/4 = 1$). Их селективные входы также соединяются параллельно и являются селективными входами следующих разрядов искомого мультиплексора. Информационные же входы соединяются с выходами мультиплексоров первой ступени.

Аналогично организуются третья и т.д. ступени.

Наконец, селективные входы мультиплексора последней ступени являются селективными входами старших разрядов искомого мультиплексора (Рис 5.2)

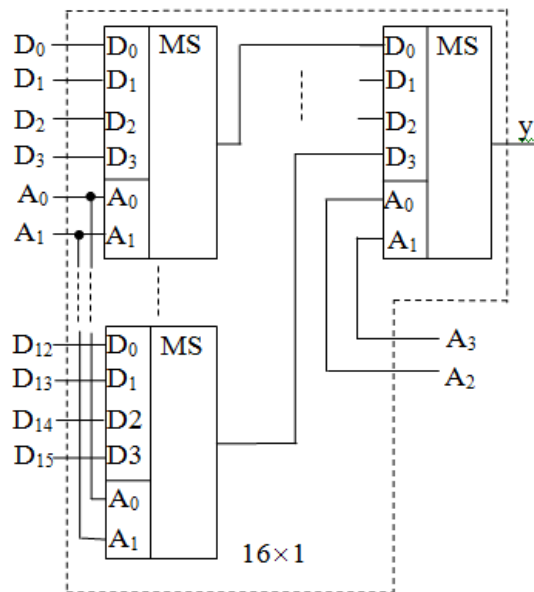


Рис. 5.2 Мультиплексор 16×1 на базе микросхем мультиплексора 4×1

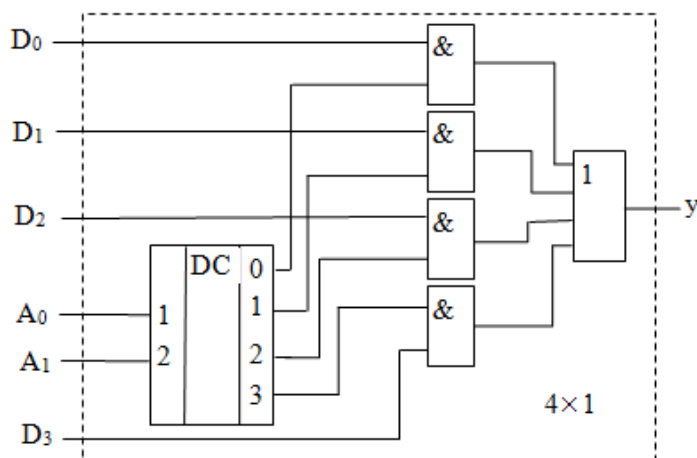
Синтез полного мультиплексора

Пример: синтеза мультиплексора 4×1 без стробирования в полном базисе (Рис. 5.3).

A_1	A_0	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$y = \bar{A}_0 \bar{A}_1 D_0 \vee A_0 \bar{A}_1 D_1 \vee \bar{A}_0 A_1 D_2 \vee A_1 A_0 D_3.$$

$y = y_0 D_0 \vee y_1 D_1 \vee y_2 D_2 \vee y_3 D_3$, где y_0, \dots, y_3 – выходы дешифратора 2×4.



A_1	A_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Рис. 5.3 – Синтеза мультиплексора 4×1

В результате такой замены синтез мультиплексора сводится к синтезу дешифратора: где y_0 соответствует D_0 , $y_1 - D_1$ и т.д.

Такой подход позволяет легко записать минимальную ФАЛ для любого из входов частично определенного мультиплексора.

5.2 Демультимплексор

Демультимплексором называется КЦУ, обеспечивающее подключение единственного информационного входа к одному из m выходов, выбор которого осуществляется n -разрядным двоичным числом на управляющих входах.

Демультимплексоры решают задачу, обратную задаче мультиплексирования.

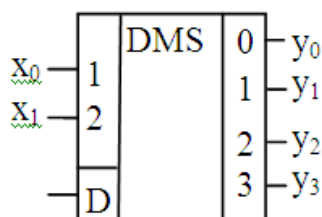


Рис. 5.4 – Демультимплексор

На рис. 5.4 приведено условное графическое обозначение демультимплексоров на примере демультимплексора 1×4 (один информационный вход и четыре выхода). Управляющие входы обозначаются весовыми коэффициентами двоичных разрядов, а для выходов используется сквозная нумерация.

x_1	x_0	y_3	y_2	y_1	y_0
0	0	1	1	1	D
0	1	1	1	D	1
1	0	1	D	1	1
1	1	D	1	1	1

Рис. 5.5 - Таблица истинности демультимплексора

Таблица истинности демультиплексоров, ограниченная демультиплексором 1×4 (рис. 5.4), показана на рис. 5.5. Из таблицы следует, что номер выхода, к которому подключается информационный вход D , является десятичным эквивалентом двоичного числа x_1x_0 на управляющих входах. Кроме того, нетрудно видеть, что в качестве демультиплексора вполне можно использовать дешифратор со стробированием, если считать стробирующий вход информационным, а информационные входы – управляющими. По этой причине в интегральном исполнении демультиплексоры не выпускаются, а дешифратор со стробированием называют дешифратором-демультиплексором, подчеркивая тем самым возможность выполнения им двух функций.

5.3 Преобразователь кода

Преобразователями кода называются КЦУ, реализующие процедуру кодирования – изменение закона расположения нулей и единиц относительно исходных двоичных наборов.

В интегральном исполнении выпускаются только преобразователи двоичного кода в двоично-десятичный или семисегментный код, а также двоично-десятичного кода в двоичный. В маркировке таких микросхем используются буквы ПР, например, К155ПР6. Другие преобразователи кода либо синтезируются как КЦУ, либо реализуются на базе программируемой логической матрицы (ПЛМ).

ПЛМ – это универсальная комбинационная схема, обеспечивающая преобразование входных n -разрядных кодовых слов в m -разрядные кодовые слова на выходе. Структурно (рис. 5.6) ПЛМ состоит из трех уровней логических элементов и двух матриц соединительных линий $M1$ и $M2$.

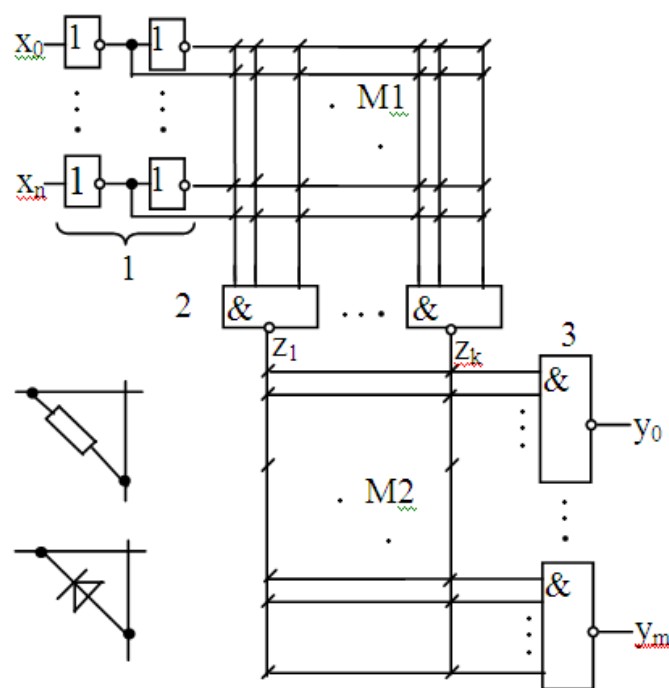


Рис. 5.6 – Структура ПЛМ

В исходном состоянии во всех точках пересечения соединительных линий обеих матриц электрический контакт может быть либо обеспечен, либо отсутствовать. В первом случае место соединения выполняется в виде плавкой перемычки, а во втором – в виде р-п перехода. В соответствии с этим программирование ПЛМ заключается либо в разрушении определенных контактов путем пережигания плавкой перемычки, либо в их установлении путем пробоя р-п перехода.

Закон преобразования кодовых слов ПЛМ представленной структуры описывается системой ФАЛ, записанных в СДНФ: $\{y_i = \vee z_j\}$, где z_j – j -й член i -й функции. Соответственно с этим логические элементы первого уровня обеспечивают прямые и инверсные значения входных переменных, необходимые для формирования членов ФАЛ. Кроме того, второй ряд логических элементов этого уровня введен с целью обеспечения минимальной нагрузки для генератора исходного кода. Собственно члены ФАЛ z_j формируются логическими элементами второго уровня совместно с матрицей $M1$. Элементы третьего уровня совместно с матрицей $M2$ обеспечивают необходимую структуру каждой из m ФАЛ.

Краткие итоги:

Изучили основные принципы работы типовых КЦУ мультиплексоров, демультиплексоров и преобразователей кода. Рассмотрели понятие управляющего сигнала, таблицы истинности типовых КЦУ. Выяснили в чем различие мультиплексоров и демультиплексоров.

Вопросы для самопроверки

1. Дайте определение мультиплексора?
2. Когда применяется мультиплексор?
3. Каким выражением описывается мультиплексор?
4. Дайте определение демультиплексора.
5. Какие задачи решает демультиплексор?
6. Назначение стробирующего входа.
7. Чем отличаются мультиплексор от демультиплексора?
8. Дайте определение преобразователю кода.
9. Опишите работу преобразователя кода.

Лекция 6

Последовательные цифровые устройства

В лекции рассматриваются последовательные цифровые устройства (ПЦУ). Дается понятие основных компонентов ПЦУ. Описывается принцип работы триггеров и синтез ПЦУ.

Цель лекции: Ознакомиться с основными принципами работы типовых ПЦУ. Дать понятия способа задания ПЦУ и триггера. Изучить классификацию триггеров. Рассмотреть поэтапно синтез ПЦУ

6.1. Понятие и способ задания ПЦУ

Последовательным называется цифровое устройство с памятью, в котором текущие двоичные сигналы на выходе зависят как от текущих двоичных сигналов на входе, так и от предыдущих состояний устройства.

Под состоянием ПЦУ понимается двоичная информация, считываемая из его памяти в данный момент времени.

ПЦУ синхронизируются тактовыми импульсами и работают циклами, автоматически переходя с каждым тактом из одного состояния в другое, формируя при этом на выходе определенные двоичные сигналы. Каждый цикл всегда начинается и заканчивается некоторым заранее определенным начальным состоянием.

Наиболее общей моделью ПЦУ является автомат Мили (рис. 6.1).

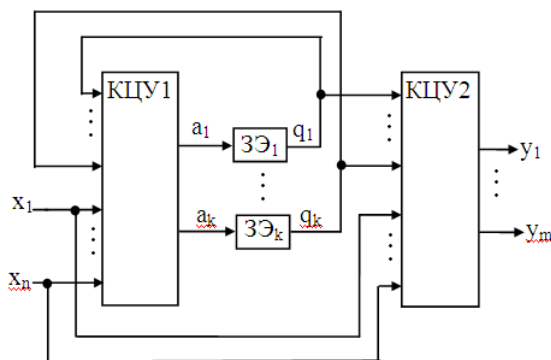


Рис. 6.1 – Автомат Мили

Запоминающие элементы (ЗЭ) хранят все состояния цикла работы

ПЦУ в заданной последовательности в виде k -разрядных двоичных наборов. КЦУ1 в зависимости от текущих сигналов $X = \{x_i\}$ на входе и текущего состояния $Q = \{q_i\}$ ПЦУ вырабатывает двоичные сигналы a_1, \dots, a_k управления памятью. Эти сигналы определяют состояние, в которое ПЦУ перейдёт в следующем такте. КЦУ2 в зависимости от текущих двоичных сигналов на входе и текущего состояния ПЦУ вырабатывает выходные двоичные сигналы.

Автомат Мили задаётся двумя системами логических функций: функций переходов $Q^{t+1} = f(X^t, Q^t)$ и функций выходов $Y^t = \varphi(X^t, Q^t)$, где индекс t соответствует текущему такту цикла, а $(t+1)$ – последующему. Однако на практике часто оказывается достаточной более простая модель, в которой отсутствует связь КЦУ2 с входом ПЦУ. В этом случае функции выходов упрощаются: $Y^t = \varphi(Q^t)$. Такая модель называется конечным автоматом Мура.

Функции переходов определяют состояние ПЦУ в следующем такте и позволяют синтезировать КЦУ1. Задаются они в виде таблицы переходов (Табл. 6.1).

Таблица 6.1

Таблица переходов

№ состояния	Вход			Состояние ЗЭ			Сигнал управления		
	x_n	...	x_1	q_k	...	q_1	a_k	...	a_1
Q_n									
.									
.									
.									

Заполняется она, начиная с начального состояния Q_n , соответствующего нулевому такту. В ее строках записываются соответствующие двоичные наборы. При этом значения управляющих сигналов выбираются из условия обеспечения следующего состояния ПЦУ.

Функции выходов определяют выходной двоичный набор в текущем такте и позволяют синтезировать КЦУ2. Задаются они в виде таблицы выходов (Табл. 6.2).

Таблица 6.2

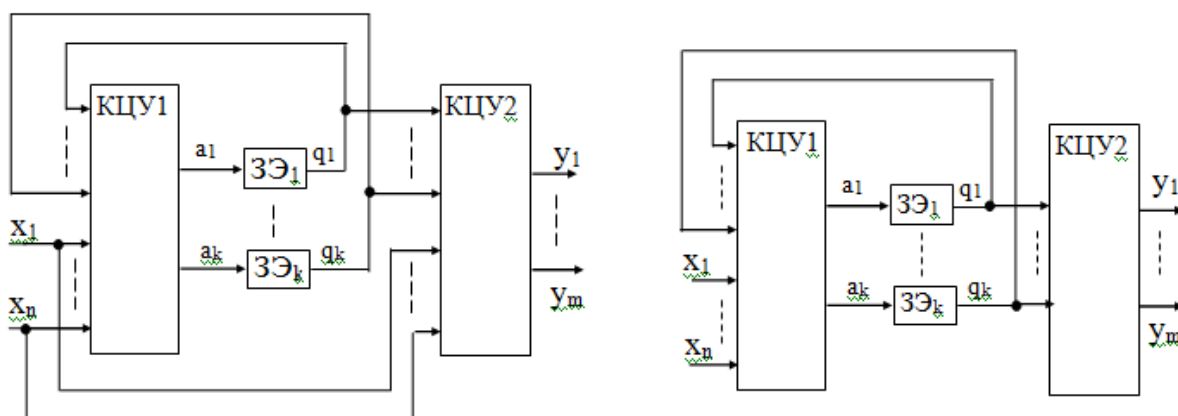
Таблица выходов

№ состояния	Состояние ЗЭ			Выход		
	q_k	...	q_1	a_k	...	a_1
Q_H						
.						
.						
.						

Обычно обе таблицы объединяются в одну, которая называется автоматной таблицей. Однако автоматная таблица не отражает динамику поведения ПЦУ. Поэтому для пояснения работы ПЦУ автоматную таблицу дополняют временными диаграммами.

Другой тип автомата – автомат Мура. Особенность автомата Мура в том, что в нем выходной сигнал зависит лишь от внутреннего состояния и не зависит от входного сигнала.

Автомат МилиАвтомат Мура



$$Q^{t+1} = f(X^t, Q^t) \quad \text{функции переходов} \quad Q^{t+1} = f(X^t, Q^t)$$

$$Y^t = \varphi(X^t, Q^t) \quad \text{функции выходов} \quad Y^t = \varphi(Q^t)$$

Рис. 6.2 - Автоматы Мили и Мура

Таблица переходов:Таблица выходов:

№ состо- яния	Вход			Сост. 3Э			Сиг. упр.		
	x_n	...	x_1	q_k	...	q_1	a_k	...	a_1
Q_H									
·									
·									
·									

№ состо- яния	Сост. 3Э			Выход		
	q_k	...	q_1	y_m		y_1
Q_H						
·						
·						
·						

Функции переходов и выходов позволяют легко установить тип ПЦУ – это автомат Мили или автомат Мура (рис. 6.2)

6.2 Понятие и классификация триггеров

Основными компонентами ПЦУ являются запоминающие элементы, которые реализуются специальными устройствами – триггерами.

Триггер – это одноразрядный элемент памяти, предназначенный для хранения одного бита информации. Основной способ построения триггеров – использование обратных связей. Именно за счёт них обеспечивается возможность запоминания.

Любой триггер имеет два выхода – прямой и инверсный, но состояние триггера определяется сигналом на прямом выходе. Число входов в зависимости от типа триггера может составлять от двух до пяти.

По способу приема информации различают асинхронные и синхронные триггеры. Синхронные триггеры записывают бит информации только при наличии активного (разрешающего) сигнала на входе синхронизации. Пассивное значение синхросигнала определяет режим хранения триггера. Асинхронные триггеры не имеют входа синхронизации и записывают бит информации в момент его подачи на информационные входы.

Синхронные триггеры могут быть со статическим или динамическим управлением по входу синхронизации. При статическом управлении активным является уровень логического 0 (инверсный синхровход) или логической 1 (прямой синхровход). На условном графическом обозначении триггера инверсный синхровход показывается кружочком ,

а прямой – без кружочка . При динамическом управлении активным является фронт или срез-синхроимпульса. На условном графическом обозначении триггера фронт показывается прямой косой линией , а срез – обратной .

По функциональным возможностям различают триггеры с отдельной установкой состояний 0 и 1 (RS-триггер), с приёмом информации по одному входу D (D-триггер), универсальный с информационными входами J и K (JK-триггер) и со счётным входом T (T-триггер).

6.3. Типовые триггеры

Асинхронные RS-триггеры имеют два информационных входа, один из которых обозначается буквой S (Set – установка), а другой – буквой R (Reset – сброс).

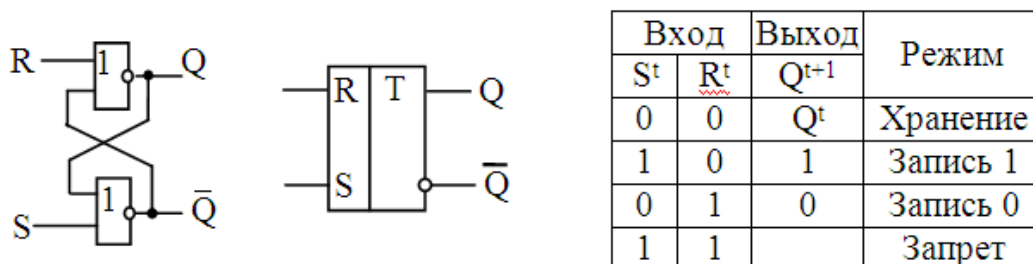


Рис. 6.3 – Асинхронный RS-триггер с прямыми входами

На рис. 6.3 приведены реализация триггера на логических элементах (ЛЭ) ИЛИ-НЕ, его условное графическое обозначение при исполнении в виде интегральной схемы и таблица переходов. Как видно, активным значением информационного сигнала является 1. Поэтому RS-триггер, построенный на логических элементах ИЛИ-НЕ, называют RS-триггером с прямыми входами. Однако асинхронный RS-триггер может быть построен и на логических элементах И-НЕ. Такая реализация триггера, ее условное графическое обозначение при исполнении в виде интегральной схемы и таблица переходов приведены на рис. 6.4.

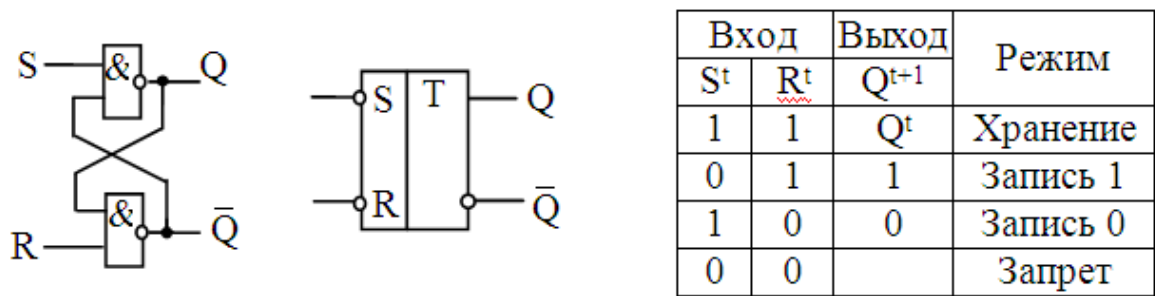


Рис. 6.4 – Асинхронный RS-триггер с инверсными входами

Здесь активным значением информационного сигнала является 0. Поэтому RS- триггер, построенный на логических элементах И-НЕ, называют RS-триггером с инверсными входами.

Из таблиц переходов рис. 6.3 и 6.4 видно, что, во-первых, активный сигнал на входе S приводит к установке триггера в единичное состояние, а на входе R – в нулевое. По этой причине у триггеров любого другого типа асинхронный вход принудительной установки в 1 обозначается буквой S, а в 0 – буквой R. Эти входы являются входами первого приоритета, то есть при активном сигнале на одном из этих входов триггер не реагирует на сигналы по другим входам. Во-вторых, существует запрещенная комбинация входных сигналов. Запрет следует понимать только в информационном смысле. То есть он имеет смысл лишь при использовании триггера в качестве запоминающего элемента. Действительно, активные сигналы одновременно на S и R входах триггера устанавливают его выходы в одинаковое единичное состояние, что приводит к неопределенности относительно записываемого бита информации. Однако такая комбинация входных сигналов вполне естественна при использовании триггера в качестве управляющего элемента устройства. Возникновение запрещенной комбинации не приводит к выходу триггера из строя.

В маркировке микросхем RS-триггеров используются буквы TR.

Триггеры всех других типов выпускаются промышленностью как синхронные.

D-триггеры имеют один информационный вход D и вход синхронизации C. В интегральном исполнении D-триггеры выпускаются как со статическим, так и с динамическим управлением по входу C. В любом случае в маркировке микросхем D-триггеров используются буквы ТМ.

D-триггеры со статическим управлением не имеют входов принудительной установки, поскольку используются исключительно в качестве запоминающих элементов. Их условное графическое обозначение (для примера с единичным активным синхросигналом – прямым синхровходом), таблица переходов и временные диаграммы, где крестик означает безразличное состояние триггера, приведены на рис. 6.5.

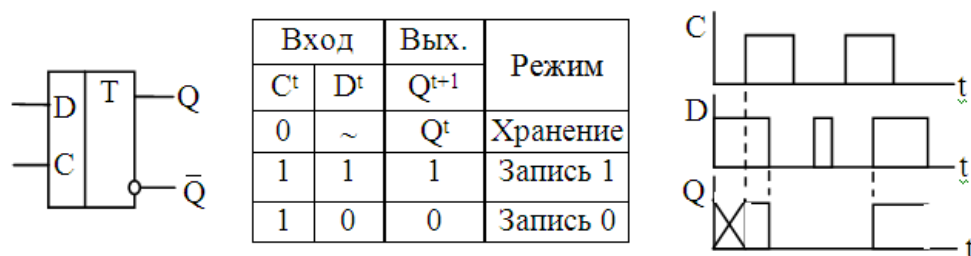


Рис. 6.5 – D-триггер со статическим управлением

Как видно, пока сигнал на входе C сохраняет активное (в данном случае единичное) значение, непрерывно производится запись информации со входа D.

D-триггеры с динамическим управлением дополняются входами S и R принудительной установки начального состояния. Их условное графическое обозначение (для примера с активным фронтом синхросигнала), таблица переходов и временные диаграммы приведены на рис. 6.6.

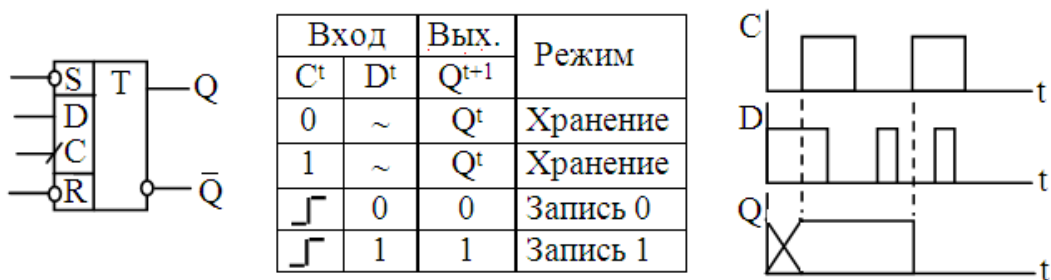


Рис. 6.6 – D-триггер с динамическим управлением

Как видно, запись информации с входа D производится только по фронту синхросигнала (тактового импульса).

JK -триггеры имеют два информационных входа J и K, вход синхронизации C и входы R и S принудительной установки начального состояния. В интегральном исполнении выпускаются только с динамическим управлением по входу синхронизации. В маркировке микросхем JK-триггеров используются буквы ТВ.

Условное графическое обозначение (для примера с активным срезом синхросигнала), таблица переходов и временные диаграммы работы JK - триггера приведены на рис. 6.7

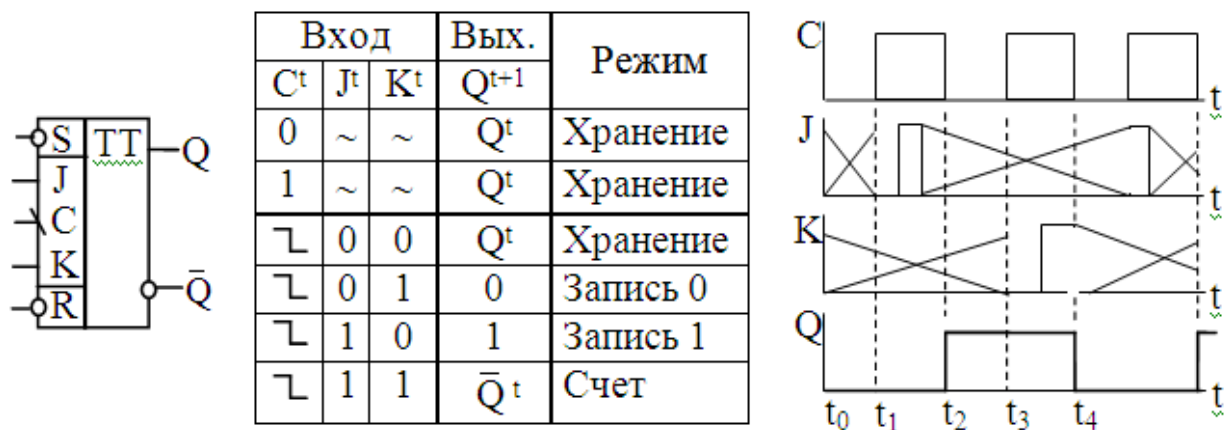


Рис.6.7 – JK-триггер

Счётным называется режим, при котором триггер с каждым активным синхросигналом (тактовым импульсом) переключается в противоположное состояние.

В зависимости от значения активного синхросигнала такт разделяется на предстартовый полутакт и пассивный полутакт. Для

триггера рис. 6.7 предстартовым полутактом является импульс такта, а пассивным полутактом – его пауза. В течение пассивного полутакта входы J и K триггера отключены от входных цепей, благодаря чему обеспечивается собственно режим хранения. В течение предстартового полутакта к входным цепям подключается только тот информационный вход, активный сигнал (для триггера рис. 6.7 – единица) на котором может изменить текущее состояние триггера. Так, применительно к триггеру рис. 6.7, в интервале времени $t_1 - t_2$ будет подключен вход J, а в интервале времени $t_3 - t_4$ будет подключен вход K. Если в течение предстартового полутакта на подключенном информационном входе хоть на мгновение появляется активный сигнал, по активному синхросигналу триггер переключится в соответствующее состояние. В противном случае текущее состояние триггера не изменится.

Т-триггеры имеют единственный вход Т, на который поступают тактовые импульсы. По этой причине и поскольку Т-триггеры работают только в режиме счета, Т-вход называют счетным входом триггера.

К Т-триггеру относится и TV-триггер – Т-триггер, дополненный входом управления V. Сигнал по входу V либо разрешает работу Т-триггера, либо переводит его в режим хранения независимо от тактовых импульсов по входу Т.

Условное графическое обозначение и таблица переходов Т- и TV-триггеров на примере последнего приведены на рис. 6.8.

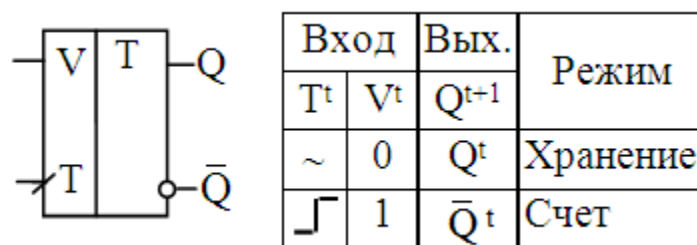


Рис. 6.8 – TV-триггер

В интегральном исполнении ни Т-, ни TV-триггеры не выпускаются,

поскольку легко могут быть построены на основе D- или JK-триггеров (рис.6.9, рис. 6.10, рис. 6.11).

Реализация Т-триггера

На базе D-триггера На базе JK-триггера

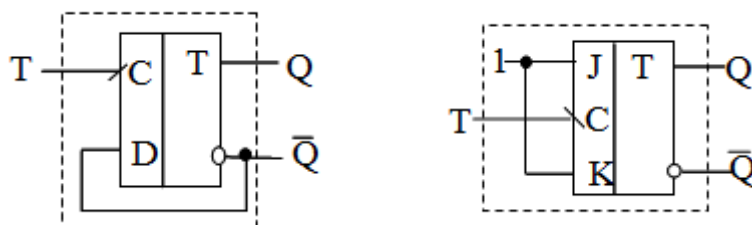


Рис.6.9 – Реализация Т-триггера

Реализация TV-триггера

На базе D-триггера

На базе JK-триггера

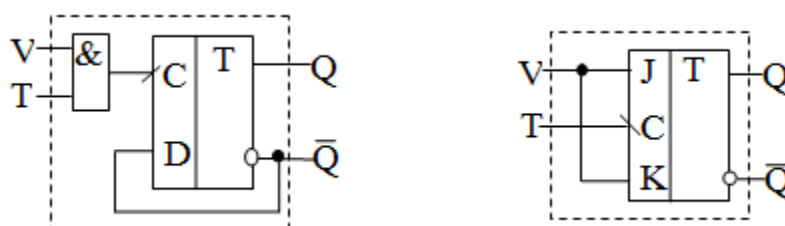


Рис. 6.10 – Реализация TV-триггера

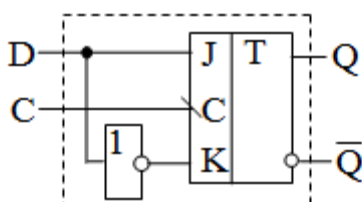


Рис. 6.11 – Реализация D-триггера на базе JK-триггера

Пусть D-триггер в состоянии 1. По правилу работы Т-триггера по фронту D-триггер должен переключиться в 0. Для этого 0 должен быть на D-входе. Этот 0 может быть взят с инверсного выхода D-триггера.

Пусть теперь D-триггер в состоянии 0. По правилу работы Т-триггера по фронту D-триггер должен переключиться в 1. Для этого 1 должна быть на D-входе. Эту 1 опять-таки можно взять с инверсного выхода D-триггера.

Пусть JK-триггер в состоянии 1. По правилу работы T-триггера по срезу JK-триггер должен переключиться в 0. Для этого 0 должен быть на J-входе (J-вход пассивен) и 1 на K-входе (K-вход активен).

Пусть теперь JK-триггер в состоянии 0. По правилу работы T-триггера по срезу JK-триггер должен переключиться в 1. Для этого 1 должна быть на J-входе (J-вход активен) и 0 на K-входе (K-вход пассивен).

Как видно, в любом случае K-вход должен быть инверсным J-входу.

6.4 Синтез ПЦУ

Синтез ПЦУ проводится в пять этапов.

На 1 этапе определяется минимальное число состояний, необходимое для построения устойчивого ПЦУ, соответствующего заданным условиям работы. Когда решение этой задачи не очевидно, следует воспользоваться графом переключений.

Графом переключений называется ориентированный граф, состоящий из узлов и направленных связей между ними (рис. 6.12).

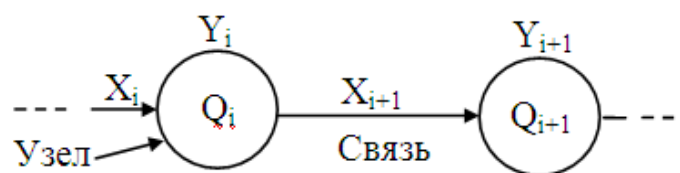


Рис. 6.12 - Граф переключений

Каждый узел отображает состояние ПЦУ в данном такте цикла. Внутри узла в десятичной или двоичной системе счисления записывается номер его состояния. Около каждого узла указывается десятичный номер выходного двоичного набора или непосредственно двоичный набор, соответствующий данному такту. Связи между узлами отображают переходы ПЦУ из одного состояния в другое. Над каждой связью выписываются входные двоичные сигналы, вызывающие переход ПЦУ из текущего i -го состояния в последующее $(i+1)$ -е состояние.

В терминах графа переключений можно сформулировать условие

устойчивости: ПЦУ устойчиво, если его граф переключений не содержит ни одного замкнутого контура, дуги которого помечены одним и тем же значением входного сигнала.

Существует два способа устранения неустойчивости ПЦУ: переход к динамическому способу управления триггерами и увеличение числа состояний. Реализация первого способа тривиальна. Второй способ более сложный, но иногда единственно возможный: для устранения одного замкнутого контура графа переключений требуется дополнительно ввести не менее двух состояний (одного триггера).

На 2 этапе определяется количество триггеров и выбирается их тип.

Количество триггеров ПЦУ определяется выражением: $N_T = \lceil \log_2 N_C \rceil$, где N_C – число состояний автомата, включая начальное.

Тип триггеров выбирается по принципу наибольшего совпадения их правил работы с правилами работы проектируемого устройства.

На 3 этапе определяются функции переходов и проводится синтез КЦУ1.

Функции переходов представляются таблицей переходов, где управляющие сигналы записываются на основе правил работы выбранных триггеров, состояния запоминающих элементов – в соответствии с принятым порядком их следования, а входные двоичные наборы – в соответствии с правилами работы проектируемого устройства. Полученная таблица рассматривается как таблица истинности КЦУ1, что позволяет воспользоваться стандартной методикой его синтеза.

На 4 этапе определяются функции выходов и проводится синтез КЦУ2.

Функции выходов представляются таблицей выходов, где управляющие сигналы записываются, как и в таблице переходов, а выходные двоичные наборы – в соответствии с правилами работы проектируемого устройства. Построение таблицы выходов основывается

на графе переключений. Полученная таблица рассматривается как таблица истинности КЦУ2, что позволяет воспользоваться стандартной методикой его синтеза.

На 5 этапе строится структурная схема ПЦУ. При этом входы синхронизации триггеров соединяются параллельно (объединяются) на общий вход синхронизации ПЦУ, называемый тактовым входом устройства. В случае обнаружения в структурной схеме типовых структур производится коррекция схемы путем использования условных графических обозначений этих структур.

Наиболее часто используемые ПЦУ – счетчики и регистры, относятся к категории типовых устройств.

Краткие итоги:

Изучили основные принципы работы ПЦУ. Рассмотрели понятие триггера. Выяснили, как классифицируются триггеры. Усвоили различия в работе триггеров, их маркировку. Освоили синтез ПЦУ.

Вопросы для самопроверки

1. Дайте определение ПЦУ.
2. Как синхронизируются ПЦУ?
3. Дайте понятие триггера.
4. Какова классификация триггеров?
5. Дайте определение синхронного и асинхронного триггера.
6. Каков принцип работы синхронного и асинхронного триггера?
7. Перечислите типовые триггеры.
8. Каков принцип работы D-триггера?
9. Каков принцип работы JK – триггера?
10. Каков принцип работы T – триггера?
11. Расскажите поэтапно синтез ПЦУ.

Лекция 7

Принципы управления микропроцессора

В лекции рассматриваются принципы управления микропроцессором. Классификация микропроцессоров, их работа.

Цель лекции: Ознакомиться с основными принципами построения микропроцессорных систем. Изучить классификацию микропроцессоров, принципы аппаратного управления (жесткой логики и гибкой логики).

В основе построения микропроцессорных систем (МПС) обработки информации лежит модульный принцип.

Модулем микропроцессорной системы является её функциональный блок, выполненный в виде конструктивно законченного устройства – обычно в виде одной или нескольких БИС либо в виде плат.

Модули соединяются между собой посредством специальных устройств, называемых интерфейсами.

Основными модулями МПС являются однокристальный микропроцессор (МП), постоянная и оперативная (основная) памяти, устройства ввода-вывода информации и блоки управления (контроллеры).

Суть проектирования микропроцессорных систем на основе выбранного однокристального микропроцессора состоит в следующем:

- 1) выбор внешних устройств, предназначенных для связи МП с устройствами ввода-вывода информации;
- 2) организация связи этих устройств с микропроцессором;
- 3) выбор ёмкости памяти для размещения программы и промежуточных данных, а также способа её организации;
- 4) программирование МП на выполнение требуемых функций при известной конфигурации МПС с использованием системы команд МП.

7.1 Классификация микропроцессоров.

Микропроцессором называется функционально законченное программно-управляемое устройство обработки цифровой информации.

Числовая информация, заданная для обработки, называется **данными**.

Физически **микропроцессоры (МП)** выпускаются в виде одной или нескольких **больших или сверхбольших интегральных схем (БИС или СБИС)**, содержащих на одном кристалле сотни тысяч элементарных транзисторов.

МП дополняются БИС для хранения, ввода-вывода данных, управления и синхронизации, сопряжения интерфейсов и т.п. Эти БИС совместимы с МП по архитектуре, конструктивному исполнению и параметрам. Такие наборы БИС называются **микропроцессорными комплектами (МПК)**.

МП можно классифицировать по весьма большому числу характеристик. Мы воспользуемся одним из возможных вариантов классификации.

1. По назначению различают универсальные и специализированные микропроцессоры.

Универсальные МП могут осуществить преобразование информации в соответствии с любым заданным алгоритмом.

Специализированные МП предназначены для решения определённого класса задач и даже, может быть, для решения одной конкретной задачи. Их особенностью являются низкая стоимость, малая потребляемая мощность, компактность, простота управления.

Среди специализированных МП выделяют **микроконтроллеры**, используемые в управлении технологическими процессами, измерениях, научных исследованиях. Их отличительной чертой является работа в реальном масштабе времени.

2. По числу БИС различают секционные (разрядно-модульные) и однокристалльные МП.

Микропроцессорная секция (модуль) представляет собой БИС для обработки небольшого числа разрядов данных (4 или 8). Такие МП обладают возможностью наращивания разрядности, что позволяет на их основе строить МП произвольной разрядности. Кроме того, секционные МП не имеют фиксированного набора команд, а программируются на микрокомандном уровне. Это позволяет реализовать оптимальный для данной задачи набор команд и отдельных процедур.

Однокристалльные МП реализуются в виде одной БИС или СБИС. Такие МП имеют фиксированный набор команд и фиксированную разрядность. Благодаря успехам современных технологий в повышении степени интеграции и быстродействия в настоящее время в микропроцессорной технике основное место занимают однокристалльные МП (процессоры).

3. По способу управления различают МП с микропрограммным и с "жестким" (аппаратным) управлением.

Микропрограммное управление характерно для секционных МП, а "жесткое" – для однокристалльных.

4. По возможности прерываний выполняемой программы МП могут не иметь, а могут иметь одно- или многоуровневую систему прерывания.

В многоуровневых системах прерывания разрешается прерывание прерывания. Такие системы используются в МП, работающих в реальном масштабе времени.

7.2. Декомпозиция МП.

Для облегчения понимания принципа работы микропроцессора, представим его в виде некоторого ПЦУ, состоящего из двух устройств: управляющего и операционного (рис. 7.1).

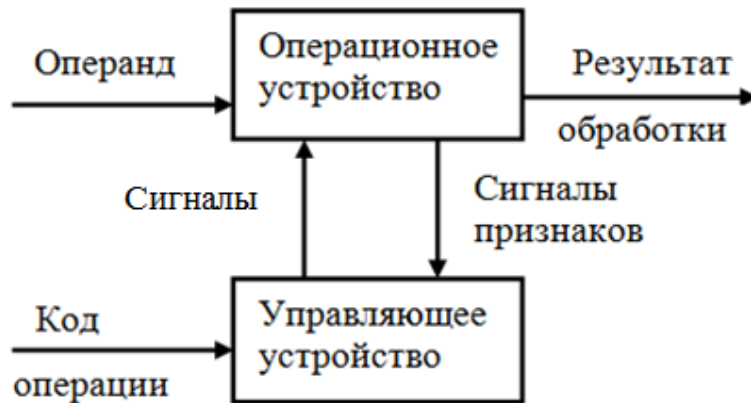


Рис.7.1 – Программное цифровое устройство

Операционное устройство производит прием операндов (чисел участвующих в операции), их хранение и преобразование. Кроме того операционный блок выдает во внешнюю среду результат преобразования, а в управляющее устройство – сигналы признаков.

Сигналы признаков (флаги) несут информацию об особенностях операндов и их отдельных разрядов, а также особенностях промежуточных и конечных результатов обработки (например, равенство нулю, чётность результата, переполнение разрядной сетки и др.).

Управляющее устройство, в зависимости от кода операции и сигналов признаков вырабатывает сигналы управления, обеспечивающие выполнение заданной операции.

Процесс функционирования МП во времени состоит из последовательности элементарных операций. Например, передача кодового слова от одного узла МП к другому, поразрядное инвертирование слова, сдвиг и др.

Элементарная операция выполняется за один такт синхросигнала и называется *микрооперацией*.

В течение такта может быть выполнено несколько микроопераций, но только если результат выполнения каждой из них не зависит от результатов выполнения остальных.

Элементарные преобразования информации производятся операционным блоком под воздействием управляющих сигналов.

Совокупность управляющих сигналов, обеспечивающих выполнение микроопераций в течение одного тактового интервала, называется микрокомандой.

Последовательность микрокоманд, **обеспечивающая выполнение данной операции (команды), называется микропрограммой.**

Управляющее устройство может быть задано как автомат Мили или как автомат Мура, для которых функции переходов и выходов определяются заданной микропрограммой. В связи с этим управляющее устройство часто именуется *микропрограммным* или *управляющим автоматом*.

Для построения управляющего автомата используются принципы "жёсткой" (схемной) и "гибкой" (программируемой) логики.

7.3 Принцип аппаратного управления ("жёсткой" логики).

Типовой управляющий автомат с "жёсткой" логикой имеет следующую структуру (рис.7.2).

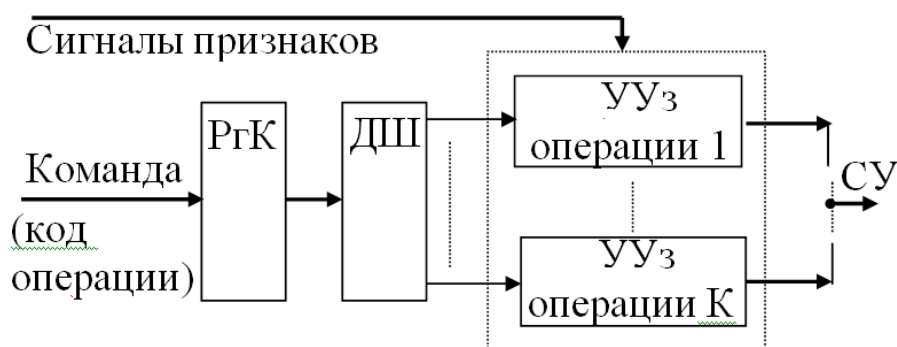


Рис.7.2 – Типовой управляющий автомат с «жесткой» логикой

В управляющем устройстве предусматривается ряд управляющих узлов (УУз), представляющих собой набор логических схем

Команда, поступающая из внешнего ОЗУ фиксируется в регистре команд (РгК) и с помощью дешифратора команд (ДШ) включает соответствующий управляющий узел.

Управляющий узел вырабатывает определённую последовательность сигналов управления (СУ), обеспечивая выполнение данной операции операционным блоком.

В общем случае значения сигналов управления зависят от сигналов признаков, отражающих ход вычислительного процесса.

При таком способе построения управляющего автомата микропрограммы операций заложены в однажды выполненные соединения между логическими схемами управляющих узлов. *Это означает, что набор команд или, иначе говоря, система команд фиксируется и соответствует числу выполняемых операций.*

Поэтому такие микропроцессоры называются с "жёсткой" логикой управления.

Невозможность изменения системы команд после изготовления МП приводит к его узкой специализации. Вместе с тем МП с "жёсткой" логикой управления обеспечивает наивысшее быстродействие при заданной технологии изготовления.

7.4. Принцип микропрограммного управления (гибкой логики).

В управляющем автомате с "гибкой" логикой предусматриваются управляющая память (УП) и блок микропрограммного управления (БМУ) (рис. 7.3).

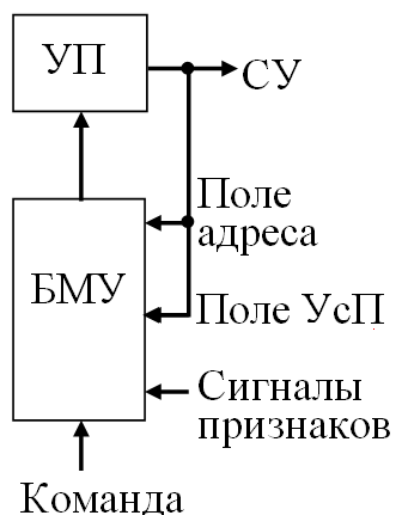


Рис.7.3 – Управляющий автомат с «гибкой» логикой

В УП для каждой операции содержится своя МКП.

УП может быть постоянной или с произвольным обращением, т.е. допускающая как считывание, так и запись. В последнем случае загрузка УП производится пользователем.

Команда, поступающая из внешней памяти, используется БМУ для определения адреса первой МК той МКП, которая реализует заданную операцию.

Далее микрокоманды найденной МКП последовательно считываются из УП. При этом *адрес следующей МК определяется БМУ на основе предыдущей МК.*

Для обеспечения такого процесса управления в МК предусматриваются три поля (три группы разрядов): поле адреса, поле условных переходов (УсП) и поле сигналов управления (СУ). Два первых поля образуют адресную часть МК, а последнее поле – её операционную часть (рис.7.4)

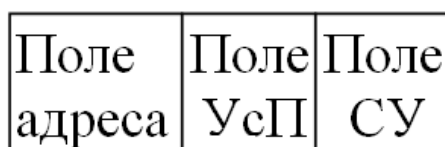


Рис.7.4 – Поля МК

В поле адреса содержится адрес очередной МК.

Поле УсП предусматривается для реализации условных и безусловных переходов.

Один из разрядов этого поля отводится для указания вида перехода (например, 0 – безусловный переход, 1 – условный переход).

Ещё один разряд определяет участие данного вида перехода в определении адреса (например, 1 – участвует, 0 – не участвует).

Остальные разряды используются для указания условий, на которые следует ориентироваться при определении адреса очередной МК.

В результате в зависимости от условия образуются два различных адреса и очередная МК, считывается из одной либо из другой ячейки УП.

Рассмотренный способ управления получил название микропрограммного, а МП с управляющим автоматом на этом принципе называются МП с программируемой логикой.

Достоинством такой организации управления является возможность гибкого изменения системы команд МП с помощью изменения совокупности МКП, реализующих эти команды. Отсюда второе название принципа – принцип "гибкой" логики.

Вместе с тем использование принципа "гибкой" логики может привести к снижению быстродействия из-за увеличения числа тактов реализации микропрограммы.

Микропрограммное управление используется не только в секционных МП, но и в устройствах управления периферийным оборудованием МПС, а также как средство для аппаратной реализации фрагментов операционных систем, трансляторов и т.д.

Управляющие автоматы с "гибкой" логикой различаются по способу формирования сигналов управления.

7.5. Способы формирования сигналов управления в управляющих автоматах с "гибкой" логикой.

Возможно горизонтальное, вертикальное и смешанное микропрограммирование.

При горизонтальном микропрограммировании каждому разряду операционной части МК ставится в соответствие определённый управляющий сигнал, т.е. определённая микрооперация.

Так, если в i -ом разряде стоит 1, то соответствующая микрооперация выполняется независимо от значения других разрядов.

При таком способе операционная часть МК содержит m разрядов, где m – общее число микроопераций.

Достоинствами горизонтального микропрограммирования являются

возможность одновременного выполнения в одном такте любого набора микроопераций и простота формирования сигналов управления.

Однако при этом требуется большая длина МК, поскольку число управляющих сигналов может достигать нескольких сотен.

Поэтому большее распространение получили другие методы.

Краткие итоги:

Изучили принципы работы МП. Их проектирование, классификацию и декомпозицию. Уяснили что такое модуль микропроцессорной системы. Описали принцип аппаратного управления ("жёсткой" логики) и принцип микропрограммного управления (гибкой логики), его достоинства и недостатки.

Контрольные вопросы

1. Что такое модуль микропроцессорной системы?
2. В чем состоит суть проектирования микропроцессорных систем?
3. Классификация микропроцессоров по назначению.
4. Классификация микропроцессоров по числу БИС
5. Классификация микропроцессоров по способу управления
6. Классификация микропроцессоров по возможности прерываний выполняемой программы
7. Каков принцип декомпозиции микропроцессора.
8. Опишите принцип аппаратного управления ("жёсткой" логики).
9. Опишите принцип микропрограммного управления (гибкой логики), его достоинства и недостатки.
10. Расскажите о способах формирования сигналов управления в управляющих автоматах с "гибкой" логикой, достоинства и недостатки.

Лекция 8

Элементы архитектуры ЦСП TMS320C6x

Цель лекции: Дать определение данным. Изучить преобразование чисел из одной системы исчисления в другую. Научиться производить арифметические операции с числами в разных системах исчисления. Уяснить, как работают методы адресации операндов.

Компания Texas Instruments (TI) вывела на рынок процессоры с архитектурой VLIW (высоко параллельная архитектура с очень длинным – 32-разрядным, командным словом), предложив для нее термин VelosTI. 5 семейства (платформ) ЦСП: от TMS320C2x до TMS320C6x.

8.1. Данные

16-ричная система счисления – используется для компактной записи чисел в программах.

Алфавит: 10 арабских цифр от 0 до 9 и шесть латинских букв: A~10, B~11, C~12, D~13, E~14 и F~15.

Преобразование десятичных чисел в 16-ричные и обратно:

$$\begin{array}{r} \text{CF}_{16} \quad 207 \quad 16 \\ \hline 192 \quad 12 \\ \hline 15 \end{array}$$

$$\text{CF}_{16} = 15 \cdot 16^0 + 12 \cdot 16^1 = 15 + 192 = 207_{10}$$

Переход от 2-х чисел к 16-ричным: *Переход* от 16-ричных чисел к двоичным:

$$\begin{array}{c} 01001011 \\ \swarrow \searrow \\ 4\text{B}_{16} \end{array}$$

$$\begin{array}{c} \text{CF}_{16} \\ \swarrow \searrow \\ 11001111_2 \end{array}$$

Представление целых чисел.

Числа бывают без знаковые и знаковые.

В n-разрядной двоичной сетке для модуля без знаковых чисел отводятся все n разрядов. При этом диапазон чисел составляет от 0 до $2^n - 1$.

В случае знаковых чисел старший разряд отводится под знак (1 – «-», 0 – «+»), а остальные разряды – под модуль. Т.о., диапазон чисел составляет от $+(2^{n-1} - 1)$ до $-(2^{n-1} - 1)$.

В процессоре знаковые числа представляются в дополнительном коде.

Дополнительный код положительного числа, есть само число.

Дополнительный код отрицательного числа образуется по правилу: символ младшего разряда вычитается из числа, равного основанию системы счисления, а символы остальных разрядов – из числа, на 1 меньшего основания системы счисления.

Примеры при $n = 8$: $-5_{10} = -0000\ 0101_2$ в прямом двоичном коде;

$-5_{10} = 1111\ 1011_2$ в дополнительном коде.

$-7E_{16} = 82_{16}$ в дополнительном коде.

Арифметические операции над целыми числами.

1. **Сложение:**

1011_2	AB_{16}
$+ 1101_2$	$+ 05_{16}$
$\hline 11000_2$	$\hline B0_{16}$

а) производится поразрядно, начиная с младших разрядов;

б) если сумма S_i чисел i -го разряда превышает или равна ОСС, то в этот разряд результата записывается разность $(S_i - k \cdot \text{ОСС})$, а в следующий, более старший разряд (включая и знаковый), переносится k в виде дополнительного слагаемого. Здесь k – целая часть от деления S_i на ОСС.

Например, в случае двоичной системы счисления и $S_i = 5$, $k = [5/2] = 2$. Следовательно, в i -й разряд результата запишется $5 - k \cdot 2 = 5 - 4 = 1$, а в следующий разряд переносится (говорим «в уме») дополнительное слагаемое 2 ($k = 2$);

в) в случае знаковых чисел перенос из знакового разряда не производится (рис.8.1)

Примеры: числа без знака
Возможно переполнение

$$\begin{array}{r} 1011_2 \quad AB_{16} \\ + 1101_2 \quad + 05_{16} \\ \hline 11000_2 \quad B0_{16} \end{array}$$

знаковые числа
Переполнения нет

$$\begin{array}{r} 1011_2 \quad AB_{16} \\ + 0111_2 \quad + 05_{16} \\ \hline 0010_2 \quad B0_{16} \\ -5+7 \quad -55_{16}+5_{16} \end{array}$$

Рис.8.1 – Примеры

Результат операции над знаковыми числами представлен в прямом коде, если он положительный и в дополнительном коде, если отрицательный.

2. Вычитание заменяется сложением чисел, предварительно представленных в дополнительном коде.

Примеры при $n = 4$:

$$6_{16} - 3_{16} = 6_{16} + D_{16} = 3_{16} \text{ (перенос из знакового разряда не производится);}$$

$$3_{16} - 6_{16} = 3_{16} + A_{16} = D_{16} \text{ (дополнительный код числа -3);}$$

$$-6_{16} - 3_{16} = A_{16} + D_{16} = 17_{16} \text{ (переполнение разрядной сетки).}$$

3. Умножение для положительных чисел выполняется обычным образом, а для отрицательных – с промежуточным преобразованием в прямой код.

Примеры: а). Числа без знака (рис.8.2):

$$\begin{array}{r} \begin{array}{l} 5E \\ \times A2 \\ \hline BC \\ + 3AC \\ \hline 3B \ 7C \end{array} \quad \begin{array}{l} (E \cdot 2 = 28 \quad 28:16=1, \text{ остаток } 12; \\ (5 \cdot 2 = 10 \quad 10+1=11) \\ (E \cdot A = 140 \quad 140:16=8, \text{ остаток } 12; \\ (5 \cdot A = 50 \quad (50+8):16=3, \text{ остаток } 10; \end{array} \quad \begin{array}{l} \text{С пишем, 1 в «уме»} \\ \\ \text{С пишем, 8 в «уме»} \\ \text{А пишем, 3 в «уме»} \end{array} \end{array}$$

$$\text{б). Знаковые числа: } 52 \times AE = 52 \times (-52) = -1A44h = E5BC.$$

Рис.8.2 – Примеры

8.2. Методы адресации операндов

Адресация – обращение к операнду, указание на который содержится в команде.

Адресный код (A_K) – это информация об адресе операнда, содержащаяся в команде.

Исполнительный адрес (A_I) – это номер физической ячейки памяти, к которой производится обращение.

Первая группа адресаций устанавливает A_I по значению A_K .

Непосредственная адресация – операнд указывается в команде константой. Эта адресация используется для ввода исходных данных и при работе с различного рода константами.

Прямая адресация – A_I совпадает с A_K .

Регистровая адресация – в команде указывается имя регистра РОН процессора, в котором хранится операнд.

Косвенная адресация – A_K указывает имя регистра процессора, в котором находится A_I . Такой регистр называют *регистром адреса* (рис.8.3).

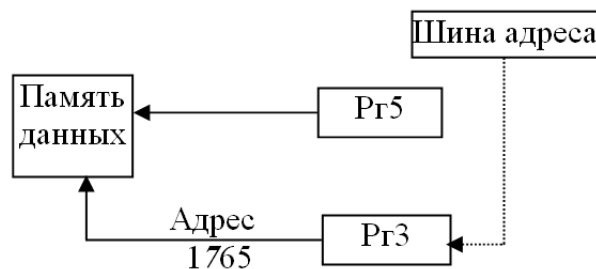


Рис.8.3 – Регистр адреса

Автоинкрементная (автодекрементная) адресация – в команде указывается имя регистра процессора, содержимое которого автоматически увеличивается (уменьшается) на 1.

Вторая группа адресаций вычисляет A_I по A_K . При этом A_I определяется алгебраической суммой A_K , называемого *базовым адресом (базой)*, и некоторого числа, называемого *смещением* (рис.8.4).

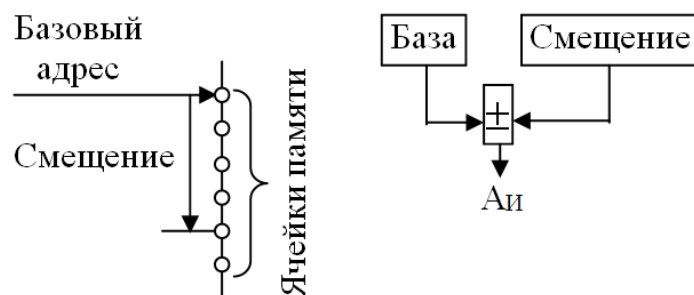


Рис.8.4 – Вторая группа адресации

«База» задает центр области ячеек памяти, а «смещение» – величину смещения относительно этого центра.

«База» содержится в регистре РОН, называемом регистром адреса или также базой.

2 метода – базирование и индексация. При базировании база не меняется, а при индексации – меняется и становится равной $A_{и}$.

Краткие итоги

Рассмотрены понятия системы исчисления, перевод из одной в другую, арифметические действия. Изучены методы адресации операндов.

Контрольные вопросы

1. Как представляются целые числа?
2. Как производится сложение?
3. Как делается умножение?
4. Как делается вычитание?
5. Дайте определение адресации, адресному коду и исполнительному адресу.
6. Что устанавливает первая группа адресации?
7. Для чего нужна вторая группа адресации?

Лекция 9

Структура ЦСП TMS320C6x

Цель лекции: Ознакомиться со структурой ЦСП TMS320C6x. Понять принцип работы регистровых пар. Процесс обработки команд.

Процессоры платформы 'C6x сделаны по новой архитектуре *VelocityTI* – высоко параллельная и детерминированная архитектура с очень длинным командным словом. Восемь независимых модулей (функциональных устройств) позволяют параллельно (одновременно) выполнять до восьми команд. Все команды содержат условия их выполнения, что позволяет сократить расходы производительности процессора на выполнение переходов и увеличить степень параллелизма обработки. Характерным для процессоров этой платформы является наличие аппаратного умножителя, позволяющего выполнять умножение двух чисел за один такт.

Три составные части: ядро (ЦПУ), две области памяти – память данных и память команд, внутренняя периферия (рис.9.1).

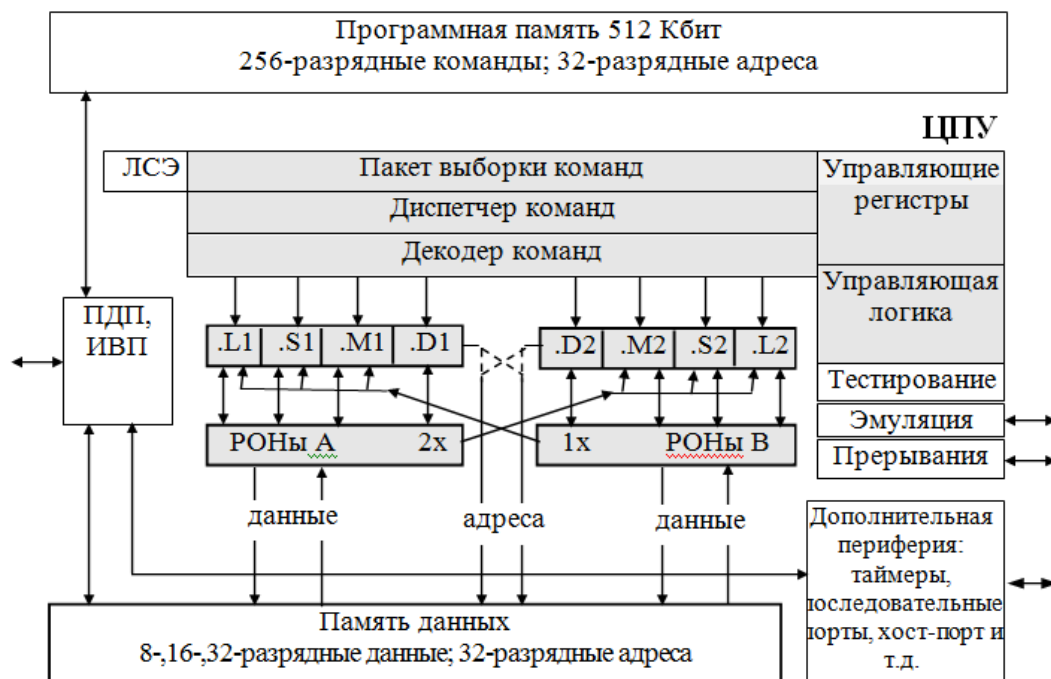


Рис.9.1 – Структура ЦСП TMS320C6x

Контроллер прямого доступа в память (ПДП) занимается передачей данных между областями памяти без участия ЦПУ. Кроме того, он используется для начальной загрузки программы в память процессора.

Интерфейс внешней памяти (ИВП) предназначен для обмена данными с внешней памятью и быстродействующими внешними устройствами.

Два 32-разрядных таймера используются для задания временных событий, реализации счетчиков, генерации импульсов, прерывания процессора. Процесс прерывания заключается в прерывании выполнения текущей программы и переходе к выполнению некоторой другой программы. По завершении выполнения этой другой программы процессор возвращается к прерванной программе и продолжает ее выполнение.

Логика снижения энергопотребления (ЛСЭ) включает один из трех возможных «спящих» режимов процессора. В первом режиме тактовые импульсы снимаются только с ядра процессора, во втором – и с периферии, размещенной на кристалле, а в третьем режиме тактовая частота снимается практически со всего кристалла.

ЦПУ включает в себя 8 независимых функциональных устройств (модулей) – два умножителя (устройства .М) и 6 арифметико-логических устройств (устройства .L, .S и .D). Арифметико-логическое устройство (АЛУ) – это КЦУ, реализующее арифметические (кроме умножения и деления) и логические операции.

Каждая четверка модулей связана с набором регистров общего назначения (РОН), создавая тем самым разделение ядра на сторону А и сторону В. РОН программно доступны, т. е. с помощью соответствующих команд программист может управлять их содержимым. Следовательно, они могут быть использованы для самых различных целей – поддержки различных типов адресации памяти данных, хранения промежуточных

результатов вычислений и в качестве источников операндов. РОН каждой стороны представляет собой 16 32-разрядных регистров, связанных с памятью данных двумя шинами – шиной загружаемых из внутренней памяти данных и шиной загружаемых в эту память данных.

Два АЛУ – .D1 и .D2, используются только для формирования исполнительного адреса ячейки памяти данных. При этом шины адреса, управляемые D-устройствами, позволяют использовать адрес, сформированный в РОН одной стороны, для операций с данными в РОН другой стороны. Данные из функциональных устройств сначала помещаются в РОН, а затем по адресам, формируемым D-устройствами, идет обмен с памятью данных. При этом возможна одновременная выборка из памяти данных до 64 разрядов по двум подаваемым адресам.

Один из модулей .L, .S или .M каждой стороны через соответствующую перекрестную шину (2х или 1х) имеет доступ к РОН противоположной стороны, причем только как к источникам операнда.

В одном и том же такте к РОН своей стороны возможен доступ всех модулей этой стороны одновременно. Однако к РОН противоположной стороны может иметь только один модуль данной стороны.

Поскольку модули независимы, в каждом такте процессор может выдавать им до восьми 32-разрядных команд, которые могут выполняться параллельно (одновременно), последовательно или параллельно-последовательно. Блок команд, которые выполняются параллельно (в одном такте), называется выполняемым пакетом. Блок команд, содержащий до 8 выполняемых пакетов, называется пакетом выборки. Таким образом, пакет выборки состоит из восьми выполняемых пакетов, если 8 команд выполняются последовательно, из одного выполняемого пакета, если 8 команд выполняются параллельно, и из двух – семи выполняемых пакетов, если 8 команд выполняются параллельно-последовательно.

Процесс обработки команд в ядре начинается загрузкой из программной памяти пакета выборки. Диспетчер команд распределяет каждую из 32-разрядных команд пакета на свой модуль для исполнения. Очередной выполняемый пакет размещается для исполнения в функциональных устройствах в каждом такте. Пакет же выборки из программной памяти не загружается до окончания выполнения текущего пакета выборки.

ПДП – передача данных между областями памяти и начальная загрузка программы в программную память.

ИВП – обмен данными между памятью данных и внешней памятью, внешней памятью и внешней периферией, памятью данных и внешней периферией.

Таймеры – задание временных событий, генерация импульсов, реализация счетчиков, прерывание процессора.

ЛСЭ – включает «спящие» режимы. Режим 1 – тактирование снимается только с ядра, режим 2 – тактирование снимается с ядра и периферии, режим 3 – тактирование снимается практически со всего кристалла.

Имена регистров РОН: A0, ..., A15, B0, ..., B15. Регистровые пары представлены в таблице 9.1.

Таблица 9.1

Регистровые пары

Сторона А		Сторона В	
A1:A0	A9:A8	B1:B0	B9:B8
A3:A2	A11:A10	B3:B2	B11:B10
A5:A4	A13:A12	B5:B4	B13:B12
A7:A6	A15:A14	B7:B6	B15:B14

Ограничения ресурсов:

- только один модуль (.L, .S или .M) и только один операнд можно взять с противоположной стороны РОН;

- результат выполнения операции модулем всегда размещается в регистре РОН своей стороны;

- в одном и том же такте к РОН одной стороны возможен доступ одновременно всех модулей этой стороны;

- для обмена данными между РОН одной стороны и памятью данных можно использовать модуль .D противоположной стороны.

Процесс обработки команд в ядре включает в себя:

- пакет выборки загружается в программную память;
- диспетчер команд формирует последовательность выполняемых пакетов;

- для каждой команды выполняемого пакета декодер выделяет код операции и передает его на соответствующий модуль.

Краткие итоги

В лекции рассмотрена структура ЦСП TMS320C6x. Даны определения ПДП, ЛЭС, ИВП. Дано объяснение, что такое регистровые пары. Описан процесс обработки команд в ядре.

Контрольные вопросы

1. Перечислите три составных части структуры ЦСП TMS320C6x
2. Опишите ядро ЦПУ
3. Дайте определение ПДП, ИВП и ЛЭС.
4. Перечислите имена Регистров РОН по парно.
5. Какие ограничения ресурсов существует?
6. Что включает в себя обработка команды в ядре?

Лекция 10

Структура командной строки ассемблера, вопросы особенности адресации и команд.

Цель лекции: Ознакомиться со структурой командной строки. Уяснить ее оформление, функциональные свойства. Понять, как отделяются поля, какие существуют запреты.

Программа на языке ассемблера должна быть оформлена в виде текстового файла. Любая командная строка ассемблерной программы может включать вплоть до семи полей в следующей последовательности: метка, параллельные полосы, условие, команда, функциональное устройство, операнды и комментарий. Поля отделяются друг от друга одним или более пробелами. Внутри поля пробелы запрещены (рис.10.1):

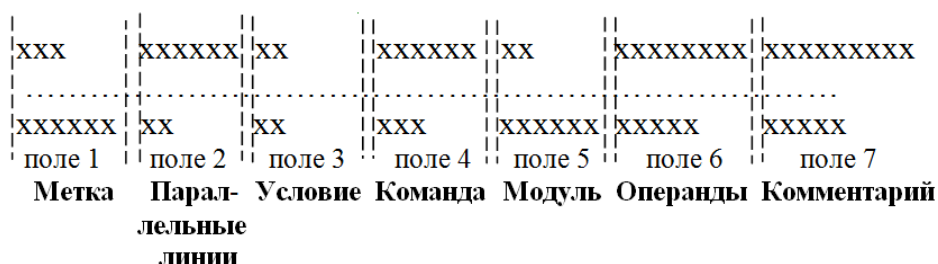


Рис.10.1 – Ассемблерные предложения

Пробелы внутри каждого из 6 первых полей запрещены.

Поле метки

правильная запись **L...: _G...:**

неправильная запись **1...: _1...: _:**

Поле параллельных линий ||

Такт (выполняемый пакет)	Команды		
1	A		
2	B		
3	C	D	E

Команда A

Команда B

Команда C переход

|| Команда D ←

|| Команда E

Поле условия

Регистры использующиеся в поле условия: A1, A2, B0, B1 и B2

Принцип выполнения условных команд:

Условие	Команда выполняется, если
[A1]	содержимое A1 не нулевое
[!A1]	содержимое A1 равно нулю

Поле команды

Мнемоника директивы или команды.

Пример директивы ассемблера: .set (на машинный язык не переводятся).

Пример команды ассемблера: add (переводятся на машинный язык).

Поле модуля

Имя конкретного функционального устройства, например **.D1** или **.L1X**, если один операнд берется с противоположной стороны.

Функциональный тип устройства, например **.M**.

Поле операндов

Имена регистров РОН, содержащие данные.

Адресация операндов.

Данные в числовой форме:

1. *Константы*:

—двоичное целое, например 01**b**, 0100**B**;

—восьмеричное целое, например 236**q**, 10**Q**;

—десятичное целое:

знаковые числа от -2 147 483 648 до 2 147 483 647,

числа без знака от 0 до 4 294 967 295;

—шестнадцатеричное целое, например 9A**h**, 5D**h**.

Требование: число должно начинаться с цифры. Так, при вводе A5h следует указывать как 0A5h;

Напоминание: если не указать систему счисления число по умолчанию считается десятичным. Это справедливо для всех языков программирования.

–символ, например ‘А’ (символ А), ‘ ’ (пробел), ‘’’ (один апостроф);

–символьная строка, например “LAN ‘’’С’’” (план "С").

2. Выражение.

Поле комментария

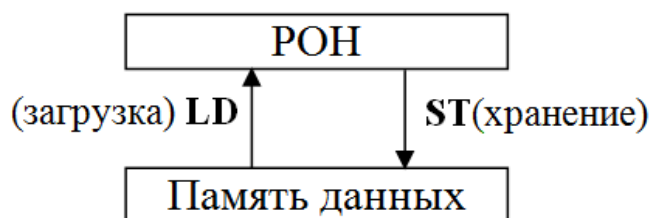
Начинается символом:

–«;», если размещается с любого (кроме первого) столбца ассемблерной строки;

–«*»,если размещается с первого столбца ассемблерной строки.

10.1 Основные команды процессора ‘Сбх для целых чисел

Команды загрузки и хранения



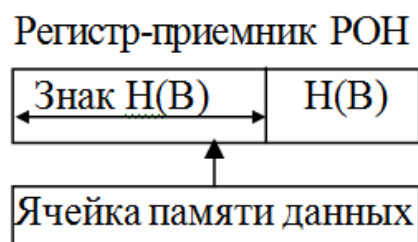
Обе команды дополняются третьей буквой определяющей объем пересылаемых данных:

W – слово (32-разрядное двоичное число)

H – полуслово (16 младших двоичных разрядов)

B – байт (8 младших двоичных разрядов)

Особенности команд загрузки:



LDH и **LDB** – расширение знаком полуслова или байта, соответственно;

LDHU и **LDBU** – расширение нулем.

Примеры для mem = 1005 B071h:

после выполнения LDH – FFFF B071h (B=1011),

после выполнения LDB – 0000 0071h (7=0111),

после выполнения LDHU – 0000 B071h.

Формат поля операндов команд загрузки/хранения:

(для LD) *<A_к>,<регистр-приемник РОН>

(для ST) <регистр-источник РОН>,*<A_к>

Обратите внимание, что в команде всегда пишется: откуда и куда осуществляется пересылка данных. Символ «*» является признаком того что регистр используется в качестве адресного, то есть в нем содержится адресный код ячейки внешней памяти.

10.2 Особенности методов базирования и индексации в Сбх

Особенности методов базирования и индексации в Сбх состоит в следующем:

–базирование указывается одним знаком операции со смещением и всегда перед именем базы, а индексация – двумя знаками;

–смещение указывается в квадратных скобках и представляется либо константой, либо содержимым регистра РОН, например: *-A7[A5], *++A7[5];

–второй смысл третьей буквы мнемоники команды: W означает предварительное учетверение смещения, H – удваивание, B – неизменность смещения;

–если величина смещения превышает 5 двоичных разрядов, в качестве базы может использоваться только регистр B14 или B15. При этом в методе базирования вычитание не поддерживается. Выходом из данной ситуации является запись смещения в другой регистр, тогда база может быть любой.

Команды поддерживают три вида адресации.

Косвенная



например:
LDx *A7,A4
STx A4,*A7
АИ = содержимому A7

Базирование



например
LDW *+A7[5],A4
 ↘
АИ = A7+4·5

STH A4,*-A7[5]
 ↘
АИ = A7- 2·5

Индексация



прединдексация

например

LDB *++A7[5],A4

$$A_{И} = A7 + 1 \cdot 5$$

STH A4,*--A7[5]

$$A_{И} = A7 - 2 \cdot 5$$



постиндексация

например

STx A4,*A7++[5]

$A_{И} = \text{содержимому } A7$

LDx *A7--[5],A4

$A_{И} = \text{содержимому } A7$

Примеры команд загрузки и хранения:

1. **LDW .D1 *A10,B1** (косвенная)

Перед командой после 4 тактов

B1 xxxx xxxhh 21F3 1996h

A10 0000 0100h 0000 0100h

Mem 100h 21F3 1996h 21F3 1996h

2. **LDB .D1 *-A5[4],A7** (базирование)

Перед командой после 4 тактов

A5 0000 0204h 0000 0204h

A7 xxxx xxxhh FFFF FFE1h

Mem 200h E1h E1h

3. **LDH .D1 *++A4[A1],A8** (преиндекс.)

Перед командой после 4 тактов

A1 0000 0002h 0000 0002h

A4 0000 0020h 0000 0024h

A8 xxxx xxxhh FFFF A21Fh

Mem 24h A21Fh A21Fh

4. **STH .D1 A1,*A10--[A11] (постиндекс.)**

Перед командой после 1 такта

A1 9A32 7634h 9A32 7634h

A10 0000 0100h 0000 00F8h

A11 0000 0004h 0000 0004h

Mem 100h B000 800Ah 0000 7634h

5. **LDB .D2 *+B14[36],B1 (конст. >25-1)**

Перед командой после 4 тактов

B1 xxxx xxxhh 0000 0012h

B14 0000 0100h 0000 0100h

Mem 124h 12h 12h

6. **LDW .D1 *A4++[1],A6 (постиндекс.)**

Перед командой после 4 тактов

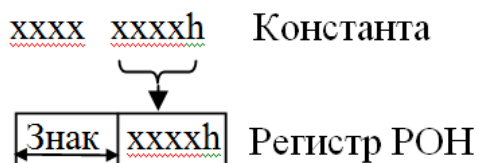
A4 0000 0100h 0000 0104h

A6 xxxx xxxhh 0798 F25Ah

Mem 100h 0798 F25Ah 0798 F25Ah

10.3 Команды ввода исходных данных

MVK (с расширением знаком)



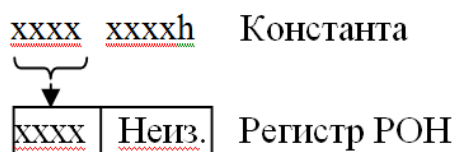
MVK .S1 293,A1

MVK .S1 0A95AF12h,A1

перед командой после 1 такта перед командой после 1 такта

A1 xxxx xxxhh 0000 0125h A1 xxxx xxxhh FFFF AF12h

MVKH



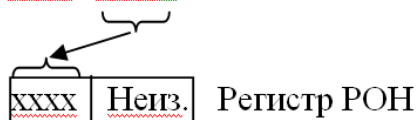
MVKN .S1 0A3291h,A1

перед командой после 1 такта

A1 xxxx 7634h 000A 7634h

MVKLN

xxxx xxxhh Константа



MVKLN .S1 7A8h,A1

перед командой после 1 такта

A1 xxxx F25Ah 07A8 F25Ah

10.4 Арифметические команды

ABS – вычисление абсолютной величины.

Особенность команды: результат представляется в дополнительном коде.

ABS .L1 A1,A5 (*отрицательное*)

перед командой после 1 такта

A1 8000 4E3Dh 8000 4E3Dh

A5 xxxx xxxhh 7FFF B1C3h

ABS .L1 A1,A5 (*положительное*)

перед командой после 1 такта

A1 3FF6 0010h 3FF6 0010h

A5 xxxx xxxhh 3FF6 0010h

ADDK – сложение со знаковой константой

ADDK .S1 15401,A1 (15401₁₀ = 3C29h)

перед командой после 1 такта

A1 0021 37E1h 0021 740Ah

ADD2 – сложение младших и старших полуслов

ADD2 .S1X A1,B1,A2

	перед командой	после 1 такта
A1	0021 37E1h	0021 37E1h
B1	039A E4B8h	039A E4B8h
A2	xxxx xxxhh	03BB 1C99h

ADD – сложение знаковых целых

Особенности команды:

–полагается, что операнды уже представлены в дополнительном коде;

–только первый операндом может быть знаковой константой, но длиной не более 5 двоичных разрядов;

–только второй операнд может быть длинным (40-разрядным);

–переполнение исключено.

ADD .L2X A1,B1,B2

	перед командой	после 1 такта
A1	0000 325Ah	0000 325Ah
B1	FFFF FF12h	FFFF FF12h
B2	xxxx xxxhh	0000 316Ch

ADDU – сложение беззнаковых целых

Особенность команды: приемником результата всегда является регистровая пара, поскольку возможно переполнение

ADDU .L1 A1,A2,A5:A4

	перед командой	после 1 такта
A1	0000 325Ah	0000 325Ah
A2	FFFF FF12h	FFFF FF12h
A5:A4	xxxx xxxhh	0000 0001:0000 316Ch

ADDU .L1 A1,A3:A2,A5:A4

перед командой	после 1 такта
----------------	---------------

A1	0000 325Ah	0000 325Ah
A3:A2	0000 00FF:FFFF FF12h	0000 00FF:FFFF FF12h
A5:A4	0000 0000: 0000 0000h	0000 0000: 0000 316Ch

нет доступа

SUB – вычитание знаковых целых

Особенности команды:

–если уменьшаемой положительное, вычитаемое предварительно преобразуется в дополнительный код, после чего производится сложение. Здесь приемником результата является регистр.

–если уменьшаемое отрицательное, то после преобразования вычитаемого в дополнительный код оба операнда расширяются знаком до 40-разрядных чисел, затем производится сложение.

SUB .L1 A1,A2,A3

	перед командой	после 1 такта
A1	0000 325Ah	0000 325Ah
A2	FFFF FF12h	0000 00EEh
A3	xxxx xxxxh	0000 3348h

SUB .L1 A1,A2,A5:A4

	перед командой	после 1 такта
A1	FFFF CDA6h	FF:FFFF CDA6h
A2	0000 00EEh	FF:FFFF FF12h
A5:A4	0000 0000h:0000 0000h	0000 00FF:FFFF CCB8h

SUBU – вычитание беззнаковых целых

Особенность команды: 32-разрядное вычитаемое расширяется знаком до 40-разрядного, затем преобразуется в дополнительный код, после чего производится сложение.

SUBU .L1 A1,A2,A5:A4

	перед командой	после 1 такта
A1	0000 325Ah	00:0000 325Ah

A2	FFFF FF12h	00:0000 00EEh
A5:A4	0000 0000:0000 0000h	0000 0000:0000 3348h
SUBU .L1 A1,A2,A5:A4		
	перед командой	после 1 такта
A1	FFFF CDA6h	00:FFFF CDA6h
A2	0000 00EEh	FF:FFFF FF12h
A5:A4	0000 0000h:0000 0000h	0000 0000:FFFF CCB8h

MPY – умножение младших знаковых полуслов

Особенность команды: только первый операнд может быть знаковой константой, но длиной не более 5 двоичных разрядов.

MPY .M1 A1,A2,A3

перед командой	после 2 такта	Комментарий:
A1 0000 0123h	0000 0123h	(FA81 – доп. код числа -057Fh)
A2 01E0 FA81h	01E0 FA81h	(-057Fh×0123h = -0006 3F5Dh)
A3 xxxx xxxxh	FFF9 C0A3h	

MPYU – умножение младших беззнаковых полуслов

MPYU .M1 A1,A2,A3

перед командой	после 2 такта	Комментарий:
A1 0000 0123h	0000 0123h	(FA81h берется как положительное)
A2 0F12 FA81h	0F12 FA81h	
A3 xxxx xxxxh	011C C0A3h	(результат без знака)

MPYUS (первый операнд – без знака, второй – знаковый)

MPYUS .M1 A1,A2,A3

перед командой	после 2 такта	Комментарий:
A1 1234 FFA1h	1234 FFA1h	(FA81h берется как положительное)
A2 1234 FFA1h	1234 FFA1h	(FA81h – доп. код числа -5Fh)
A3 xxxx xxxxh	FFA1 2341h	(FA81h×-5Fh = -005E DBFh)

MPYSU (первый операнд – знаковый, второй – без знака)

MPYSU .M1 13,A1,A2

	перед командой	после 2 такта
A1	3497 FFF3h	3497 FFF3h
A2	xxxx xxxh	000C FF57h

Краткие итоги

В лекции были рассмотрены правила построения ассемблерной строки. Перечислены поля входящие в строку и особенности их формирования. Освещены вопросы адресации и особенности команд загрузки/хранения, а также математических команд. Рассмотрены вопросы базирования и индексации.

Контрольные вопросы

1. Как строится ассемблерное предложение?
2. Перечислите все поля, которые входят в ассемблерное предложение.
3. Как происходит адресация операндов?
4. Как задаются данные в числовой форме?
5. Как задаются данные в выражении?
6. Перечислите основные команды процессора 'Сбх для целых чисел.
7. Назовите особенности команды загрузки.
8. Перечислите особенности методов базирования и индексации.
9. Сколько команд поддерживают адресацию, дайте их характеристику.
10. Какие команды отвечают за хранение данных?
11. Какие команды отвечают за передачу данных?
12. Какие команды отвечают за ввод исходных данных?
13. Какие команды отвечают за сложение и их особенности?
14. Какие команды отвечают за вычитание и их особенности?

Лекция 11

Организация стандартных алгоритмических структур

Цель лекции: Ознакомиться с алгоритмами процессов обработки информации в МПС и как они представляются в графическом виде. Из чего состоит алфавит языка SDL

Наиболее часто алгоритмы процессов обработки информации в МПС представляют графически в виде схем, используя язык спецификаций и описаний SDL .

Алфавит языка SDL состоит из набора графических символов, которые подразделяются на 2 группы.

Первая группа используется для представления диаграмм взаимодействия, а **вторая** – диаграмм процесса схем алгоритмов.

Приведём основные символы языка SDL (рис.11.1).

На диаграмме взаимодействия МПС представляется набором функциональных БЛОКОВ, соединённых между собой и с окружающей средой однонаправленными КАНАЛАМИ.

Каждый БЛОК состоит из одного или нескольких ПРОЦЕССОВ.

SDL-ПРОЦЕССУ можно поставить в соответствие произвольный элемент или совокупность элементов МПС, выполняющих некоторый набор логических функций, имеющих конечную память внутренних состояний и взаимодействующих с другими элементами МПС посредством дискретного и конечного множества входов и выходов.

В СПИСКАХ СИГНАЛОВ перечисляются сигналы, передаваемые между ПРОЦЕССАМИ в пределах одного БЛОКА, а также сигналы, передаваемые по КАНАЛУ в направлении некоторого БЛОКА или от него.







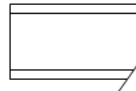
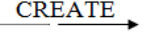
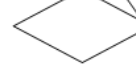






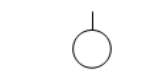

Символы диаграмм взаимодействия		Символы диаграмм процесса	
Название символа	Графическое представление	Название символа	Графическое представление
Блок		Начало	
Процесс		Состояние	
Список сигналов		Задача	
Среда	ENVIRONMENT	Запрос	
Создать		Решение	
Канал или поток сигналов		Запоминание	
Окончание		Вход	
Выход		Соединительная линия	
Входной соединитель		Выходной соединитель	

Рис. 11.1 – Основные символы языка SDL

На **диаграмме процессов** каждый ПРОЦЕСС определяется в терминах теории конечных автоматов и может находиться либо в фазе СОСТОЯНИЕ, либо в фазе ПЕРЕХОД.

В фазе СОСТОЯНИЕ действие ПРОЦЕССА приостановлено в ожидании поступления некоторого сигнала ВХОД.

Этот сигнал переводит ПРОЦЕСС в фазу ПЕРЕХОД, которая характеризуется последовательностью действий, определяемых информацией в сигнале ВХОД.

По окончании фазы ПЕРЕХОД ПРОЦЕСС вновь оказывается в фазе СОСТОЯНИЕ и вырабатывает сигнал ВЫХОД, являющийся в свою очередь сигналом ВХОД для другой фазы СОСТОЯНИЕ.

Среди последовательности действий фазы ПЕРЕХОД следует

различать:

–РЕШЕНИЕ – действие, связанное с выбором одного из нескольких возможных путей продолжения ПРОЦЕССА;

–ЗАДАЧА – действие, не являющееся ни РЕШЕНИЕМ, ни ВЫХОДОМ;

–ЗАПОМИНАНИЕ – действие, выражающееся в задержке одновременно поступивших СИГНАЛОВ с целью их последующей последовательной обработки.

Основные положения теории конечных автоматов, на базе которой строится язык SDL, налагают следующие ограничения на порядок следования графических символов:

1.Первым символом диаграммы процесса является символ НАЧАЛО, указывающий начальную точку вновь созданного ПРОЦЕССА.

Далее ПРОЦЕСС пребывает либо в фазе СОСТОЯНИЕ, либо в фазе ПЕРЕХОД, перемещаясь по направлению к очередной фазе СОСТОЯНИЕ.

2.За символом СОСТОЯНИЕ может следовать один или несколько символов ВХОД или ЗАПОМИНАНИЕ.

3.Каждому символу ВХОД или ЗАПОМИНАНИЕ должен предшествовать один символ СОСТОЯНИЕ.

4.За символом ВХОД должен следовать строго один символ, за исключением символов ВХОД и ЗАПОМИНАНИЕ.

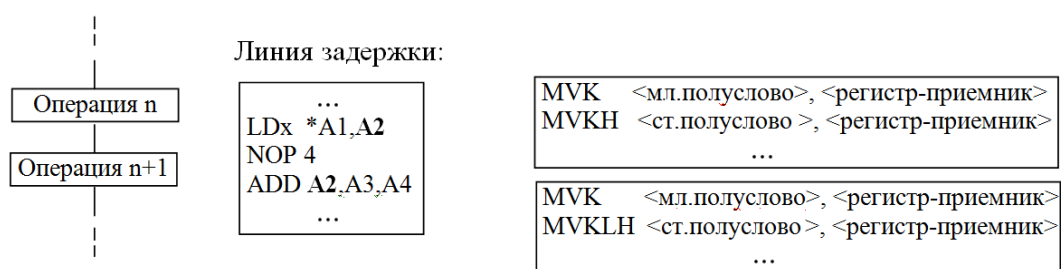
5.За символом ЗАДАЧА или ВЫХОД должен следовать строго один символ, который не может быть символом ВХОД или ЗАПОМИНАНИЕ.

6.За символом РЕШЕНИЕ должны следовать не менее двух символов, которые не могут быть символами ВХОД или ЗАПОМИНАНИЕ.

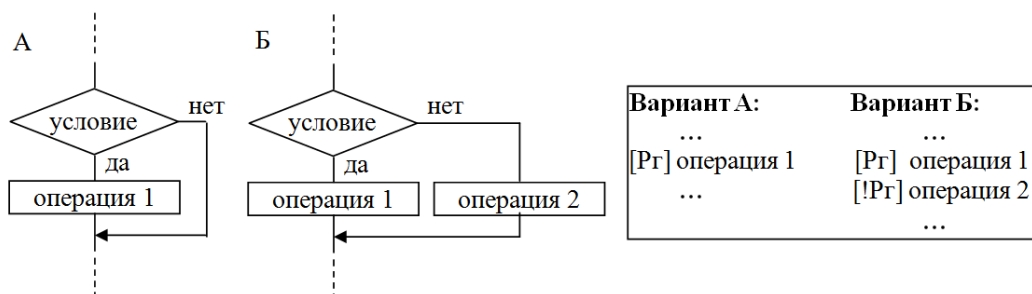
За символом ЗАПОМИНАНИЕ не могут следовать никакие символы.

Основные применяемые алгоритмы это линейные, ветвление и цикл.

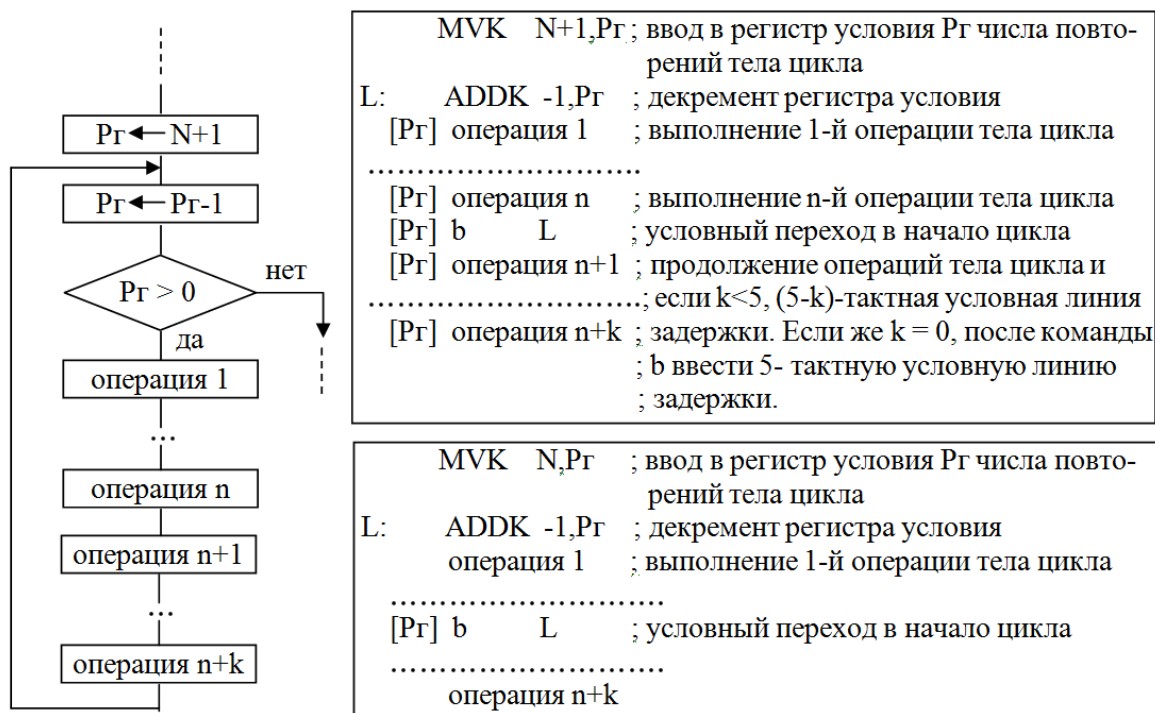
Линейные

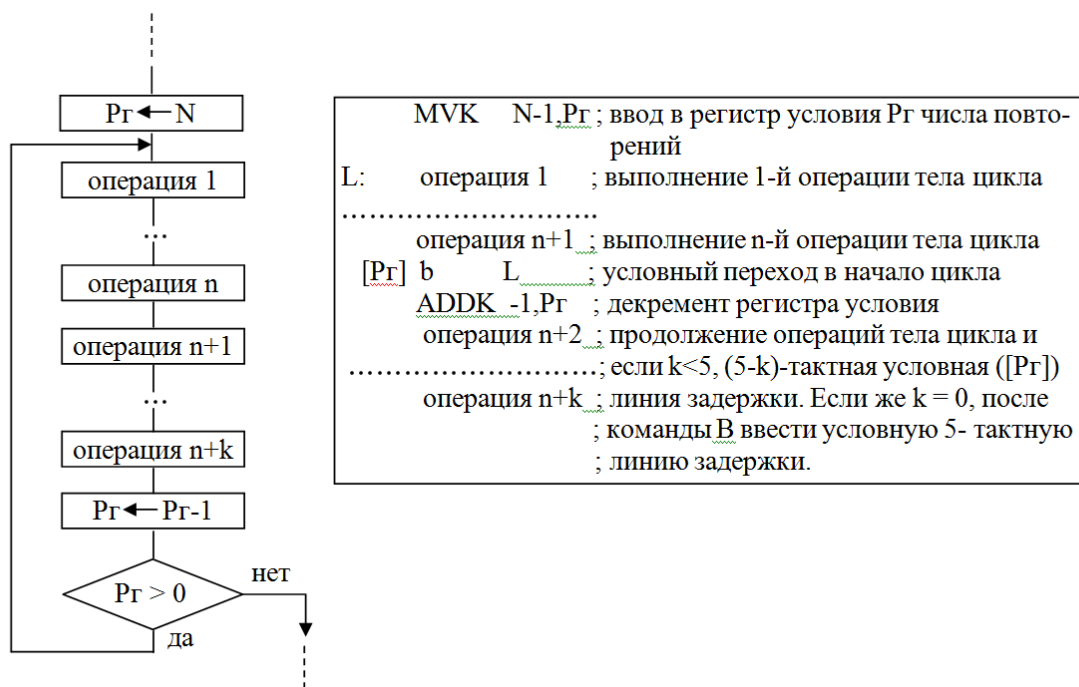


Ветвление



Цикл





Краткие итоги

В лекции были рассмотрены вопросы, относящиеся к организации стандартных алгоритмических структур. Правила использования алфавита языка SDL. Рассмотрен порядок следования графических символов.

Контрольные вопросы

1. Для чего используется первая группа алфавита языка SDL?
2. Для чего используется вторая группа языка SDL?
3. Приведите примеры основных символов языка SDL?
4. Как в диаграмме процессов определяется «ПРОЦЕС»?
5. Каков порядок следования графических символов?
6. Опишите линейный алгоритм.
7. Опишите ветвление.
8. Опишите цикл.

Лабораторные работы

Блок лабораторных работ предназначен для закрепления материала и развития навыков программирования на языке Ассемблера.

Состоит из 11 лаб. работ

Содержание отчета

1. Название лабораторной работы.
2. Код группы, фамилия и инициалы студента.
3. Формулировка индивидуальных заданий лабораторной работы.
4. Ход выполнения лаб. работы, цель, схемы, выводы.

В лабораторных работах по ассемблеру необходимо

1. «Блок-схема алгоритма решения задачи».
2. Таблица, содержащая структурированную программу, каждая командная строка которой сопровождается прогнозом содержимого используемых регистров РОН, а также указанием номера и содержимого используемой ячейки памяти данных (ЯПД) процессора в 16-ричной системе счисления.

Командная строка	Регистры РОН командной строки		ЯПД процессора, используемая в командной строке	
	Имя	Прогноз содержимого, Нех	Номер, Нех	Содержимое, Нех

Данная таблица предъявляется преподавателю до прогона программы с целью выявления возможных методических ошибок и получения указаний по адаптации программы к особенностям симулятора команд.

Исправленные в процессе отладки фрагменты исходной программы, заносятся в таблицу дополнительными строками. По завершении отладки окончательная таблица вновь предъявляется преподавателю с устными пояснениями исправлений.

Лабораторная работа 1

Исследование логических элементов

Логические переменные – как функция, так и аргументы, могут принимать только два значения. Обычно их обозначают символами 0 и 1, что в вычислительной технике соответствует низкому и высокому уровню напряжения сигнала соответственно

Цель работы Исследование логических элементов с помощью моделирования в программе Multisim

Подготовка к работе

По указанной выше литературе изучить:

- Понятие функции алгебры логики
- Закон функционирования ЛЭ
- Элементарные ФАЛ одного аргумента

Задание. Исследование логических элементов

Собрать схему, показанную на рис. 1.1. С помощью подобной схемы исследовать два логических элемента, типы которых выбрать из таблицы 1.1 в соответствии со своим вариантом.

Таблица 1.1

Варианты задания

Вариант	Логические элементы	Вариант	Логические элементы
1	ИЛИ-НЕ, ИСКЛ. ИЛИ	9	И, ИЛИ
2	ИЛИ-НЕ, И	10	ИЛИ, И-НЕ
3	ИЛИ-НЕ, РАВНОЗНАЧН.	11	ИЛИ, РАВНОЗНАЧНОСТЬ.
4	ИЛИ-НЕ, И-НЕ	12	ИЛИ, ИСКЛ. ИЛИ
5	ИЛИ-НЕ, ИЛИ	13	И-НЕ, РАВН.
6	И, ИСКЛ. ИЛИ	14	ИСКЛ. ИЛИ, И-НЕ
7	И, РАВНОЗНАЧНОСТЬ.	15	ИЛИ-НЕ, ИЛИ
8	И, И-НЕ	16	ИСКЛ. ИЛИ, РАВНОЗНАЧ.

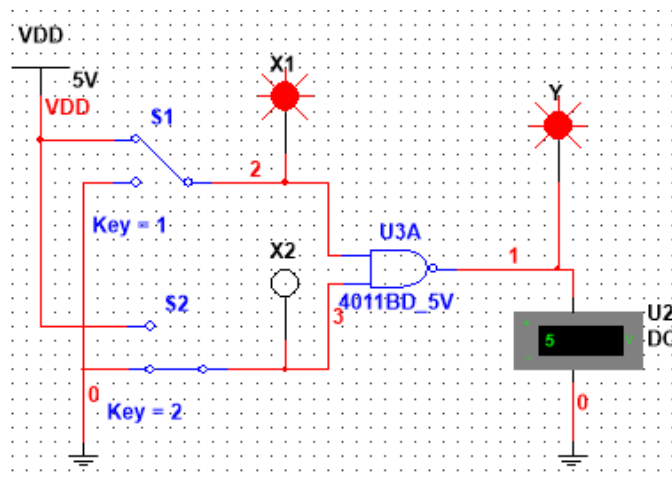


Рис.1.1 – Пример построения схемы

Результаты исследования логических элементов занести в отчет в виде таблицы 1.2. При выполнении задания нужно обратить внимание на то, что логическая единица соответствует высокому напряжению на выходе элемента, а логический ноль - низкому уровню напряжения.

Таблица 1.2

x_2	x_1	y	V, В
0	0		
0	1		
1	0		
1	1		

Контрольные вопросы

1. Составьте таблицу истинности логического элемента И.
2. Составьте таблицу истинности логического элемента ИЛИ.
3. Составьте таблицу истинности логического элемента ИЛИ-НЕ.
4. Составьте таблицу истинности логического элемента ИСКЛЮЧАЮЩЕЕ ИЛИ.
5. Составьте таблицу истинности логического элемента И-НЕ.
6. Запишите аксиомы алгебры логики.
7. Запишите тождества алгебры логики.
8. Запишите законы алгебры логики.
9. Перечислите способы задания логических функций.

Лабораторная работа 2

Моделирование работы комбинационных цифровых устройств

Отражение двоичных наборов физических значений разрядов. Эта терминология, которая применяется в комбинированных цифровых устройствах. Описывается последовательность синтеза и способы задания КЦУ.

Цель работы Исследование комбинированных цифровых устройств с помощью моделирования в программе Multisim

Подготовка к работе

По указанной выше литературе изучить:

- Понятие функции КЦУ.
- Способы задания КЦУ.
- Отличия работы сумматора и полусумматора.

Задание 1. Исследование полусумматора

Собрать схему, показанную на рис. 2.1. По результатам исследования полусумматора заполнить таблицу 2.1.

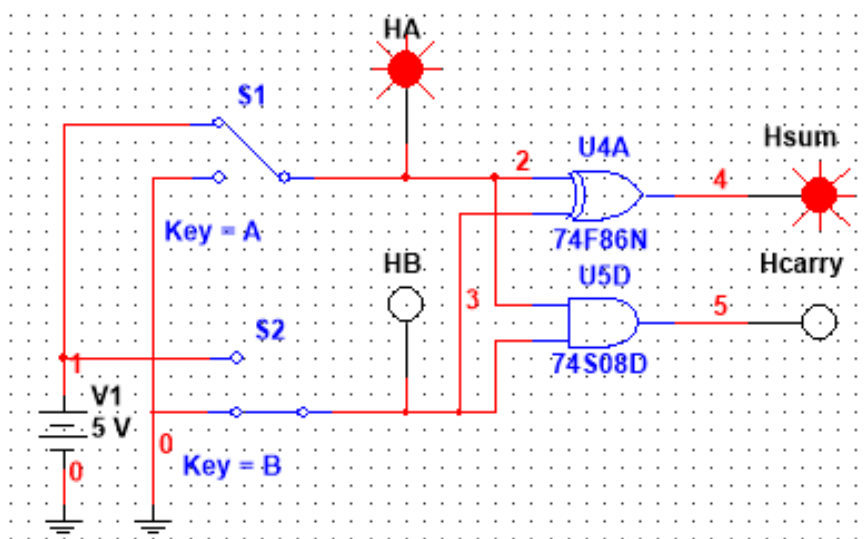


Рис. 2.1 – Пример схемы полусумматора

Таблица 2.1

A	B	Сумма (Σ)	Перенос (C_o)
0	0		
0	1		
1	0		
1	1		

Задание 2. Исследование полного сумматора

Собрать схему, показанную на рис. 2.2. По результатам исследования сумматора заполнить таблицу 2.2.

Таблица 2.2.

Вход			Выход	
A	B	Перенос (C_i)	Сумма (Σ)	Перенос (C_o)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

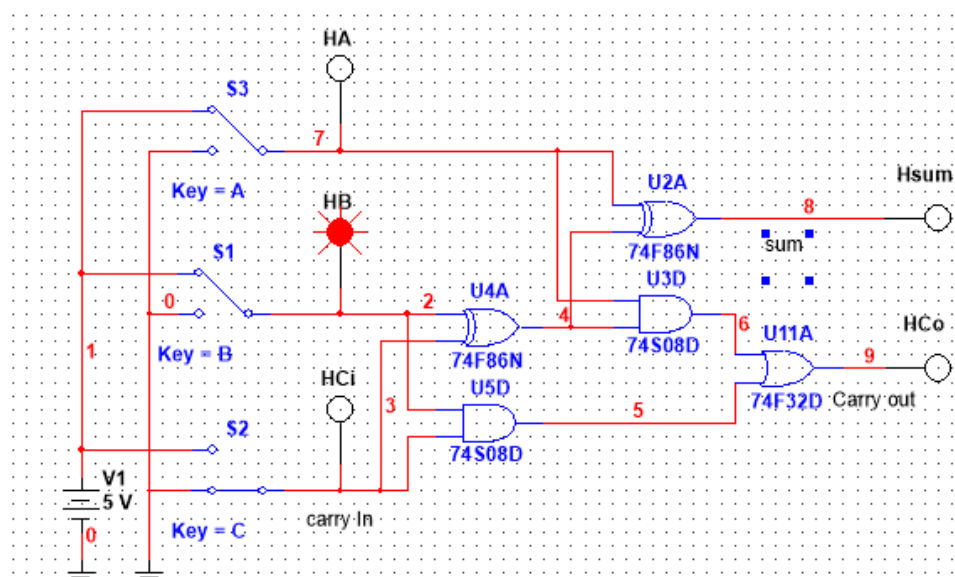


Рис. 2.2 – Пример построения сумматора

Задание 3. Обработка результатов исследований

Записать логические функции в СДНФ по результатам выполнения заданий 1, 2 (полусумматор и сумматор).

Методические указания

Следует запомнить, что светящийся индикатор сигнализирует о наличии логической единицы в данном узле схемы, а потухший индикатор – о наличии логического нуля.

При выполнении заданий 2 и 3 входные сигналы целесообразно обозначить символами А и В. На схемах сумматоров сигналы переноса обозначены символом С (от английского слова Carry).

Контрольные вопросы

1. Чем отличаются полусумматор и полный сумматор?
2. Дайте определение комбинационного цифрового устройства.
3. Как записывается логическая функция в СДНФ?
4. Как записывается логическая функция в СКНФ?
5. Полусумматор: определение, принцип работы и условное обозначение.
6. Полный одноразрядный сумматор: определение, принцип работы и условное обозначение.
7. Способ построения многоразрядных сумматоров.

Лабораторная работа 3

Шифраторы и дешифраторы

На входы типовых КЦУ могут подаваться два вида сигналов – информационные сигналы и сигналы управления. Информационные сигналы отображают обрабатываемую информацию, а сигналы управления выполняют одну или несколько из следующих функций

Цель работы Исследовать работу шифратора и дешифратора уяснить разницу в их функционировании.

Подготовка к работе

- Принцип работы и построения типовых КЦУ
- Основное назначение дешифраторов
- Основное назначение шифраторов

Задание 1. Исследование шифратора

Собрать схему, показанную на рис. 3.1. Входные сигналы задать в соответствии с рис. 3.1. Исследования проводить в пошаговом режиме. Зарисовать временные диаграммы. Объяснить полученные результаты.

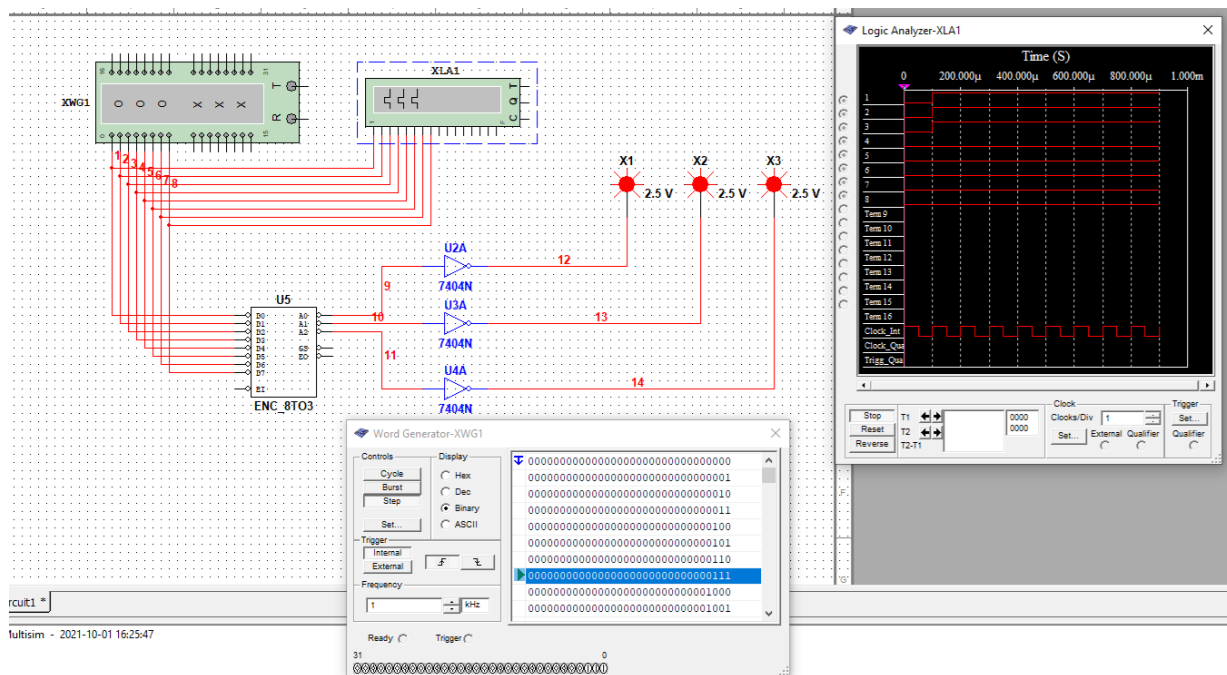


Рис. 3.1 – Исследование шифратора

Задание 2. Исследование дешифратора

Собрать схему, показанную на рис. 3.2. Входные сигналы задать в соответствии с рис. 3.2. Исследование целесообразно проводить в пошаговом режиме. Зарисовать временные диаграммы. Объяснить полученные результаты.

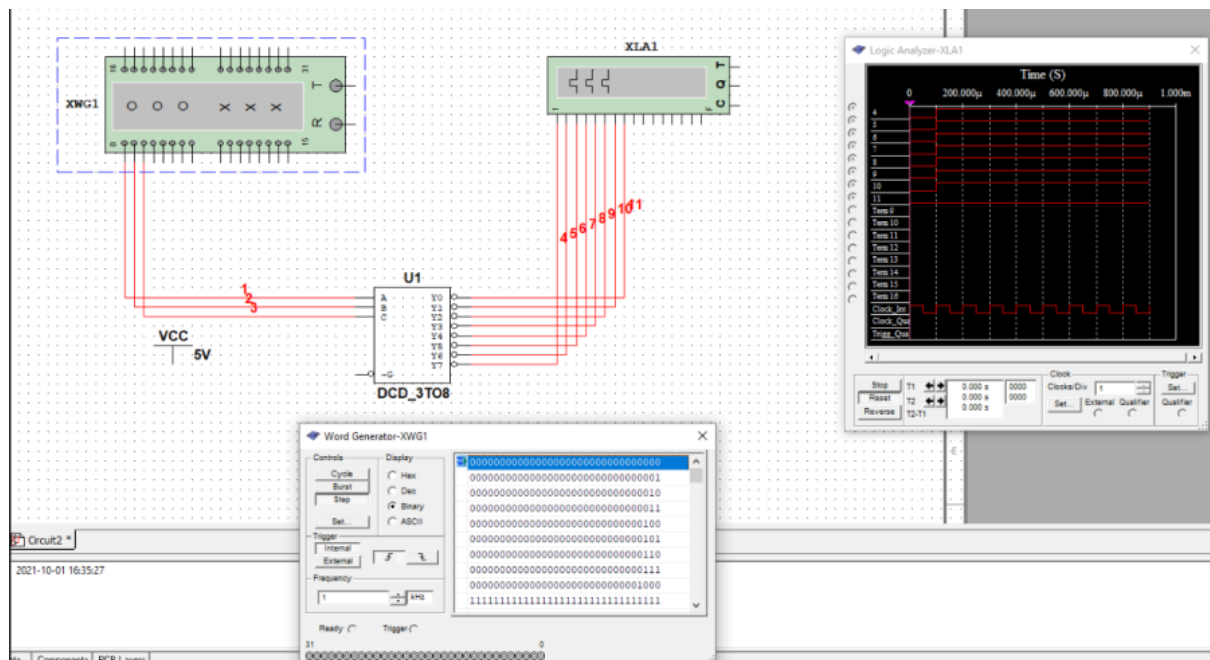


Рис. 3.2 – Исследование дешифратора

Задание 3. Исследование дешифратора 74154

Собрать схему, показанную на рис. 3.3. Задать входные сигналы. Исследование целесообразно проводить в пошаговом режиме. Зарисовать временные диаграммы. Объяснить полученные результаты.

Результаты занести в таблицу, которая должна отображать четыре входных и шестнадцать выходных сигналов. Входные сигналы следует фиксировать в двоичной и шестнадцатеричной системах счисления.

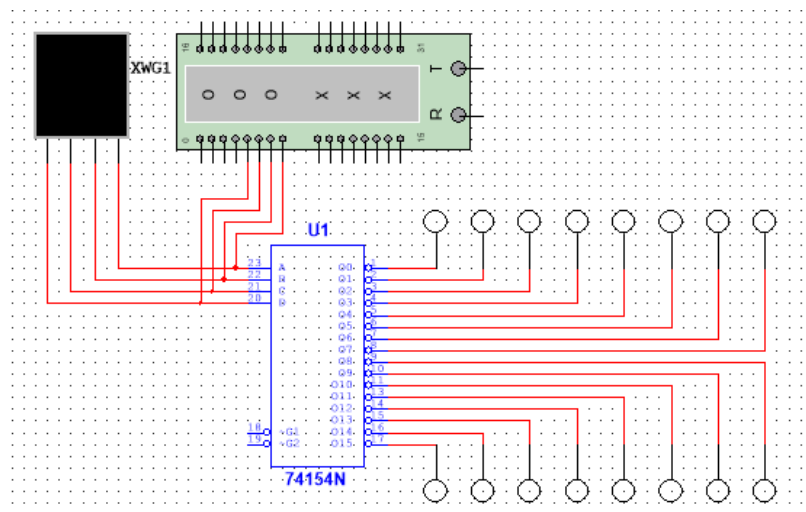


Рис.3.3 – Исследование дешифратора 74154

Шифратор – КЦУ, которое получает активный сигнал на одном из входов и формирует соответствующий двоичный код на выходах.

Активным сигналом в шифраторе (задание 1) является ноль. К выходам шифратора подключены три инвертора, что облегчает восприятие закодированной информации в двоичном коде.

У исследуемого шифратора входные зажимы обозначены цифрами 0, 1,..., 7. Выходы обозначены символами A0, A1, A2. Шины E0, E1, GS являются управляющими.

"Светящийся" индикатор соответствует логической единице, а потухший – логическому нулю.

В системе Multisim индикаторы можно выбрать на панели радиоэлементов, либо с помощью главного меню: Window – Indicators.

Дешифратор – КЦУ, на входы которого подаются двоичные коды, а активный сигнал появляется лишь на одном из нескольких выходов.

Во втором и третьем заданиях активным выходным сигналом является логический ноль.

Во втором задании входные сигналы подаются на шины A, B C. Входы G2A, G2B, G1 являются управляющими. На вход G1 необходимо подать логическую 1. Это сделано с помощью источника напряжения +5V.

В третьем задании входные сигналы подаются на входы А, В, С, D. Выходные шины обозначены цифрами: 0, 1, ..., 15. Входы G1 и G2 являются управляющими.

Исследование КЦУ удобно производить в пошаговом режиме генератора слов, либо в цикле на частоте 1 Гц.

В первом задании время развертки логического анализатора следует установить 1 с/дел (Time Base: 1 s/div).

Контрольные вопросы

1. Что называется шифратором?
2. Что называется дешифратором?
3. Что называется приоритетным шифратором?
4. Что называется комбинационным цифровым устройством?
5. Чем отличаются полный и неполный дешифраторы?
6. Сколько выходов имеет полный дешифратор с тремя входами?
7. Где могут использоваться шифраторы и дешифратор?
8. Нарисуйте временные диаграммы дешифратора с тремя входами.
9. Составить таблицу истинности для шифратора с восьмью входами.
10. Приведите условное графическое обозначение дешифратора.
11. Приведите условное графическое обозначение шифратора.

Лабораторная работа 4

Мультиплексоры и демультимплексоры

К типовым КЦУ относятся так же мультиплексор, демультимплексор. Мультиплексоры используются для организации мультиплексной линии или перехода от параллельной передачи двоичных наборов к последовательной. Демультимплексоры решают задачу, обратную задаче мультиплексирования

Цель работы: Исследовать работу мультиплексор, демультимплексор и уяснить разницу.

Подготовка к работе

Изучить материал по лекциям:

- Основное назначение мультиплексора
- Основное назначение демультимплексора

Задание 1. Исследование мультиплексора

Собрать схему, показанную на рис 4.1. Зарисовать временные диаграммы. Входные сигналы задать в соответствии с таблицей 4.1.

Входные сигналы логической единицы и нуля следует формировать с помощью соответственно источника напряжения 5В и заземления.

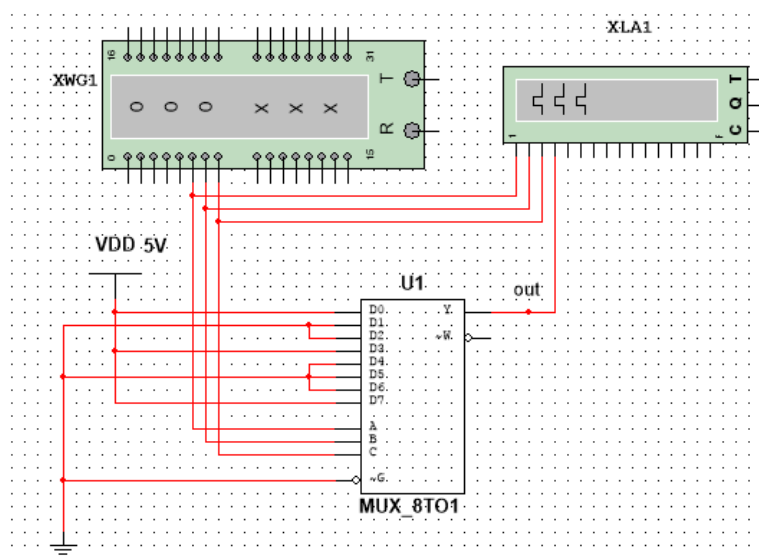


Рис. 1.4 – Исследование мультиплексора

Таблица 4.1

	Входные сигналы							
Варианты	D0	D1	D2	D3	D4	D5	D6	D7
Нечетные	1	1	1	0	0	1	0	1
Четные	0	1	1	0	1	0	0	1

Задание 2. Исследование демультиплексора

Собрать схему, показанную на рис. 4.2. Адресные сигналы задать в соответствии с таблицей 4.2. Задание выполнить дважды: первый раз сформировать информационный сигнал $G = 0$, второй раз $G = 1$. Нужные информационные сигналы устанавливаются ключом Q.

Результаты занести в таблицу 4.2.

Исследование демультиплексора следует производить в пошаговом режиме. Для этого используется кнопка STEP генератора слов.

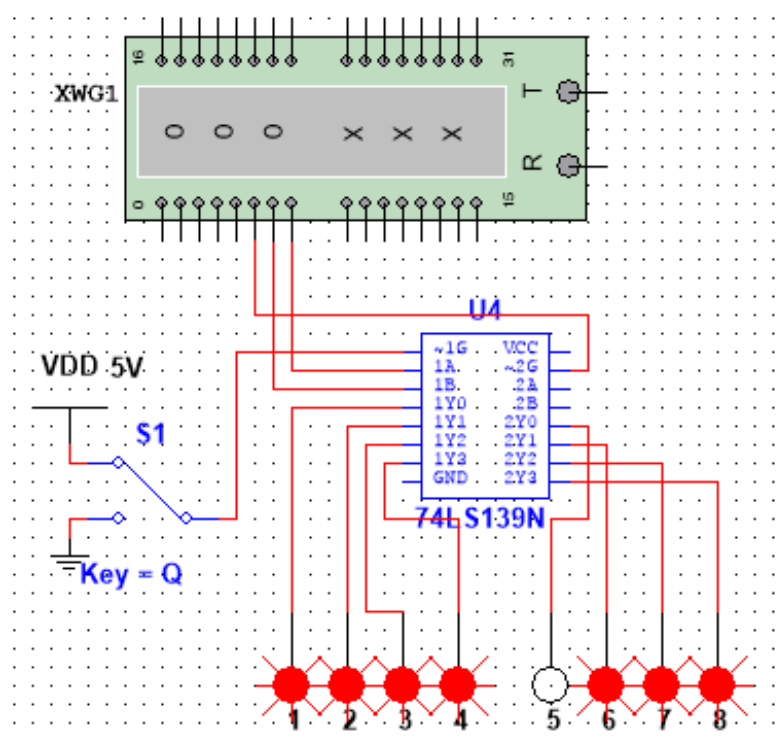


Рис. 4.2 – Схема демультиплексора

Таблица 4.2

Адрес			Выходы							
С	В	А	0	1	2	3	4	5	6	7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Замечание рис. 4.3

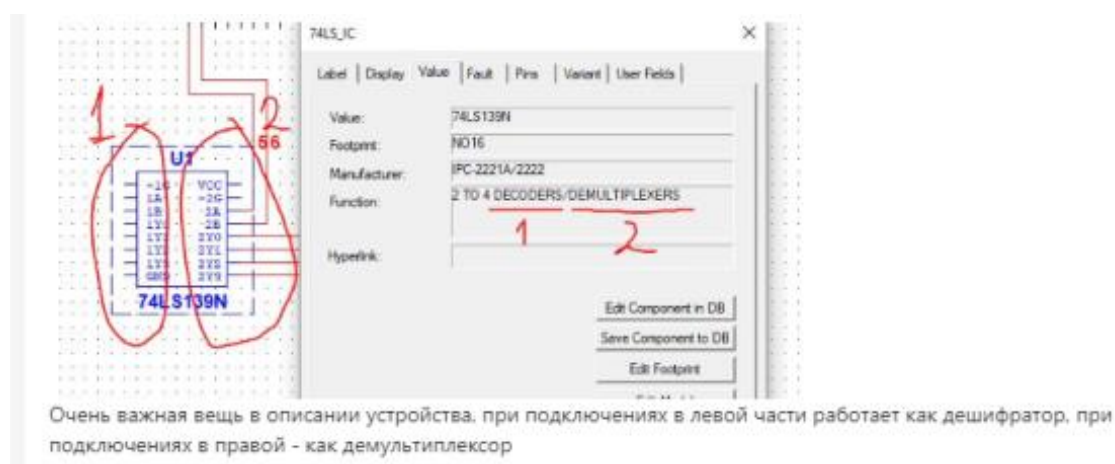


Рис. 4.3

Задание 3. Исследование двухканального мультиплексора

Собрать схему, показанную на рис. 4.4. С помощью логического преобразователя (конвертора) получить таблицу истинности мультиплексора 2:1. Полученную таблицу зарисовать в отчет, объяснить полученный результат.

На схеме входные шины (Data Input) обозначены символами DI1 и DI2, выходная шина (Data Output) – DO, адресный вход – буквой А.

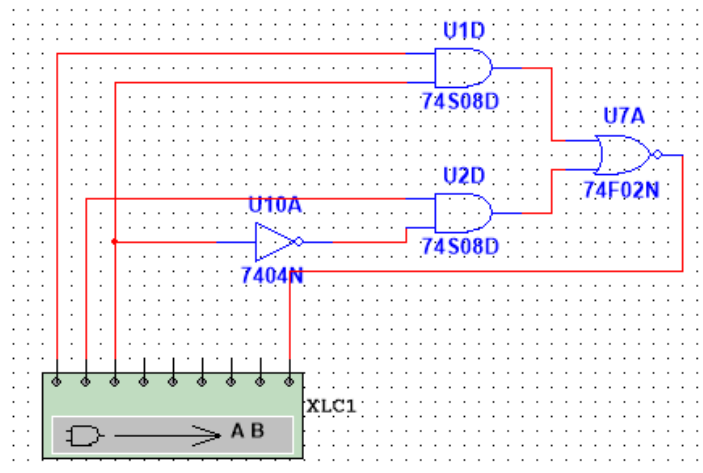


Рис. 4.4–Схема двухканального мультиплексора

Мультиплексор – электронный коммутатор, который поочередно подключает несколько информационных входов к единственному выходу. Выбор нужного входа осуществляется с помощью адресных входов.

Три адресных входа мультиплексора 8:1 в задании 1 обозначены символами А, В, С. Информационные входы обозначены символами D0, D1,..., D7. Выход обозначен буквой Y, а инверсный выход – буквой W. Подача на вход G логического нуля разрешает работу мультиплексора, а – логической единицы запрещает работу.

Демультимплексор выполняет обратную по отношению к мультиплексору функцию: подключает единственный информационный вход G к восьми выходам (под управлением адресных сигналов А, В, С).

Заметим, что младшим адресным разрядом является шина А (задание 2).

Генератор слов должен последовательно формировать управляющие сигналы 000, 001,..., 111.

Одной из интересных возможностей логического преобразователя является его способность автоматически формировать таблицу истинности для заданного цифрового устройства

Контрольные вопросы

1. Что называется мультиплексором?
2. Что называется демultipлексором?
3. Чем отличается демultipлексор от дешифратора?
4. Как демultipлексор превратить в дешифратор?
5. Сколько выходов может иметь демultipлексор, если число адресных входов равно m ?
6. Приведите условное графическое обозначение мультиплексора.
7. Приведите условное графическое обозначение демultipлексора.
8. Каким выражением описывается мультиплексор?
9. Нарисуйте схему простейшего мультиплексора.
10. Нарисуйте схему простейшего демultipлексора.
11. Нарисуйте временные диаграммы для мультиплексора с двумя адресными входами.
12. Почему мультиплексор называют универсальным комбинационным цифровым устройством?

Лабораторная работа 5

Триггеры

Основными компонентами ПЦУ являются запоминающие элементы, которые реализуются специальными устройствами – триггерами

Цель работы: Исследовать асинхронный RS-триггер, D-триггера, T-триггера, JK-триггера.

Подготовка к работе

Изучить материал по теме

- Основное назначение триггеров
- Их принцип работы, классификация триггеров.

Задание 1 Исследование асинхронного RS-триггера

Собрать схему, показанную на рис 5.1. Заполнить таблицу 5.1.

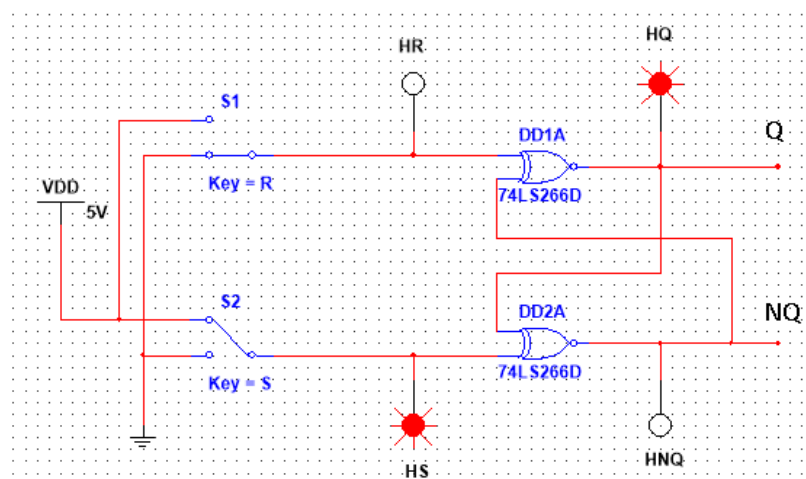


Рис. 5.1 – Схема асинхронного RS-триггера

Таблица 5.1

Полная таблица переходов асинхронного RS-триггера

Номер состояния	Такт t			Такт t+1		Примечание
	Q	R	S	Q	\overline{Q}	
0	0	0	0			
1	0	0	1			
2	0	1	0			
3	0	1	1			
4	1	0	0			
5	1	0	1			
6	1	1	0			
7	1	1	1			

Задание 2. Исследование D-триггера

Собрать схему, показанную на рис. 5.2, подключив к входам триггера генератор слов (Word Generator), а к выходам – логический анализатор - многоканальный осциллограф (Logic Analyzer).

Установить рабочую частоту (FREQUENCY) генератора слов 1 Гц (1 Hz). Время развертки логического анализатора 1 с/деление (1 s/div). Установка осуществляется с помощью стандартных элементов – счетчиков.

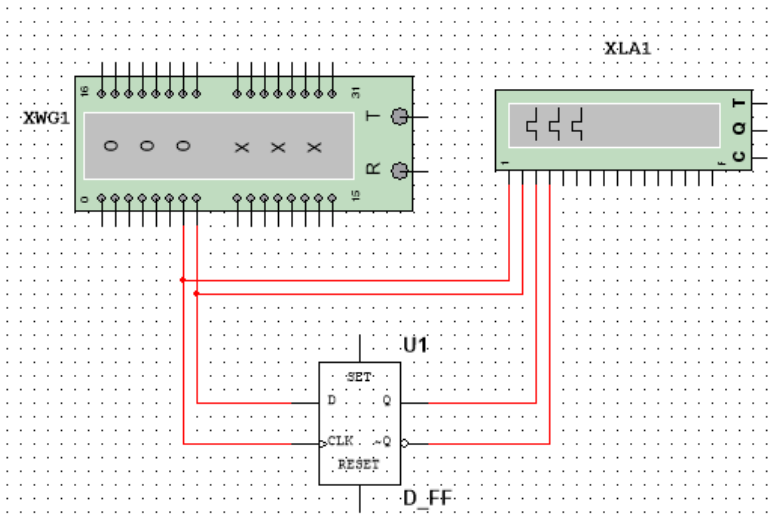


Рис. 5.2 – Схема D-триггера

Зарисовать осциллограммы (D-вход, тактовый С - вход, прямой и инверсный выходы). Осциллограммы должны быть обозначены в соответствии со схемой исследования.

Сигналы на выходах генератора слов должны быть установлены в соответствии с таблицей 5.2.

Таблица 5.2

	Последовательность слов															
Варианты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Нечетные	1	3	1	3	0	2	0	2	1	3	1	3	1	2	0	2
Четные	0	2	1	3	1	3	0	2	0	2	1	3	1	3	1	2

Примечание.

Предположим, что у Вас вариант 6 (четный). Значит, слово 0 имеет значение 0, слово 1 имеет значение 2 и т.д.

Задание 3. Исследование Т-триггера

Составить схему, показанную на рис. 5.3.

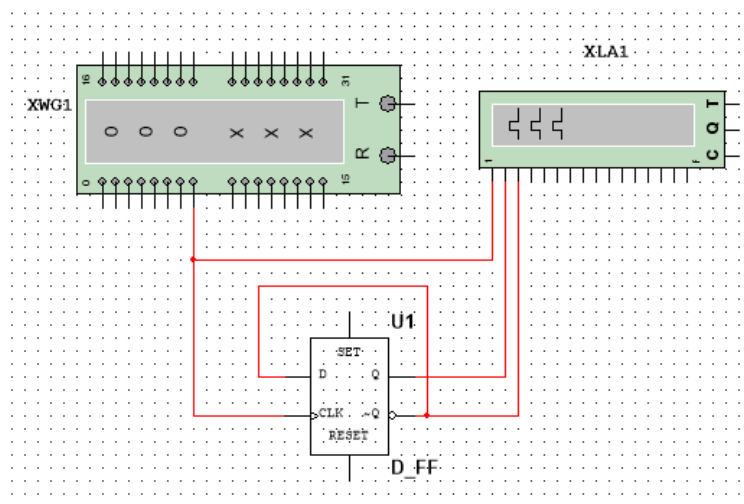


Рис 5.3 – Схема Т-триггера

С выхода генератора слов на тактовый вход D-триггера подать последовательность сигналов 1- 0 - 1 - 0....т.е. прямоугольные импульсы (меандр).

Зарисовать осциллограммы, появляющиеся на входе триггера и на двух его выходах. Объяснить полученный результат.

Задание 4. Исследование JK-триггера

Составить схему, показанную на рис. 5.4.

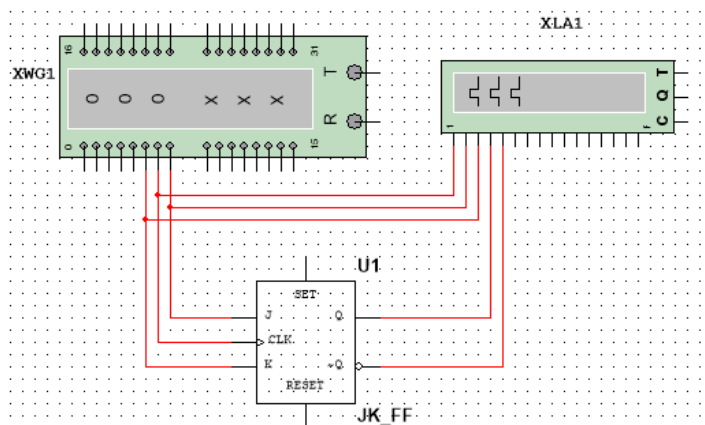


Рис. 5.4 – Схема JK-триггера

Таблица 5.4

	Последовательность слов															
Варианты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Нечетные	6	4	2	0	3	1	2	0	7	5	7	5	7	0	2	4
Четные	7	4	2	0	2	1	3	0	6	4	2	5	7	5	7	1

Зарисовать осциллограммы на трех входах триггера и на двух его выходах. Объяснить полученный результат.

Методические указания

В задании 1 исследуется так называемая таблица переходов, которая показывает, в каком состоянии окажется RS-триггер после подачи определенных сигналов на входы R и S.

RS-триггер собран на двух элементах ИЛИ-НЕ (DD1 и DD2). Активным сигналом является логическая единица (триггер с прямыми входами). Прямой выход триггера на схеме обозначен символом Q, а инверсный - символом NQ.

При выполнении задания 1 нужно вначале установить триггер в состояние, указанное в табл. 1 в колонке Q (текущий такт t). Затем подать сигналы на входы R и S, указанные в одной строке, и зафиксировать состояние, в которое перейдет (или не перейдет) триггер. При исследовании триггера результаты заносятся в колонки, где размещены сигналы, появляющиеся на выходах триггера в следующем такте (такт t+1).

В последнюю колонку нужно занести примечания типа: "Хранение", "Установка 1", "Установка 0", "Запрещенное состояние".

Особое внимание нужно уделить запрещенному состоянию, которое противоречит алгебре Буля (прямые и инверсные сигналы на выходах триггера одинаковые).

Чтобы установить в генераторе слов нужную комбинацию двоичных разрядов (то есть слово, байт), следует выбрать соответствующую позицию и напечатать 1 или 0.

Обратите внимание, что младший разряд генератора слов должен быть подключен к D-входу.

Для формирования циклической развертки на генераторе слов должна быть нажата командная кнопка CYCLE. Для детального исследования временных диаграмм рекомендуется использовать пошаговый режим развертки (командная кнопка STEP).

Целесообразно исследовать работу триггера в пошаговом режиме, стремясь заранее предсказать возможные переключения триггера на каждом такте (так можно проверить правильность своих представлений о работе триггера). При выполнении задания 1 целесообразно "держать в голове" таблицу истинности для элемента ИЛИ-НЕ, с помощью которой легко объяснить работу триггера.

В данной работе RS-триггер является асинхронным (его переключение происходит под управлением только сигналов, подаваемых на информационные входы). Т-, D- и JK-триггеры являются динамическими синхронными триггерами (они переключаются лишь при подаче тактовых импульсов). Причем JK-триггер срабатывает по заднему фронту, а D-триггер – по переднему фронту.

Контрольные вопросы

1. Дайте определение последовательностных цифровых устройств.
2. Как из D-триггера получить Т-триггер?
3. Как из JK-триггера получить Т-триггер?
4. Как из JK-триггера получить D-триггер?
5. Что такое запрещенное состояние для RS-триггера?
6. Существуют ли запрещенные состояния для JK-триггеров?
7. Нарисуйте временные диаграммы синхронного RS-триггера.
8. Нарисуйте временные диаграммы асинхронного RS-триггера.
9. Нарисуйте временные диаграммы JK-триггера.
10. Нарисуйте временные диаграммы D-триггера.
11. Нарисуйте временные диаграммы Т-триггера.

Лабораторная работа 6

Знакомство с симулятором TMS320C6201

Симуляторы являются программами, имитирующими работу процессора на уровне его команд. Используются они для тестирования и улучшения программного кода. Симуляторы предоставляют возможность как пошагового, так и автоматического выполнения (прогона) программы.

Цель работы

Изучить основные приемы работы с симулятором команд ассемблера сигнального процессора TMS320C6х.

Подготовка к работе

По указанной выше литературе изучить:

- структуру процессора TMS320C6х;
- этапы разработки программы;
- процесс подготовки исполняемой программы;
- средства отладки прикладных программ: аппаратные эмуляторы, проверочные модули, симуляторы команд, отладчики.

Задание и порядок выполнения работы

1. Запустить симулятор (C:\C6XTOOLS\CODEGEN.110\BIN\SIM62x.exe) и изучить окна его интерфейса.

В режиме отладки автоматически создаются четыре окна с отображением чисел в 16-ричной системе счисления:

- окно Disassembly отображает команды дизассемблера – ассемблера, восстановленного по объектному коду, содержащемуся в программной памяти симулятора. В первой слева колонке этого окна отображаются адреса ячеек программной памяти симулятора, во второй – их содержимое (объектный код), в третьей – мнемоники команд и имена исполняющих их модулей процессора, в четвертой – поле операндов командной строки

ассемблера;

–окно CPU показывает содержимое регистров процессора. При необходимости его можно задавать принудительно. Для этого двойным щелчком компьютерной мыши по имени регистра выделяется его содержимое, вводится требуемое 16-ричное число и нажимается клавиша <Enter>;

–окно Memory по умолчанию дублирует первые две слева колонки окна Disassembly. Здесь также можно изменить содержимое ячеек программной памяти посредством двойного щелчка компьютерной мыши по нужным разрядам выбранной ячейки памяти;

–окно Command включает область ввода команд управления симулятором и область отображения сообщений об ошибке загрузки программы, результате выполнения введенной команды управления симулятором и служебной информации.

2. Создать исполняемый программный модуль.

Исполняемый модуль непосредственно загружается в симулятор. Получается он в результате следующей последовательности действий.

1). В текстовом редакторе «Блокнот» сформировать текст исходной программы на языке ассемблера.

В данной работе ввести текст:

```
k .set 2          ; присвоение символу k значения 2
mvk k,a2          ; ввод значения k в РОН a2
mv a2,b2          ;копирование содержимого a2 в РОН b2
add a2,b2,a2       ;сложение содержимого a2 и b2 с
```

*размещением результата в a2.

Обратите внимание:

- каждая командная строка начинается как минимум с одного пробела, поскольку предыдущие поля не используются;
- символ «;» открывает текст комментария (на машинный язык не переводится) при его размещении в данной командной строке, а символ «*» – если он начинается с первого поля строки ассемблера.

2). Сохранить текст исходной программы в рабочей папке <Family_N> под именем lb1_N.asm.

С этой целью в пункте «Файл» оконного меню редактора «Блокнот» выбрать команду *Сохранить как ...* В открывшемся окне диалога выполнить следующее:

- указать место хранения, то есть на дереве папок найти и открыть свою рабочую папку;
- в списке Тип файла выбрать Все файлы;
- в поле Имя файла ввести полное имя файла (здесь lb1_N.asm);
- компьютерной мышью «щелкнуть» кнопку Сохранить.

3). Получить объектный файл (здесь lb1.obj), для чего из рабочей папки запустить программу ассемблера (файл ASM6x.exe) и в открывшемся окне ввести имя ассемблируемого файла, причем расширение имени указывать не обязательно (в данном случае достаточно ввести lb1_N). После этого нажать клавишу <Enter>.

Отсутствие в рабочей папке объектного файла означает, что в исходном файле (здесь lb1_N.asm) имеются ошибки. Определить их удобно с помощью файла листинга программы, который получается следующим образом:

- из командной строки
- (Пуск\Программы\Стандартные\Выполнить) запустить программу Ассемблера: <путь к рабочей папке>\<имя рабочей папки>\ASM6x.exe -l

- (опция -l отделяется от имени файла пробелом);
- в окне программы ассемблера, как и ранее, ввести имя ассемблируемого файла (здесь lb1_N);
 - нажать клавишу <Enter>.

В результате в рабочей папке образуется файл листинга (здесь lb1_N.lst), содержащий сведения об ошибках.

После коррекции текста исходного файла сохранить его (команда *Сохранить*) и повторить ассемблирование.

4). Получить исполняемый файл (здесь lb1_N.out), для чего из рабочей папки запустить программу компоновщика (файл LNK6x.exe) и в открывшемся окне (рис. 6.1) в режиме диалога последовательно и построчно ввести сведения о компокуемых файлах. При этом достаточно ограничиться лишь именем объектного файла, причем без расширения, согласившись тем самым с именем исполняемого файла (здесь lb1_N.out), предлагаемым компоновщиком (рис. 6.1).

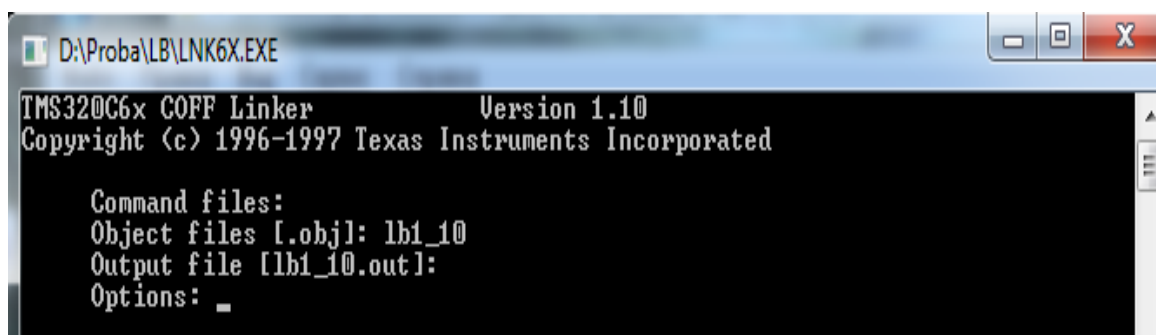


Рис. 6.1 – Окно программы компоновщика

3. Загрузить исполняемый модуль в симулятор:

- в пункте File оконного меню симулятора выбрать команду Load Program;
- в одноимённом окне диалога открыть список поля Папка и на дереве папок найти и открыть рабочую папку;
- в поле Имя того же окна диалога выделить имя исполняемого модуля (здесь lb1_N.out), после чего с помощью компьютерной мыши нажать

кнопку «Открыть».

В окне Command симулятора появиться сообщение о факте загрузки отлаживаемого файла.

4. Прогон программы.

В режиме отладки программ симулятор обеспечивает два основных способа их прогона – по контрольным точкам (точкам останова выполнения программы) и пошаговый.

В первом случае выполнение каждого участка программы (между двумя соседними контрольными точками) инициируется командой *Run* из пункта Target оконного меню либо одноимённой кнопкой панели инструментов окна симулятора (рис. 6.2), либо клавишей <F5> клавиатуры.

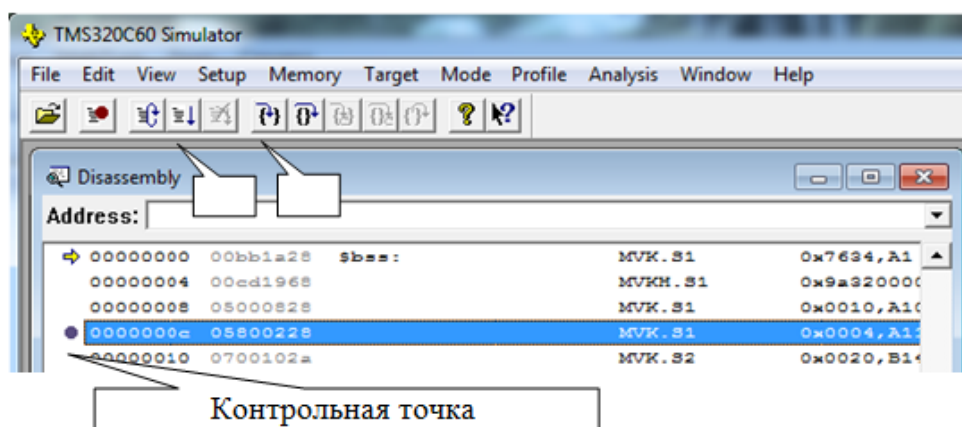


Рис. 6.2 – Способы прогона программы

Устанавливаются контрольные точки в окне Disassembly щелчком компьютерной мыши слева от требуемой строки (рис. 6.2). Снимается контрольная точка щелчком мыши по её изображению.

Для апробации данного режима прогона установите контрольную точку на строке с адресом 00000004 и инициируйте команду *Run*. Снимите контрольную точку и инициируйте команду *Restart* (найти и компьютерной мышью «щелкнуть» соответствующую кнопку панели инструментов окна симулятора).

Во втором случае выполнение каждой командной строки инициируется командой *Step* из пункта *Target* оконного меню либо одноимённой кнопкой панели инструментов окна симулятора (рис. 6.2), либо клавишей <F8> клавиатуры.

В процессе отладки контролируются следующие окна:

- *Disassembly*, где исполняемая на следующем шаге командная строка автоматически отмечается слева стрелкой;
- *CPU*, содержимое регистров которого сравнивается с прогнозом выполнения текущей командной строки.

Для апробации выполните в пошаговом режиме модуль *lb1_N.out*, отмечая изменения в окне *CPU* симулятора.

5. Завершить работу с симулятором.

Закрывать симулятор можно любым из трех способов:

- с помощью кнопки «×», расположенной в правом верхнем углу строки заголовка окна симулятора;
- ввести в командную строку окна *Command* команду *quit*;
- в пункте оконного меню *File* выбрать команду *Exit*.

Опробовать все способы и выбрать для себя наиболее удобный из них.

Контрольные вопросы

1. Поясните структуру строки ассемблера процессора TMS320C6x.
2. Чем отличаются директива и команда ассемблера?
3. Назовите основные этапы преобразования исходной программы в исполняемый программный модуль.
4. Дайте краткую характеристику этапа редактирования текста исходной программы и соответствующим инструментальным средствам.

5. Дайте краткую характеристику этапа трансляции исходной программы.
6. Дайте краткую характеристику этапа загрузки объектных модулей в оперативную память.
7. Дайте краткую характеристику этапа компоновки объектных модулей.
8. Поясните назначение файла листинга исходной программы и способ его получения с помощью инструментов симулятора команд процессора TMS320C6х.
9. Поясните назначение и содержание окна Disassembly симулятора команд процессора TMS320C6х.
10. Поясните назначение и содержание окна CPU симулятора команд процессора TMS320C6х.
11. Поясните процесс загрузки исполняемого модуля в симулятор команд процессора TMS320C6х.
12. Поясните способы прогона программы в симуляторе команд процессора TMS320C6х.

Лабораторная работа 7

Пересылка данных

Пересылка данных включает операции ввода исходных данных в РОН процессора, обмена данными между РОН (копирование содержимого одного регистра в другой), а также между РОН и памятью данных процессора.

Цель работы

Изучить особенности команд пересылки данных, сохранения в память и загрузки из памяти ассемблера процессора TMS320C6x.

Подготовка к работе

1. По указанной выше литературе изучить:
 - структуру командной строки ассемблера процессора TMS320C6x;
 - процесс выполнения программы процессором TMS320C6x;
 - методы адресации операндов;
 - форматы команд пересылки данных (между РОН, загрузки/хранения и ввода исходных данных) ассемблера TMS320C6x и особенности их выполнения.
2. Ознакомиться с методическими указаниями
3. Подготовить отчет.

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6x подготовить программу, соответствующую следующей последовательности операций.

Операция 1. В регистр R1 (назначить из РОН по своему усмотрению) ввести число, выбранное из табл. 7.1 в соответствии с номером варианта N (№ студента в списке группы).

Таблица 7. 1

Исходный операнд

N	Число, Нех	N	Число, Нех	N	Число, Нех	N	Число, Нех
1	80A1F5C1	9	A123F1C0	17	C345A5B7	25	E456F792
2	A90C7D5	10	C70A4B2	18	E34F5B1	26	8E2C795
3	C5D0A5	11	E4C6A0	19	9AC580	27	C1A293
4	AF9C5	12	CD3A0	20	EC7A6	28	890A7
5	90B3E8C9	13	B23495A0	21	D046E890	29	F598C4E5
6	B30B5A8	14	D89E5F0	22	F67C3E4	30	9F6B287
7	D9A2B3	15	F7B0D3	23	B9B483	31	D0E184
8	BE7C4	16	DA0B9	24	FB1D0	32	9A3CB

Операция 2. В регистр R2 (назначить из РОН по своему усмотрению, но с учетом последующих заданий работы и $R2 \neq R1$) ввести число 50h в качестве базового адреса памяти данных (ПД) процессора.

Операция 3. Сохранить содержимое R1 в ПД процессора в соответствии с условиями из таблицы 7.2

Таблица 7. 2

Условия сохранения операнда

N	Условие сохранения
1	В ячейке памяти F4h с изменением содержимого R2.
2	В ячейке памяти 50h с изменением содержимого R2 до величины 48h.
3	В ячейке памяти 50h с изменением содержимого R2 до величины 64h.
4	В ячейке памяти 4Ch без изменения содержимого R2.
5	В ячейке памяти 60h с изменением содержимого R2.
6	В ячейке памяти 50h с изменением содержимого R2 до величины 4Ch.
7	В ячейке памяти 70h без изменения содержимого R2.
8	В ячейке памяти 50h с изменением содержимого R2 до величины F0h.
9	В ячейке памяти 50h без изменения содержимого R2.
10	В ячейке памяти 48h с изменением содержимого R2.
11	В ячейке памяти 54h без изменения содержимого R2.
12	В ячейке памяти 30h с изменением содержимого R2.
13	В ячейке памяти 50h с изменением содержимого R2 до величины 54h.
14	В ячейке памяти 34h без изменения содержимого R2.
15	В ячейке памяти 58h с изменением содержимого R2.
16	В ячейке памяти 51h.
17	В ячейке памяти 50h с изменением содержимого R2 до величины 3Ch.
18	В ячейке памяти 80h без изменения содержимого R2.

N	Условие сохранения
19	В ячейке памяти 50h с изменением содержимого R2 до величины 60h.
20	В ячейке памяти 38h с изменением содержимого R2.
21	В ячейке памяти 5Ch без изменения содержимого R2.
22	В ячейке памяти 20h с изменением содержимого R2.
23	В ячейке памяти 4Ch.
24	В ячейке памяти 50h с изменением содержимого R2 до величины 68h.
25	В ячейке памяти 40h без изменения содержимого R2.
26	В ячейке памяти 6Ah с изменением содержимого R2.
27	В ячейке памяти 50h с изменением содержимого R2 до величины 48h.
28	В ячейке памяти 78h без изменения содержимого R2.
29	В ячейке памяти 50h с изменением содержимого R2 до величины 90h.
30	В ячейке памяти 50h с изменением содержимого R2 до величины 51h.
31	В ячейке памяти 40h с изменением содержимого R2.
32	В ячейке памяти 58h без изменения содержимого R2.

Операция 4. Из ячейки ПД процессора, используемой в предыдущем пункте, загрузить в регистр R3 (назначить из РОН по своему усмотрению и $R3 \neq R1$, $R3 \neq R2$) число в соответствии с условиями, выбранными из таблицы 7.3.

Таблица 7.3

Условия загрузки числа в РОН

N	Условия загрузки
1	Полуслово с расширением знаком и без изменения содержимого R2.
2	Байт без расширения знаком и изменения содержимого R2.
3	Полуслово без расширения знаком и с изменением содержимого R2.
4	Байт с расширением знаком и изменением содержимого R2.
5	Полуслово с расширением знаком и изменением содержимого R2 до величины 52h.
6	Байт без расширения знаком и изменения содержимого R2.
7	Полуслово без расширения знаком и с изменением содержимого R2.
8	Байт без расширения знаком и с изменением содержимого R2.
9	Полуслово с расширением знаком и изменением содержимого R2 до величины 76h.
10	Байт с расширением знаком и без изменения содержимого R2.
11	Полуслово без расширения знаком и с изменением содержимого R2.
12	Байт без расширения знаком и с уменьшением содержимого R2 на 1.
13	Полуслово с расширением знаком и без изменения содержимого R2.
14	Байт с расширением знаком и изменением содержимого R2.

N	Условия загрузки
15	Полуслово без расширения знаком и с увеличением содержимого R2 на 1.
16	Байт без расширения знаком и изменения содержимого R2.
17	Полуслово с расширением знаком и без изменения содержимого R2.
18	Байт с расширением знаком и изменением содержимого R2.
19	Полуслово без расширения знаком и изменения содержимого R2.
20	Байт без расширения знаком и с уменьшением содержимого R2 на 1.
21	Полуслово с расширением знаком и изменением содержимого R2.
22	Байт с расширением знаком и уменьшением содержимого R2 на величину 12h.
23	Полуслово без расширения знаком и с увеличением содержимого R2 на величину Eh.
24	Байт без расширения знаком и изменения содержимого R2.
25	Полуслово с расширением знаком и изменением содержимого R2.
26	Байт с расширением знаком и без изменения содержимого R2.
27	Полуслово без расширения знаком и изменения содержимого R2.
28	Байт без расширения знаком и с изменением содержимого R2.
29	Полуслово с расширением знаком и изменением содержимого R2.
30	Байт с расширением знаком и без изменения содержимого R2.
31	Полуслово без расширения знаком и изменения содержимого R2.
32	Байт без расширения знаком и с изменением содержимого R2.

Операция 5. 1. Переслать (скопировать) содержимое R3 в R1.

2. Получить исполняемый программный модуль.

3. Загрузить исполняемый модуль в симулятор.

4. В пошаговом режиме выполнить прогон программы, сравнивая данные прогноза с соответствующими данными окна CPU симулятора.

5. Предъявить результаты выполнения работы преподавателю, после чего завершить работу с симулятором.

Методические указания

Операция 1. Для ввода исходных данных используются команды MVK, MVKH и MVKLN. При этом если:

- число не превышает полуслова (от -32767 до +32767) достаточно одной команды MVK. При этом следует помнить, что эта команда выполняется с расширением знаком;

- число превышает полуслово, необходимы две команды: сначала MVK, а затем MVKH или MVKLN. Последняя команда обязательна при 16-ричном представлении операнда и буквой в старшем его разряде.

Операция 3. Поскольку речь идет о сохранении всего содержимого регистра R1, следует воспользоваться командой:

STW R1,*A_к,

где A_к – адресный код.

При формировании A_к в зависимости от целей дальнейшего использования регистра базового адреса (базы) R2 применяется один из следующих типов адресации:

- косвенная, если R2 содержит требуемый исполнительный адрес (Ai) и изменять содержимое R2 не следует;
- базирование, если R2 не содержит требуемый Ai и изменять содержимое R2 не следует;
- преиндексация, если R2 не содержит требуемый Ai и после операции пересылки необходимо заменить содержимое R2 на Ai;
- постиндексация, если R2 содержит требуемый Ai, но после операции пересылки можно или нужно заменить содержимое R2 на Ai;
- преавтоинкремент или преавтодекремент, если содержимое R2 на 1 меньше или, соответственно, больше требуемого Ai и после операции пересылки содержимое R2 необходимо заменить на Ai;
- поставтоинкремент или поставтодекремент, если R2 содержит требуемый Ai, но после операции пересылки можно или нужно изменить содержимое R2 на 1.

Операция 4. Операции загрузки в зависимости от объёма данных реализуются посредством команд:

LDW(H, B) *A_к,r,

где r – имя регистра-приемника операнда. При этом следует помнить:

- третья буква мнемоники помимо объёма пересылаемых данных определяет закономерность изменения величины смещения;
- команды LDH и LDB выполняются с расширением знаком пересылаемой части слова. При необходимости расширения нулем следует применять команды LDW(H, B)U;
- команды загрузки имеют 4 слота задержки, то есть результат доступен для использования только спустя 4 такта после объявления команды. Так, если команда объявлена в n-ом такте, результат ее выполнения сформируется в (n+4)-ом такте, а использовать его можно, начиная только с (n+5)-го такта. Таким образом, в данной работе после команды загрузки необходимо объявить 4-тактный мультицикл NOP (нет операции): NOP 4.

Обобщённые алгоритм и программа (без указания конкретных чисел, имен регистров РОН, номера ячейки ПД (ЯП) и адресного кода A_k) представлены на рис. 4, где (Z) – содержимое объекта Z (регистра РОН или ячейки памяти данных), а символ \leftarrow обозначает операцию пересылки данных в требуемом направлении (рис.7.1).

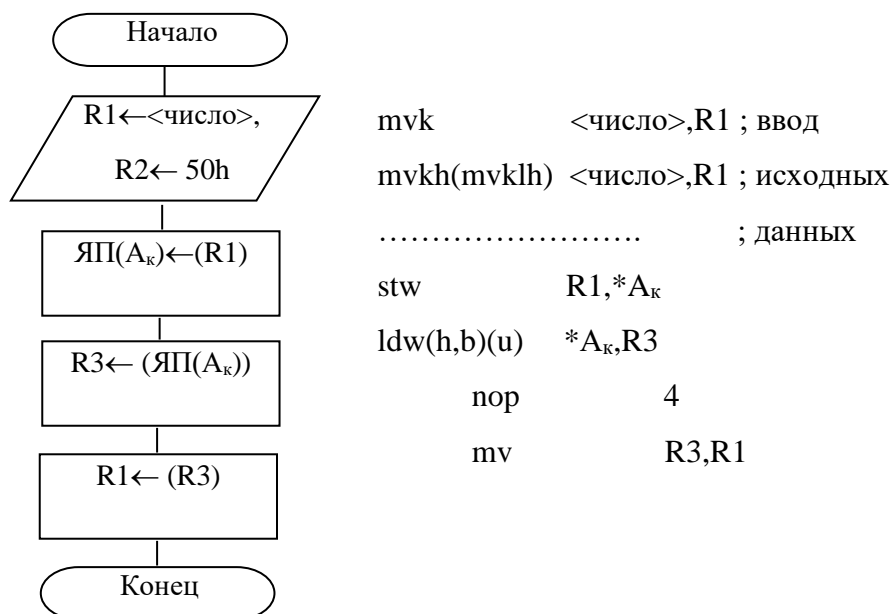


Рис. 7.1. Блок-схема и программа в обобщённом виде

Контрольные вопросы

1. Какие существуют методы адресации?
2. Сформулируйте правило формирования результата при выполнении команды вычитания над знаковыми операндами.
3. Сформулируйте правило формирования результата при выполнении команды ABS.
4. Приведите формат арифметической команды, заданной преподавателем.
5. Укажите функциональные особенности команды, заданной преподавателем.
6. Укажите ограничения на операнды команды, заданной преподавателем.
7. Определите результат выполнения команды, заданной преподавателем.
8. Назовите метод адресации, характерный практически для всех арифметических команд.
9. Какая особенность формирования адреса в командах загрузки/хранения в зависимости от третьей буквы в команде?
10. Какой объем данных пересылается в командах загрузки/хранения в зависимости от третьей буквы в команде?

Лабораторная 8

Арифметические операции

Функциональные устройства (модули) ядра процессора способны вычислять суммы, разности и произведения операндов как знаковых, так и без знака.

Цель работы

Изучить особенности арифметических команд ассемблера процессора TMS320C6х.

Подготовка к работе

- По указанной выше литературе изучить форматы и особенности выполнения арифметических команд ассемблера TMS320C6х.
- Ознакомиться с методическими указаниями
- Подготовить отчет.

Задание и порядок выполнения работы

1. На языке ассемблера TMS320C6х подготовить программу, соответствующую следующей последовательности операций.

Операция 1. Выбирается из таблицы 8.1 по номеру варианта N с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению).

Таблица 8.1

Вариант операции

N	Содержание операции
1	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды знаковые числа.
2	Сложение знаковых чисел 9C5A8600h и 78B05D03h.
3	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды – числа без знака.
4	Вычитание над числами без знака: 78B05D03h - 9C5A8600h.
5	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число без знака, второй – со знаком.
6	Сложение знаковых чисел 9C5A8600h и A8B05D03h.
7	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число со знаком, второй – без знака.
8	Вычитание над знаковыми числами: 78B05D03h - 9C5A8600h.
9	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды – знаковые числа.

N	Содержание операции
10	Сложение 9C5A8600h с 5-разрядной положительной константой (выбрать по своему усмотрению).
11	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды – числа без знака.
12	Вычитание над знаковыми числами: 9C5A8600h - 78B05D03h.
13	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число без знака, второй – со знаком.
14	Сложение 5C5A8600h с 5-разрядной константой без знака (выбрать по своему усмотрению).
15	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число со знаком, второй – без знака.
16	Вычитание над знаковыми числами: BC5A8600h - 98B05D03h.
17	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, операнды со знаком.
18	Сложение чисел 4C5A8600h и 78B05D03h без знака.
19	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды – числа без знака.
20	Вычитание над знаковыми числами: 7C5A8600h - 48B05D03h.
21	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
22	Вычитание над знаковыми числами: 3C5A8600h - A8B05D03h.
23	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.
24	Вычитание над знаковыми числами: AC5A8600h - 38B05D03h.
25	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды – знаковые числа.
26	Сложение чисел EC5A8600h + D8B05D03h без знака
27	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды – числа без знака.
28	Вычитание над знаковыми числами: 9C5A8600h - B8B05D03h.
29	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
30	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.

Операция 2. Вычислить абсолютную величину содержимого R1 (для регистровой пары использовать только четный регистр) с размещением

результата в регистре R2 (назначить по своему усмотрению).

Операция 3 1). с размещением результата в регистре R2:

- для нечетных вариантов N содержимое R2 сложить с 16-разрядной знаковой константой (принять по своему усмотрению);
- для четных вариантов N сложить старшую и младшую половины R2 соответственно со старшей и младшей половинами числа, выбранного из таблицы 8.2.

Таблица 8.2

Второй операнд

N	Число, Нех	N	Число, Нех	N	Число, Нех
2	80A1F5C1	12	E4C6A0	22	D046E890
4	C5D0A5	14	B23495A0	24	B9B483
6	90B3E8C9	16	F7B0D3	26	E456F792
8	D9A2B3	18	C345A5B7	28	C1A293
10	A123F1C0	20	9AC580	30	F598C4E5

- 2). Получить исполняемый программный модуль.
- 3). Загрузить исполняемый модуль в симулятор.
- 4). В пошаговом режиме выполнить прогон программы, сравнивая данные прогноза с соответствующими данными окна CPU симулятора.
- 5). Предъявить результаты выполнения работы преподавателю, после чего завершить работу с симулятором.

Методические указания

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

При выполнении данного задания следует:

- предусмотреть регистры R3, R4, ... для размещения исходных данных;
- помнить, что команды умножения имеют 1 слот задержки.
- В остальном алгоритм и программа строятся аналогично предыдущей работе (рис. 8.1):

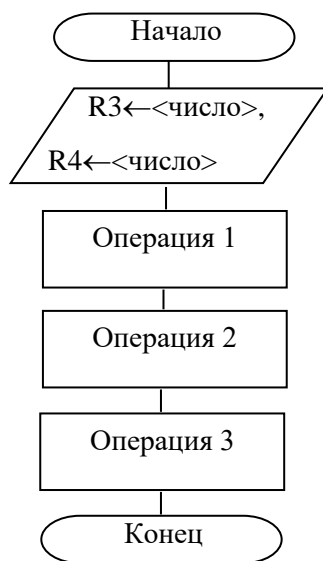


Рис. 8.1 – Блок-схема алгоритма в обобщённом виде

Контрольные вопросы

1. Сформулируйте правило формирования результата при выполнении команды вычитания над знаковыми операндами.
2. Сформулируйте правило формирования результата при выполнении команды ABS.
3. Приведите формат арифметической команды, заданной преподавателем.
4. Укажите функциональные особенности команды, заданной преподавателем.
5. Укажите ограничения на операнды команды, заданной преподавателем.
6. Определите результат выполнения команды, заданной преподавателем.
7. Назовите метод адресации, характерный практически для всех арифметических команд.

Лабораторная работа 9

Ветвление с простым условием

Ветвления позволяют реализовать альтернативные вычисления в зависимости от каких-либо условий. Наиболее просто ветвление организуется для одного условия с одним отношением, например $a < b$.

Цель работы

Изучить особенности реализации простых условий на языке ассемблера процессора TMS320C6x.

Подготовка к работе

- По указанной выше литературе изучить поле условия строки ассемблера TMS320C6x, а также форматы и особенности выполнения логических и сервисных команд.
- Уяснить особенности реализации нестрогих отношений в условии ветвления.
- Ознакомиться с методическими указаниями.
- Подготовить отчет.

Задание и порядок выполнения работы

1 На языке ассемблера TMS320C6x подготовить программу, реализующую алгоритм рис. 9.1.

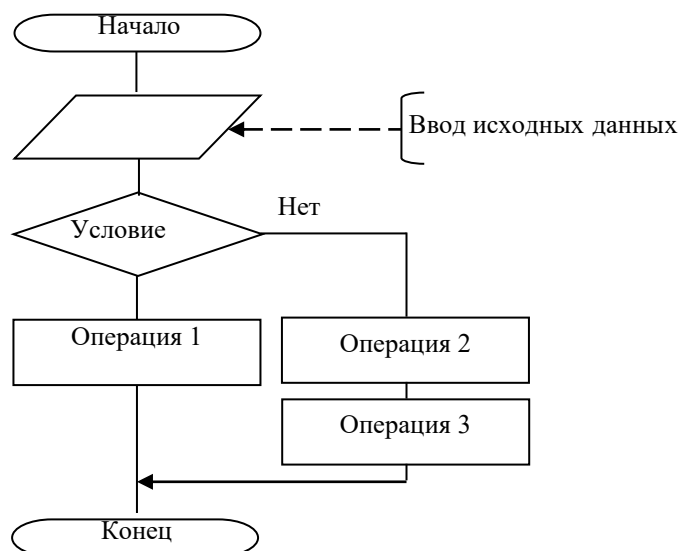


Рис. 9.1 Алгоритмическая структура «ветвление»

При этом:

Операция 1 заключается в вычислении абсолютной величины числа, взятого из таблицы 9.1.

Таблица 9.1.

Операнд							
N	Число, Нех	N	Число, Нех	N	Число, Нех	N	Число, Нех
1	9F6B287	9	F67C3E4	17	D89E5F0	24	D9A2B3
2	F598C4E5	10	D046E890	18	B23495A0	25	B30B5A8
3	890A7	11	EC7A6	19	CD3A0	26	90B3E8C9
4	C1A293	12	9AC580	20	E4C6A0	27	AF9C5
5	8E2C795	13	E34F5B1	21	C70A4B2	28	C5D0A5
6	E456F792	14	C345A5B7	22	A123F1C0	29	A90C7D5
7	FB1D0	15	DA0B9	23	BE7C4	30	80A1F5C1
8	B9B483	16	F7B0D3				

Операция 2 с размещением результата в регистре (регистровой паре) R1 (назначить из РОН по своему усмотрению) выбрать из таблицы 9.2.

Таблица 9.2

Операция	
N	Содержание и условия операции
1	Вычитание знаковых чисел 9C5A8600h - B8B05D03h.
2	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.

N	Содержание и условия операции
3	Сложение чисел EC5A8600h + D8B05D03h без знака.
4	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды числа без знака.
5	Вычитание знаковых чисел AC5A8600h - 38B05D03h.
6	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где операнды знаковые числа.
7	Вычитание знаковых чисел 3C5A8600h - A8B05D03h.
8	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.
9	Вычитание знаковых чисел 7C5A8600h - 48B05D03h.
10	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где первый операнд – число без знака, второй – со знаком.
11	Сложение чисел 4C5A8600h и 78B05D03h без знака.
12	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды числа без знака.
13	Вычитание знаковых чисел BC5A8600h - 98B05D03h.
14	Умножение 16 старших бит числа 9100A000h на 16 младших бит числа 8100C000h, где операнды знаковые числа.
15	Сложение 5C5A8600h с 5-разрядной константой без знака (выбрать по своему усмотрению).
16	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число со знаком, второй – без знака.
17	Вычитание знаковых чисел 9C5A8600h - 78B05D03h.
18	Умножение 16×16 старших бит чисел 9100A000h и 8100C000h, где первый операнд – число без знака, второй – со знаком.
19	Сложение 9C5A8600h с 5-разрядной положительной константой (выбрать по своему усмотрению).
20	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды числа без знака.
21	Вычитание знаковых чисел 78B05D03h - 9C5A8600h.
22	Умножение 16×16 старших бит чисел 7100A000h и 8100C000h, где операнды знаковые числа.
23	Сложение знаковых чисел 9C5A8600h и A8B05D03h.
24	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число со знаком, второй – без знака.
25	Вычитание чисел без знака 78B05D03h - 9C5A8600h.
26	Умножение 16×16 младших бит чисел 9A000h и 7C000h, где первый операнд – число без знака, второй – со знаком.
27	Сложение знаковых чисел 9C5A8600h и 78B05D03h.

N	Содержание и условия операции
28	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды числа без знака.
29	Умножение 16×16 младших бит чисел 109A00h и C0081h, где операнды знаковые числа.
30	Умножение 16 младших бит числа 9100A000h на 16 старших бит числа 8100C000h, где первый операнд – число со знаком, второй – без знака.

Операция 3 заключается в вычислении абсолютной величины содержимого R1 (для регистровой пары использовать только четный регистр) с размещением результата в регистре R2 (назначить из РОН по своему усмотрению);

– условие и способ формирования содержимого регистра условия R_{ус} (назначить из РОН по своему усмотрению) выбрать из таблицы 9.3.

Таблица 9.3.

Условие		
N	Условие ветвления	Способ формирования содержимого регистра условия R _{ус}
1	$A8 \geq A9$	Сравнение чисел без знака
2	0 в пяти старших разрядах B8	Сброс области бит
3	$A8 \geq 0$	Арифметический сдвиг вправо
4	B8 четное	Операция конъюнкции
5	$A8 < A9$	Сравнение знаковых чисел
6	$B8 = B9$	Операция «неравнозначность»
7	$A8 \geq 0$	Выделение старшего бита без расширения знаком
8	0 в пяти старших разрядах B8	Логический сдвиг вправо
9	$A8 > A9$	Сравнение чисел без знака
10	B8 нечетное	Сброс области бит
11	$A8 \geq 0$	Сравнение знаковых чисел
12	$B8 \neq B9$	Операция «неравнозначность»
13	$A8 \leq A9$	Сравнение знаковых чисел
14	0 в пяти старших разрядах B8	Выделение области бит с расширением знаком
15	$A8 \geq 0$	Сброс области бит
16	B8 четное	Сдвиг влево
17	$A8 \geq A9$	Сравнение знаковых чисел
18	В пяти старших разрядах	Выделение области бит без расширения

N	Условие ветвления	Способ формирования содержимого регистра условия R_{yc}
	$B8$ хотя бы одна 1	знаком
19	$A8 \geq 0$	Выделение области бит с расширением знаком
20	0 в пяти старших разрядах $B8$	Операция конъюнкции
21	$A8 < A9$	Сравнение чисел без знака
22	$B8$ нечетное	Выделение области бит без расширения знаком
23	$A8 \geq 0$	Логический сдвиг вправо
24	$B8 = B9$	Сравнение чисел
25	$A8 > A9$	Сравнение знаковых чисел
26	1 в пяти старших разрядах $B8$	Сравнение чисел без знака
27	$A8 \geq 0$	Операция конъюнкции
28	$B8$ четное	Выделение области бит с расширением знаком
29	$A8 \leq A9$	Сравнение чисел без знака
30	$B8 \neq B9$	Сравнение чисел

В таблице имена регистров РОН совпадают с их содержимым.

- Получить исполняемый программный модуль.
- Загрузить исполняемый модуль в симулятор.
- В пошаговом режиме выполнить прогон программы, сравнивая данные прогноза с соответствующими данными окна CPU симулятора.
- Предъявить результаты выполнения работы преподавателю, после чего завершить работу с симулятором.

Методические указания

Алгоритмическая структура, приведенная на рис. 6, используется при программировании на языках высокого уровня. Однако в языке ассемблера,

Во-первых, возможны только два признака ветвления – содержимое регистра условия R_{yc} нулевое (выполняются операции одной ветви) или ненулевое (выполняются операции другой ветви). Для формирования этих

признаков требуются дополнительные логические и/или сервисные операции, что должно быть отражено в алгоритмической структуре (рис. 9.2).

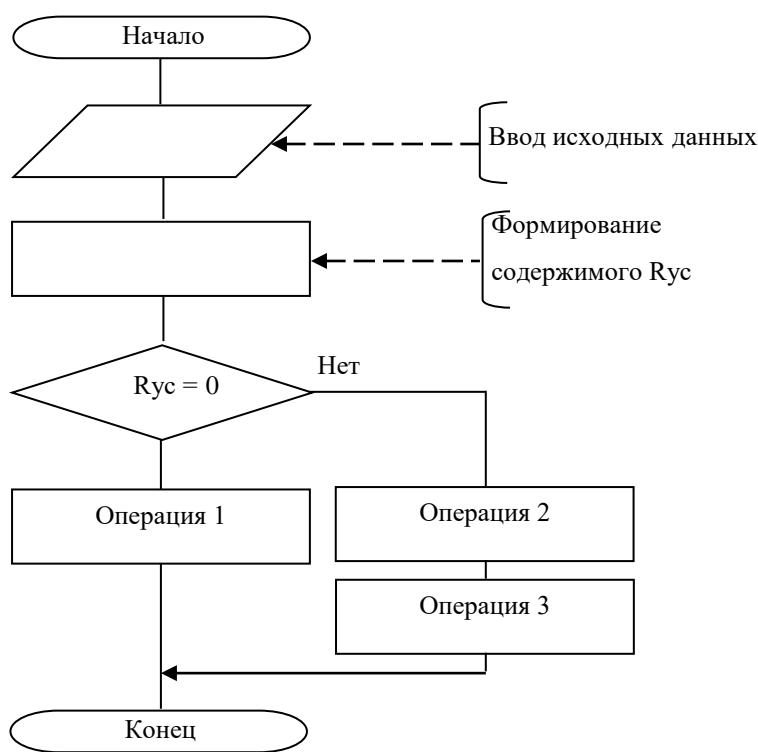


Рис. 9.2 – Обобщенная структура ветвления, приведенная к ассемблеру

При этом для одного и того же исходного условия признаки ветвления могут быть получены разными способами. Например:

1. В случае отношений $a \geq 0$ либо $a < 0$ помимо команд сравнения (обсуждаются ниже) можно использовать знак числа (значение 31-го двоичного разряда). Для его выделения пригодны конъюнкция (команда AND) с числом 80000000h, выделение области из одного старшего бита (команда EXT или EXTU), сдвиг вправо на 31 разряд (команда SHR или SHRU), обнуление 31 младших разрядов (команда CLR).

2. Для определения четности числа можно использовать его конъюнкцию с числом 00000001h, выделение области из одного младшего бита, сдвиг влево на 31 разряд, обнуление 31 старших разрядов.

Указанная множественность решений отражена в разделе «Способ

формирования содержимого регистра условия R_{yc} » таблицы задания, где для одного и того же условия в зависимости от варианта предлагаются различные команды его реализации.

Во-вторых, команды сравнения ассемблера реализуют только строгие отношения: $a < b$, $a > b$ и $a = b$. Поэтому в случае условий $a \leq b$, $a \geq b$ и $a \neq b$ следует использовать альтернативные условия: $a > b$, $a < b$ и $a = b$, соответственно. При этом признаки выполнения ветвей «Да» и «Нет» блока ветвления меняются местами. Например, некоторая операция должна выполняться, только если $a \leq b$ (признак «Да»). При использовании условия $a > b$ заданная операция должна выполняться только в случае его нарушения, то есть признак «Да» заменяется признаком «Нет» и наоборот.

В-третьих, все команды, соответствующие операциям той или иной ветви, должны быть условными. В противном случае команды одной из ветвей будут выполняться всегда, то есть вне зависимости от условия ветвления.

Обобщенное решение представленного задания:

Пример обобщенных алгоритма и программы (без указания конкретных данных) приведен на рис. 9.3.

При отладке требуется два прогона программы. В первом прогоне задается такое содержимое РОН, которое соответствует выполнению условия. Во втором прогоне содержимое одного из этих РОН изменяется так, чтобы условие нарушалось.

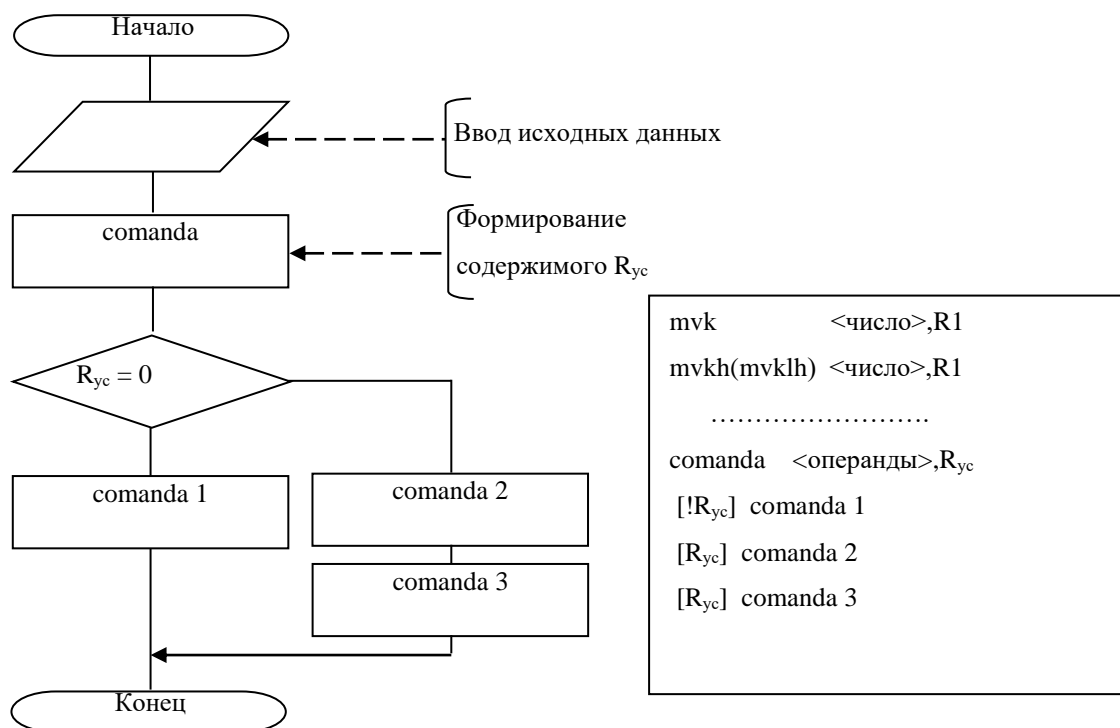


Рис. 9.3.Обобщенная структура ветвления, приведенная к ассемблеру

Примечание: при выполнении некоторых команд ассемблера происходит дополнение старших разрядов знаком и если число отрицательное в этих разрядах появятся единицы, что может привести к неверному результату.

Контрольные вопросы

1. Укажите регистры РОН, допустимые для использования в качестве регистра условия.
2. Сформулируйте особенности реализации нестрогих отношений в условии ветвления.
3. Поясните общие принципы организации ветвлений при программировании на языке ассемблера.
4. С пояснениями приведите примеры поля условия строки ассемблера.

5. Приведите формат логической команды, заданной преподавателем.
6. Приведите формат сервисной команды, заданной преподавателем.
7. Определите результат выполнения логической команды, заданной преподавателем.
8. Определите результат выполнения сервисной команды, заданной преподавателем.
9. Приведите программную реализацию ветвления по любому другому варианту.

Практические занятия

Блок практических заданий состоит из 5 работ предусмотрены задания на самостоятельное рассмотрение. Практические задания способствуют усвоению материала.

Практическая работа 1

Элементарные ФАЛ. Законы и тождества алгебры логики

1. Перевести в двоичную систему счисления десятичное число 55; 43.
2. Перевести в десятичную систему счисления двоичное число 11001; 11100.
3. Каков диапазон десятичных чисел, обрабатываемых цифровой системой при длине входных двоичных наборов 5?
4. Определить длину двоичных наборов, необходимую для обработки системой десятичных чисел от 12 до 25?
5. На рис. 1.1 представлены элементы

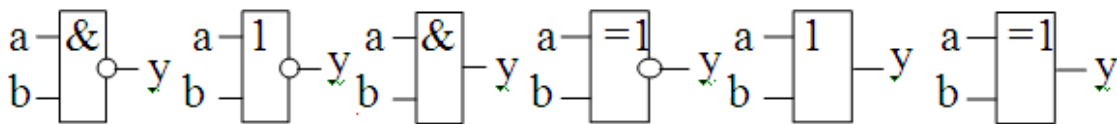
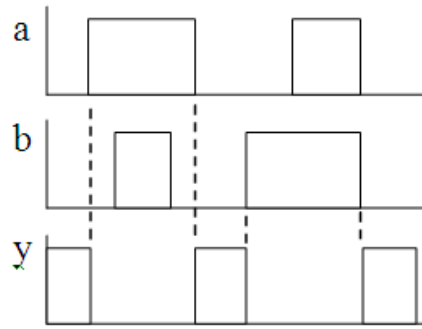
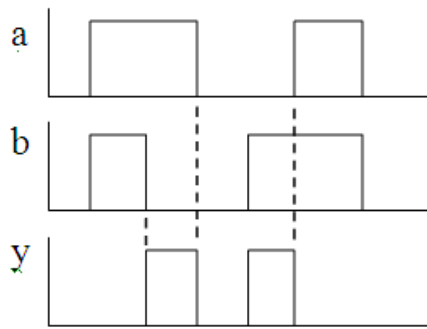


Рис.1.1 – Логические элементы

- назвать каждый из логических элементов;
- записать элементарную ФАЛ, реализуемую каждым из логических элементов;
- для элементов 1, 2, 4 и 6 записать сложную ФАЛ;
- какие из элементов можно включить в режиме инвертора и как?

6. По временным диаграммам установить тип логического элемента.



7. Используя законы и тождества логики, доказать равенство:
 $a \oplus b = \bar{a}\bar{b} \vee ab$.

8. Используя законы и тождества алгебры логики преобразовать следующие ФАЛ к элементарной:

$$y = \overline{\bar{a} \vee \bar{b}} \wedge a \vee b \wedge \overline{\bar{a} \wedge \bar{b}};$$

$$y = \overline{(\bar{a} \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee \bar{b} \vee c) \vee \bar{a} \wedge \bar{a} \wedge b};$$

$$y = \overline{\overline{\overline{abba}}}.$$

9. Используя законы и тождества алгебры логики преобразовать следующие ФАЛ к ФАЛ с двумя элементарными операциями:

$$y = \overline{a \wedge (b \vee \bar{c}) \vee \bar{b} \vee c};$$

$$y = \overline{a \vee \bar{b} \wedge c \vee \overline{\bar{a} \wedge b \vee c}};$$

$$y = \overline{a \vee \bar{b} \vee \bar{c} \vee a \vee \bar{b} \wedge \bar{c}}.$$

Практическая работа 2

Минимизация ФАЛ методом карт Вейча-Карно и минимальные базисы

1. Для каждой из указанных карт Карно показать:

- исходную ФАЛ;
- области минимизации;
- минимальную ФАЛ.

1		1	~
	1	1	
	1	1	
~	1	1	~

		~	
0	0		~
~	0	~	~
		0	

1	~	~	1
	~	1	
~	~		
1	~	1	~

2. Методом карт Вейча-Карно минимизировать ФАЛ:

$F(n=3) = (0,1,2)$. Записать F_{\min} в базисе И-НЕ;

$F(n=4) = [2,3,9,8,(5,6,7,10,11,12,13,14,15)]$. Записать F_{\min} в базисе

ИЛИ-НЕ;

$F(n=4) = (6,8,11,14,15,[0,1,2,3,12,13])$. Записать F_{\min} в базисе И-НЕ;

$F(n=4) = [0,2,5,7,12,14,(8,9,10,11,13,15)]$. Записать F_{\min} в базисе

ИЛИ-НЕ.

3. Для заданных схемы и входных сигналов построить временную диаграмму изменения выходного сигнала (рис.2.1)

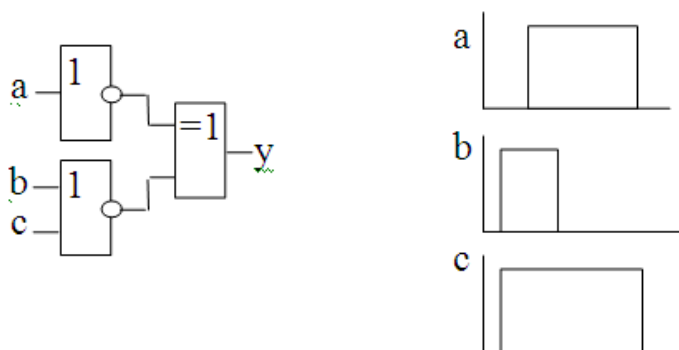


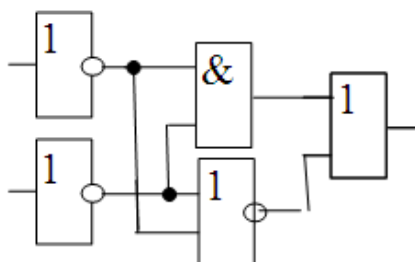
Рис.2.1 – Схемы и входных сигналов

Рекомендация: сначала записать ФАЛ $y = \bar{a} \oplus \overline{b \vee c} = \overline{ab \vee c} \vee \bar{a}(b \vee c)$. Затем переписать ее при $a = 0$ и при $a = 1$:

$$y = \begin{cases} b \vee c, & \text{если } a=0, \\ \overline{b \vee c}, & \text{если } a=1. \end{cases}$$

Теперь задача решается легко.

4. Упростить схему:



Рекомендация: записать ФАЛ, упростить, а затем построить новую схему.

Практическая работа 3

Синтез дешифратора и шифратора

Синтез дешифраторов произвольной разрядности

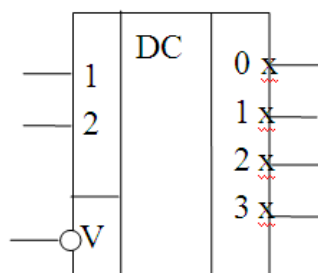
1. Какой выход дешифратора 4×16 будет в активном состоянии, если на его информационные входы подан двоичный код 1010; 0101.

2. Построить дешифратор 3×8 на базе дешифратора 2×4 с

а) инверсными выходами;

б) прямыми выходами.

Можно ли включить указанный дешифратор в режиме демультиплексора и если да, то как?



3. Синтезировать в полном базисе дешифратор 3×4 с:

а) инверсными выходами и безразличными входными наборами 1, 3, 5, 7;

б) прямыми выходами и безразличными входными наборами 0, 2, 4, 6.

Синтез шифраторов

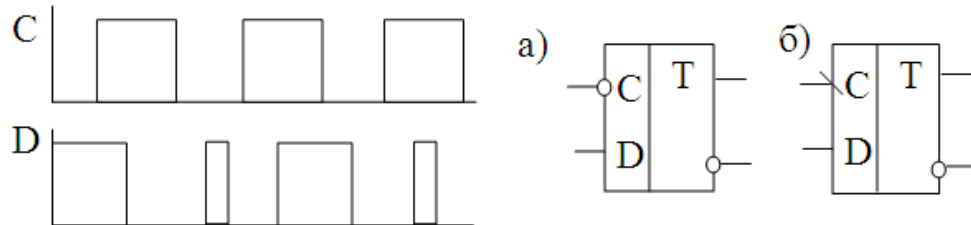
1. Синтезировать в полном базисе неприоритетный шифратор 4×2 с инверсными входами.

2. Синтезировать в полном базисе приоритетный шифратор 3×2 с прямыми входами, если двоичное слово 01 на его выходе не востребуется.

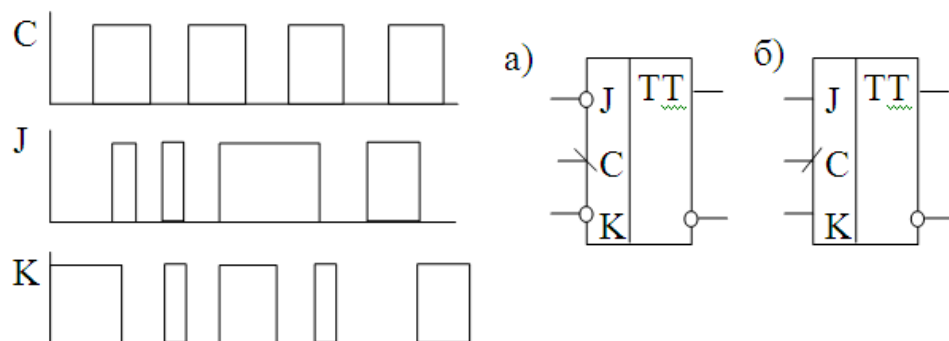
Практическая работа 4

Триггеры

1. Построить диаграмму изменения состояния триггера:



2. Построить диаграмму изменения состояния триггера:



4. Реализовать Т-триггер на базе JK-триггера (без использования и с использованием обратных связей).

5. Реализовать TV-триггер на базе D-триггера.

Практическая работа 5

Синтез двоичных счетчиков

1. На D-триггерах построить структурную схему и оценить быстродействие асинхронного суммирующего счетчика по модулю 5. ($t_3^T = 40$ нс, $t_3^{ЛЗ} = 10$ нс)

2. На базе микросхемы K155ИЕ7 реализовать вычитающий счетчик по модулю 10 при начальном состоянии а) 8; б) стандартном. В первом случае оценить быстродействие счетчика, если по входам «С» и «L» $t_3 = 40$ нс.

3. На базе микросхемы K155ИЕ5 реализовать суммирующий счетчик по модулю 11 и оценить его быстродействие.

4. На JK-триггерах построить структурную схему и оценить быстродействие синхронного суммирующего счетчика с последовательным переносом по модулю 9.

Приложение

Система команд TMS320C6x для чисел с фиксированной запятой

Условные обозначения в описании команд:

1. R1 – первый операнд, R2 – второй операнд, R3 – третий операнд, Rsm – смещение, Rbas – регистр адреса, Rres – регистр результата, cstn – n-разрядная константа.

Буква R символизирует как имя регистра общего назначения, так и его содержимое. Константа – n-разрядное целое, непосредственно указываемое в команде.

2. Тип операнда обозначается аббревиатурой: int – 32-разрядное целое (слово), short – короткое (16-разрядное) целое (полуслово), long – длинное (40-разрядное) целое.

Перед каждой аббревиатурой типа буква s означает величину со знаком (число представляется в дополнительном коде), буква u – величину без знака (число представляется в прямом коде), а буква x – что операнд исходит из регистра общего назначения противоположной стороны.

3. Слот задержки (slot) – интервал времени (число тактов) между началом выполнения команды и моментом времени, когда результат становится доступным для чтения.

Например, если slot = 1 и команда объявлена в i-м такте, то результат ее выполнения пишется в Rres в (i + 1)-м такте, а воспользоваться этим результатом можно лишь в (i + 2)-м такте.

Команды пересылки данных

MV R1,Rres Перемещение из одного регистра в другой

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	xsint	нет	sint	1-тактная	0
.D1 или .D2	sint	нет	sint		
.L1 или .L2	slong	нет	slong		

MVK R1,Rres Перемещение 16-разрядной константы в пределах регистра и расширение знаком

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	scst16	нет	sint	1-тактная	0

Описание: 16-разрядная константа размещается в Rres, свободные старшие разряды которого заполняются знаком константы.

MVKH, MVKLN R1,Rres Перемещение 16-разрядной константы в старшие разряды регистра

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	scst16	нет	sint	1-тактная	0

Описание: 16 старших (MVKH) или 16 младших (MVKLN) разрядов константы загружаются в старшие разряды Rres. 16 младших разрядов Rres остаются неизменными.

Команды загрузки/хранения

LDB, LDBU, LDH, LDHU, LDW R1,Rres Загрузка из памяти с 5-разрядной беззнаковой константой или с регистром смещения

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.D1 или .D2	См. синтаксис R1	нет	нет	Содержимое ячейки памяти	Загрузка	4

Описание: каждая из этих команд загружает из внутренней памяти данных в регистр Rres общего назначения. Синтаксис R1, определяющий правило формирования адреса:

$*+Rbas[Rsm]$ или $*+Rbas[ucst5]$ – положительное смещение. К Rbas добавляется Rsm или ucst5, причём Rbas не изменяется. Исполнительным адресом является результат суммирования;

$*-Rbas[Rsm]$ или $*-Rbas[ucst5]$ – отрицательное смещение. Из Rbas вычитается Rsm или ucst5, причём Rbas не изменяется. Исполнительным адресом является результат вычитания;

$*++Rbas[Rsm]$ или $*++Rbas[ucst5]$ – преинкремент. К Rbas добавляется Rsm или ucst5, изменяя Rbas. Исполнительным адресом является изменённое содержимое Rbas;

$*-Rbas[Rsm]$ или $*-Rbas[ucst5]$ – предекремент. Из Rbas вычитается Rsm или ucst5, изменяя Rbas. Исполнительным адресом является изменённое содержимое Rbas;

$*Rbas++[Rsm]$ или $*Rbas++[ucst5]$ – постинкремент. К Rbas добавляется Rsm или ucst5, изменяя Rbas. Исполнительным адресом является содержимое Rbas до его изменения;

$*Rbas--[Rsm]$ или $*Rbas--[ucst5]$ – постдекремент. Из Rbas вычитается Rsm или ucst5, изменяя Rbas. Исполнительным адресом является содержимое Rbas до его изменения.

Если смещение (Rsm или ucst5) не задаётся, ассемблер назначает нулевое смещение, а инкременты и декременты равны 1.

Rbas и Rsm должны быть на той же стороне ЦПУ, что и используемое устройство .D.

Rsm и cst5 до операции по формированию адреса сдвигаются влево на 0 (LDB и LDBU), 1 (LDH и LDHU) или 2 (LDW) разряда.

Адресная арифметика (сложение или вычитание) по умолчанию выполняется в линейном способе. Однако для A4 – A7 и B4 – B7 способ может быть изменён на циклический путём записи соответствующей величины в регистр AMR.

Для команд LDB(U) и LDH(U) загружаются только младшие 8 и 16 бит, соответственно, а остальные разряды Rres заполняются знаком (LDB и LDH) или нулями (LDBU и LDHU). Для LDW заполняются все 32 разряда Rres.

LDB, LDBU, LDH, LDHU, LDW R1,Rres Загрузка из памяти с 15-разрядной беззнаковой константой смещения

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.D2	*+B14[ucst15] или *+B15[ucst15]	нет	нет	Содержимое ячейки памяти	Загрузка	4

Описание: каждая из этих команд загружает из внутренней памяти данных в регистр Rres общего назначения. До операции по формированию адреса (вычитание не поддерживается) cst15 сдвигается влево на 0 (LDB и LDBU), 1 (LDH и LDHU) или 2 (LDW) разряда.

Адресная арифметика всегда выполняется в линейном способе.

Для команд LDB(U) и LDH(U) загружаются только младшие 8 и 16 бит, соответственно, а остальные разряды Rres заполняются знаком (LDB и LDH) или нулями (LDBU и LDHU). Для LDW заполняются все 32 разряда Rres.

STB, STH, STW R,*R1 Загрузка в память с 5-разрядной беззнаковой константой смещения или с регистром смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.D1 или .D2	См. синтаксис R1	нет	нет	Хранение	0

Описание: каждая из этих команд загружает во внутреннюю память

данных из регистра R общего назначения. Синтаксис R1 и выполнение Rbas и Rsm такие же, как у команд LD.

STB, STH, STW R,*R1 Загрузка в память с 15-разрядной беззнаковой константой смещения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.D1 или .D2	См. синтаксис R1	нет	нет	Хранение	0

Описание: каждая из этих команд загружает во внутреннюю память данных из регистра R общего назначения. Синтаксис R1 и выполнение Rbas и Rsm такие же, как у команд LD.

Арифметические команды

ABS R1,Rres Абсолютная величина целого

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	xsint	нет	нет	sint	1-тактная	0

Описание: абсолютная величина R1 устанавливается в Rres.

ADD2 R1,R2,Rres Два 16-разрядных целых добавляются к старшей и младшей половинам регистра

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	sint	xsint	нет	sint	1-тактная	0

Описание: старшая и младшая половины R1 добавляются, соответственно, к старшей и младшей половинам R2. Перенос из младшей половины результата в старшую не производится. Результат устанавливается в Rres.

ADD R1,R2,Rres Сложение знаковых целых

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	sint sint xsint scst5 scst5	xsint xsint slong xsint slong	нет	sint slong slong sint slong	1-тактная	0
.S1 или .S2	sint scst5	xsint xsint		sint sint		
.D1 или .D2	sint ucst5	sint sint		sint sint		

Описание: R2 добавляется к R1 (для устройств .L и .S) или R1 добавляется к R2 (для устройств .D) и результат устанавливается в Rres.

ADDK cst,Rres Сложение целых с использованием знаковой 16-разрядной константы

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	scst16	Rres	нет	uint	1-тактная	0

Описание: 16-разрядная знаковая константа добавляется к регистру Rres и результат устанавливается в Rres.

ADDU R1,R2,Rres Сложение беззнаковых целых

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	uint xuint	xuint ulong	нет	ulong ulong	1-тактная	0

Описание: R2 добавляется к R1 и результат устанавливается в Rres.

MPY, MPYU, MPYUS, MPYSU R1,R2,RresУмножение 16×16

младших бит знаковых или беззнаковых целых

Устройств о	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	s-мл.16 бит	xs-мл.16 бит	sint	Умножение 16 на 16	1
	u-мл.16 бит	xu-мл.16 бит	uint		
	u-мл.16 бит	xs-мл.16 бит	sint		
	s-мл.16 бит	xu-мл.16 бит	sint		
	scst5	xs-мл.16 бит	sint		
	scst5	xu-мл.16 бит	sint		

Описание: 16 младших бит R1 умножаются на 16 младших бит R2 и результат размещается в Rres. В команде MPY оба операнда знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYN, MPYNU, MPYNUS, MPYNSU R1,R2,RresУмножение $16 \times$

16 старших бит знаковых или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	s-ст.16 бит	xs-ст.16 бит	sint	Умножение 16 на 16	1
	u-ст.16 бит	x-уст.16 бит	uint		
	u-ст.16 бит	xs-ст.16 бит	sint		
	s-ст.16 бит	xu-ст.16 бит	sint		

Описание: 16 старших бит R1 умножаются на 16 старших бит R2 и результат размещается в Rres. В команде MPY оба операнда знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYHL, MPYHLU, MPYHULS, MPYHSLU R1,R2,Rres Умножение 16 старших бит на 16 младших бит знаковых или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	s-ст.16 бит	xs-мл.16 бит	sint	Умножение 16 на 16	1
	u-ст.16 бит	xu-мл.16 бит	uint		
	u-ст.16 бит	xs-мл.16 бит	sint		
	s-ст.16 бит	xu-мл.16 бит	sint		

Описание: 16 старших бит R1 умножаются на 16 младших бит R2 и результат размещается в Rres. В команде MPYHL оба операнда знаковые, а в команде MPYHLU – оба беззнаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

MPYLH, MPYLHU, MPYLUHS, MPYLSHU R1,R2,Rres Умножение 16 младших бит на 16 старших бит знаковых или беззнаковых целых

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.M1 или .M2	s-мл.16 бит	xs-ст.16 бит	sint	Умножение 16 на 16	1
	u-мл.16бит	x-уст.16 бит	uint		
	u-мл.16бит	xs-ст.16 бит	sint		
	s-мл.16бит	xu-ст.16 бит	sint		

Описание: 16 младших бит R1 умножаются на 16 старших бит R2 и результат размещается в Rres. В команде MPYLH оба операнда знаковые, а в команде MPYLHU – оба без знаковые. Позиция буквы S в мнемонике команды соответствует знаковому из двух сомножителей R1 и R2, другой из которых – без знака.

SUB, SUBU R1,R2,Rres Вычитание знакового или беззнакового целого без насыщения

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2	sint	xsint	sint	1-тактная	0
	xsint	sint	sint		
	sint	xsint	slong		
	xsint	sint	slong		
	uint	xuint	ulong		
	xuint	uint	ulong		
	scst5	slong	slong		
.S1 или .S2	sint	xsint	sint		
	scst5	xsint	sint		
.D1 или .D2	sint	sint	sint		
	sint	ucst5	sint		

Описание: R2 вычитается из R1 и результат размещается в Rres.

Логические команды

AND R1,R2,Rres Поразрядное И

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2, .L1 или .L2	uint	xuint	нет	uint	1-тактная	0
	scst5	xuint	нет	uint		

Описание: над R1 и R2 выполняется поразрядное И. Результат устанавливается в Rres. Константа распределяется на 32 разрядах.

CMPEQ R1,R2,Rres Сравнение знаковых целых на равенство

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	sint	xsint	нет	uint	1-тактная	0
	scst5	xsint		uint		
	xsint	slong		uint		
	scst5	slong		uint		

Описание: сравнение R1 с R2. Если R1 = R2, то в Rres записывается 1. В противном случае в Rres записывается 0.

CMPGT R1,R2,Rres Сравнение знаковых целых в отношении «больше»

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	sint	xsint	нет	uint	1-тактная	0
	scst5	xsint		uint		
	xsint	slong		uint		
	scst5	slong		uint		

Описание: сравнение R1 с R2. Если $R1 > R2$, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPGTU R1,R2,Rres Сравнение беззнаковых целых в отношении «больше»

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	uint	xuint	нет	uint	1-тактная	0
	ucst4	xuint		uint		
	xuint	ulong		uint		
	ucst4	ulong		uint		

Описание: сравнение R1 с R2. Если $R1 > R2$, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPLT R1,R2,Rres Сравнение знаковых целых в отношении «меньше»

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	sint	xsint	нет	uint	1-тактная	0
	scst5	xsint		uint		
	xsint	slong		uint		
	scst5	slong		uint		

Описание: сравнение R1 с R2. Если $R1 < R2$, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

CMPLTU R1,R2,Rres Сравнение беззнаковых целых в отношении

«меньше»

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.L1 или .L2	uint ucst4 xuint ucst4	xuint xuint ulong ulong	нет	uint uint uint uint	1-тактная	0

Описание: сравнение R1 с R2. Если $R1 < R2$, то в Rres записывается 1. В противном случае в Rres записывается 0. Если в команде константа указана на месте R2, то ассемблер сам переставит её на первое место.

NOP R1 Нет операции

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
нет	ucst4	нет	нет	NOP	0

Описание: R1 задаёт число тактов, в течение которых никакие операции (за исключением перехода) не производятся. NOP без операнда рассматривается как NOP 1.

NOT R1,Rres Поразрядное НЕ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	xuint	нет	uint	1-тактная	0

OR R1,R2,Rres Поразрядное ИЛИ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	uint scst5	xuint xuint	uint uint	1-тактная	0

Примечание: константа представляется в дополнительном коде и расширяется знаком вплоть до 32 бит.

SHL R1,R2,Rrest Арифметический сдвиг влево

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	xsint	uint	sint	1-тактная	0
	slong	uint	slong		
	xuint	uint	ulong		
	xsint	ucst5	sint		
	xuint	ucst5	ulong		
	slong	ucst5	slong		

Описание: операнд R1 загружается в Rrest, после чего перемещается влево на число разрядов, определённых константой R2. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если $39 < R2 < 64$, R1 перемещается влево на 40 разрядов. Освобождающиеся при сдвиге разряды заполняются нулями.

SHR R1,R2,Rrest Арифметический сдвиг вправо

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	xsint	uint	sint	1-тактная	0
	slong	uint	slong		
	xsint	ucst5	sint		
	slong	ucst5	slong		

Описание: операнд R1 загружается в Rrest, после чего перемещается вправо на число разрядов, определённых константой R2. Результат расширяется знаком. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если $39 < R2 < 64$, R1 перемещается вправо на 40 разрядов.

SHRU R1,R2,Rrest Логический сдвиг вправо

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	xuint	uint	uint	1-тактная	0
	ulong	uint	ulong		
	xuint	ucst5	uint		
	ulong	ucst5	ulong		

Описание: операнд R1 загружается в Rrest, после чего перемещается вправо на число разрядов, определённых константой R2. Результат расширяется нулями. Когда в качестве R2 используется регистр, величину сдвига определяют шесть его младших бит. Если $39 < R2 < 64$, R1 перемещается вправо на 40 разрядов.

XOR R1,R2,Rres Исключающее ИЛИ

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2	uint	xuint	uint	1-тактная	0
	scst5	xuint	uint		

Описание: поразрядное исключающее ИЛИ выполняется между R1 и R2. Результат устанавливается в Rres. Константа расширена знаком до 32 бит.

Команды перехода**B R1** Переход

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	метка	нет	нет	переход	Ветвление	5
.S2	xuint					

Сервисные команды**CLR R1, R2, R3, Rres** Очистить область бит беззнакового целого

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	uint	ucst15	ucst15	uint	1-тактная	0
	xuint	uint	нет	uint		

Описание: область в R1, заданная константами R2 и R3, сбрасывается в ноль, и результат устанавливается в Rres. Константа R2 определяет номер младшего бита (относительно нулевого разряда R1), а константа R3 – номер старшего бита (относительно нулевого разряда R1) области, которая должна очищаться. Константы могут быть определены десятью младшими битами регистра R2, где первая константа является битами с 0 по 4, а вторая – битами с 5 по 9.

EXT R1,R2,R3,Rres Выделение и расширение знаком области бит

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	sint	ucst5	ucst5	sint	1-тактная	0
	xsint	uint	нет	sint		

Описание: константами R2 и R3 определяется область в R1. Номер старшего бита области равен разности 31–R2 (первая константа), а номер младшего бита области – разности R3 (вторая константа)–R2 (первая константа). Константы могут быть определены как 10 младших бит регистра R2, где первая константа является битами с 5 по 9, а вторая – битами с 0 по 4. Выделенная область переносится в Rres на то же место, которое она занимает в R1, сдвигается влево на число разрядов, равное первой константе, а затем – вправо на число разрядов, равное второй константе. При сдвигах вправо освобождавшиеся разряды заполняются знаком (старшим разрядом области).

EXTU R1,R2,R3,Rres Выделение и расширение нулями области бит

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	uint	ucst5	ucst5	uint	1-тактная	0
	xuint	uint	нет	uint		

Описание: всё также как и в команде EXT, но при сдвигах вправо освобождавшиеся разряды заполняются нулём.

SAT R1,Rrest Округление 40-разрядного целого до 32-разрядного

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.S1 или .S2	slong	нет	sint	1-тактная	0

SET R1,R2,R3,Rrest Установка области бит

Устройство	Тип операндов			Тип результата	Тип команды	Слоты задержки
	R1	R2	R3			
.S1 или .S2	uint	ucst5	ucst5	uint	1-тактная	0
	xuint	uint	нет	uint		

Описание: область в R1, определенная константой R2 и константой R3, размещается в Rrest, после чего устанавливается во все единицы. Первая (R2) и вторая (R3) константы определяют (относительно нулевого разряда) номер, соответственно, младшего и старшего разряда области. Константы могут быть определены как десять младших бит регистра R2, причём биты 0 – 4 соответствуют второй константе, а биты 5 – 9 – первой.

ZERO Rres Обнуление регистра

Устройство	Тип операндов		Тип результата	Тип команды	Слоты задержки
	R1	R2			
.L1 или .L2, .S1 или .S2, .D1 или .D2	нет	нет	sint	1-тактная	0
.L1 или .L2	нет	нет	slong		

Предметный указатель

D-триггеры	52	ОСС.....	8
JK-триггер	50	ПДП	77
RS-триггер	50	ПЛМ.....	43
SDL.....	91	ПЦУ	46
БИС	60	РгК	64
БМУ	64	РОН.....	72
ДШ	64	СБИС	60
ЗЭ	47	СДНФ	20
ИВП.....	77	СКНФ	20
КЦУ.....	16	СУ	64
ЛСЭ	77	Т-триггер	50
ЛЭ.....	10	УП	64
МК.....	65	УсП	65
МП.....	59	УУз.....	64
МПК.....	60	ФАЛ.....	10
МПС	59	ЦУ	22
ОЗУ	64		

Список используемой литературы

1. Хлесткин, А.Ю. Машинно-зависимые языки программирования. [Текст]: учебное пособие/ А.Ю. Хлесткин, А.Г. Солодов, Т.А. Коваленко— Саратов, изд-во Десятая Муза, 2016.
2. Солодов, А.Г. Программирование на языке ассемблера процессора TMS320C6x [Текст]: методические указания/ А.Г. Солодов, А.М. Стефанов, – Самара, ПГУТИ, 2014.
3. Коваленко, Т.А., Солодов А.Г., Вычислительная техника и информационные технологии. [Текст]: учебное пособие/ Т.А. Коваленко, А.Г. Солодов - LA LAMBERT Academic Publishing GmbH & Co. KG ISBN:978-3-659-97255-3, 2016.
4. Захаров, Н. Г. Вычислительная техника [Текст]: учебник / Н. Г. Захаров, Р. А. Сайфутдинов. - Ульяновск: УлГТУ, 2007.
5. Угрюмов, Е. Цифровая схемотехника [Текст]/ Е. Угрюмов – С-Петербург, изд-во «БХВ-Петербург», 2002.
6. Уэйкерли, Дж.Ф. Проектирование цифровых устройств, т.1, 2 [Текст]/ Дж.Ф. Уэйкерли Пер. с англ. – М.: ПОСТМАРКЕТ, 2002.
7. Солонина А. Алгоритмы и процессоры цифровой обработки сигналов [Текст]/ А. Солонина, Д. Улахович, Л. Яковлев - С-Петербург, изд-во «БХВ-Петербург», 2001.
8. Цифровая и вычислительная техника: Учебник для вузов [Текст] / Э.В. Евреинов, Ю.Т и др.; Под ред. Э.В. Евреинова. - М.: Радио и связь, 1991.
9. Вычислительные системы, сети и телекоммуникации: Учебник. – 2-е изд., перераб. И доп. [Текст] / А.П. Пятибратов, и др; Под ред. А.П. Пятибратова – М.: Финансы и статистика, 2004.
10. Основы современных компьютерных технологий: Учебник [Текст] / Под ред. проф. А.Д. Хомоненко. – СПб.: КОРОНА принт, 2005.
11. Алексеев, А.П. Информатика 2015: учебное пособие [Текст] / А.П.

Алексеев– М: СОЛОН-Пресс, 2016. – 104 с.

12. Акчурин, Э. А. Имитационное моделирование канала связи с использованием нечеткой логики [Текст] / Э. А. Акчурин, Т. А. Коваленко // Информационно-вычислительные технологии и их приложения : сб. науч. тр. XI Междунар. науч.-техн. конф. – Пенза : Изд-во РИО ПГСХА, 2009. - С. 3-5.
13. Дьяконов, В. П. MATLAB 6.5 SP1/7+Simulink 5/6 в математике и моделировании [Текст] / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2005. - 472 с.
14. Дьяконов, В. П. MATLAB 6.5 SP 1/7/7 SP1/7 SP2 Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики [Текст] / В. П. Дьяконов, В. В. Круглов. – М. : СОЛОН-ПРЕСС, 2006. – 195 с.
15. Математическая модель А. Н. Тихонов [Электронный ресурс] / книги БСЭ Режим доступа: <http://slovari.yandex.ru/~книги/БСЭ/> Свободный – Загл. с экрана
16. Книги БСЭ [Электронный ресурс] Режим доступа: http://twi.mpei.ac.ru/ochkov/VPU_Book_New/mas/index.html Свободный – Загл. с экрана

«ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЯЗЫКИ ПРОГРАММИРОВАНИЯ» Часть 2

Учебное пособие

Коваленко Татьяна Анатольевна

Солодов Александр Геннадьевич

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И
МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Поволжский государственный университет

телекоммуникаций и информатики»

443010, г. Самара, ул. Льва Толстого 23

Подписано в печать 2022 г. Формат 60 x 84/16

Бумага офсетная №1. Гарнитура Таймс. Заказ 1001088. Печать
оперативная. Усл. печ. л. 4,33. Тираж 20 экз

Отпечатано в издательстве учебной и научной литературы
Поволжского государственного университета телекоммуникаций и
информатики 443090, г. Самара, Московское шоссе 77, т. (846) 228-00-44