

Министерство науки и высшего образования Российской Федерации

Томский государственный университет  
систем управления и радиотехники

А.И. Воронин

## **ЦИФРОВАЯ И МИКРОПРОЦЕССОРНАЯ ТЕХНИКА**

Методические указания к практическим занятиям для  
студентов направления подготовки 11.03.04  
"Электроника и микроэлектроника"

Томск  
2018

УДК 321.3  
ББК 32.853  
В75

**Рецензент:**

Бородин К.В., доцент кафедры промышленной электроники ТУСУР,  
канд. техн. наук.

Воронин, Александр Иванович

В75 Цифровая и микропроцессорная техника: Методические указания к практическим занятиям для студентов направления подготовки 11.03.04 "Электроника и наноэлектроника" / А.И. Воронин. – Томск: Томск. гос. ун-т систем упр. радиоэлектроники, 2018.– 45 с.

Настоящее учебно-методическое пособие по практическим занятиям составлено с учетом требованиям федерального государственного образовательного стандарта высшего образования (ФГОС ВО).

Учебное-методическое пособие ориентировано на студентов очной формы обучения направления подготовки 11.03.04 "Электроника и наноэлектроника" кафедры промышленной электроники и содержит примеры решения задач, примеры контрольных работ по дисциплине "Цифровая и микропроцессорная техника".

Одобрено на заседании каф. ПрЭ протокол № 3 от 21.03.18

УДК 321.3  
ББК 32.853

© Воронин А.И., 2018  
© Томск. гос. ун-т. систем  
упр.и радиоэлектроники

## СОДЕРЖАНИЕ

1 Введение.....	4
2 Примеры решения задач по цифровой схемотехнике.....	5
3 Примеры решения задач по микропроцессорной технике.....	12
4 Примеры контрольных работ по цифровой схемотехнике.....	21
5 Примеры контрольных работ по микропроцессорной технике.....	23
6 Список рекомендуемой литературы .....	27
7 Приложение А (справочное). Условно-графическое обозначение микросхем.....	28
8 Приложение Б (справочное). Система команд МК51.....	35

## 1 Введение

Дисциплина "Цифровая и микропроцессорная техника" студентами очной формы обучения изучается в трех учебных семестрах. В первом семестре обучения рассматриваются вопросы синтеза цифровых устройств на "жесткой" логике, когда функциональность устройства жестко связана со схмотехникой устройства. Во втором и третьем семестре рассматриваются вопросы проектирования устройств на программируемой логике, а именно, на микроконтроллерах.

### **Цели изучения дисциплины:**

Формирование навыков схмотехнического проектирования цифровых устройств на "жесткой логике" и программируемой логике, в том числе с применением микропроцессорных устройств. Сформировать у студентов следующие компетенции: ОПК3, ПКС-5, ПКС-6, ПКР-3.

### **Задачи дисциплины:**

1. Формирование знаний о предмете, принципах, современных и перспективных направлениях, математическом аппарате цифровой схмотехники.
2. Формирование знаний о назначении, характеристиках и параметрах цифровых микросхем.
3. Выработка у обучающихся навыков синтеза, анализа комбинационных и последовательностных цифровых устройств.
4. Формирование знаний об архитектуре микропроцессоров, навыков программирования и отладки программ для микропроцессоров на языке Ассемблер и языках высокого уровня.

### **Список компетенций:**

**ОПК-3:** Способен применять методы поиска, хранения, обработки, анализа и представления в требуемом формате информации из различных источников и баз данных, соблюдая при этом основные требования информационной безопасности.

**ПКР-3:** Способен выполнять расчет и проектирование электронных приборов, схем и устройств различного функционального назначения в соответствии с техническим заданием с использованием средств автоматизации проектирования.

**ПКС-5:** Способен учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности.

**ПКС-6:** Способен разрабатывать проектную и техническую документацию, оформлять законченные проектно-конструкторские работы.

### **Наименование компетенций:**

**ОПК** – общепрофессиональные компетенции.

**ПКР, ПКС** – профессиональные компетенции, рекомендуемые или определяемые вузом самостоятельно.

## 2 Примеры решения задач по цифровой схемотехнике

**Задача 1.** В приведенном ниже списке интегральных микросхем укажите номера цифровых микросхем комбинационного типа.

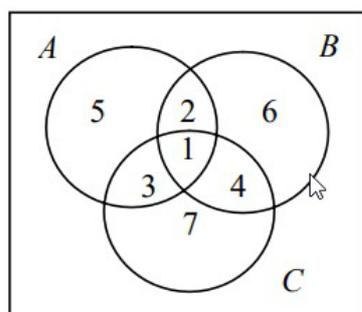
1	K555ИМ3	6	K1533ИЕ6
2	K133ТМ2	7	K531ИД3
3	K142ЕН5	8	K1554ИР24
4	K537РУ8	9	K1561КП1
5	K556РТ5	10	K140УД20

**Ответ:** 1 5 7 9. Указаны микросхемы сумматора, ПЗУ, дешифратора и мультиплексора. Кроме них в списке приведены обозначения двух аналоговых микросхем (стабилизатора постоянного напряжения и операционного усилителя) и цифровых микросхем последовательностного типа ( $D$ -триггера, ОЗУ, счетчика и регистра).

**Задача 2.** Записать в виде восьмиразрядного двоичного числа со знаком дополнительный код числа минус 35.

**Ответ:** 11011101. Он соответствует двоичному коду числа  $256 - 35 = 221$ .

**Задача 3.** Указать сегмент диаграммы Венна, которому



соответствует логическое выражение .

**Ответ:** 7. Это часть круга  $C$ , в которой надо исключить области, принадлежащие кругу  $A$  и кругу  $B$ . К аналогичным рассуждениям приводит и эквивалентное преобразование логического выражения:

$$C \cdot \overline{(A + B)} = C \cdot \overline{A} \cdot \overline{B}.$$

**Задача 4.** Указать логические соотношения, в которых допущена ошибка.

1.  $\overline{AB} \cdot \overline{BC} = \overline{B} + \overline{A} + \overline{C}$
2.  $(A+B)(A+C) = A+BC$
3.  $\overline{A} \oplus B = \overline{AB} \cdot (A+B)$
4.  $\overline{AB} + \overline{AC} = \overline{AB}(A+C)$
5.  $\overline{A} \oplus B = \overline{A} \oplus \overline{B}$
6.  $\overline{AB} + \overline{BC} = \overline{ABC}$

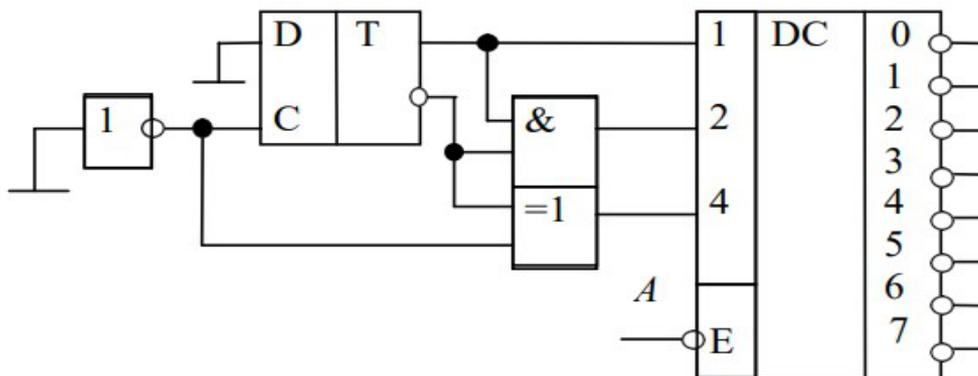
**Ответ: 3, 6.** Для доказательства справедливости представленных соотношений можно воспользоваться теоремами Булевой алгебры.

**Задача 5.** Указать значения булевой функции на восьми наборах таблицы истинности, соответствующих указанным на рисунке клеткам карты Карно ( $f_7 \dots f_0$ ).

	$A$			
	0	2	6	4
$C$	1	3	7	5
	$B$			

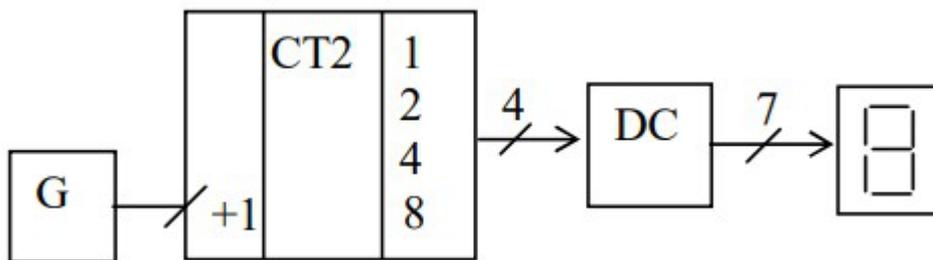
**Ответ:** 01101010. Блок дает 1 в клетке 6. Блок дает 1 в клетках 1 и 3. Блок заполняет единицами  $B$  клетки 1 и 5.

**Задача 6.** На каком выходе дешифратора повторяется сигнал  $A$ ?



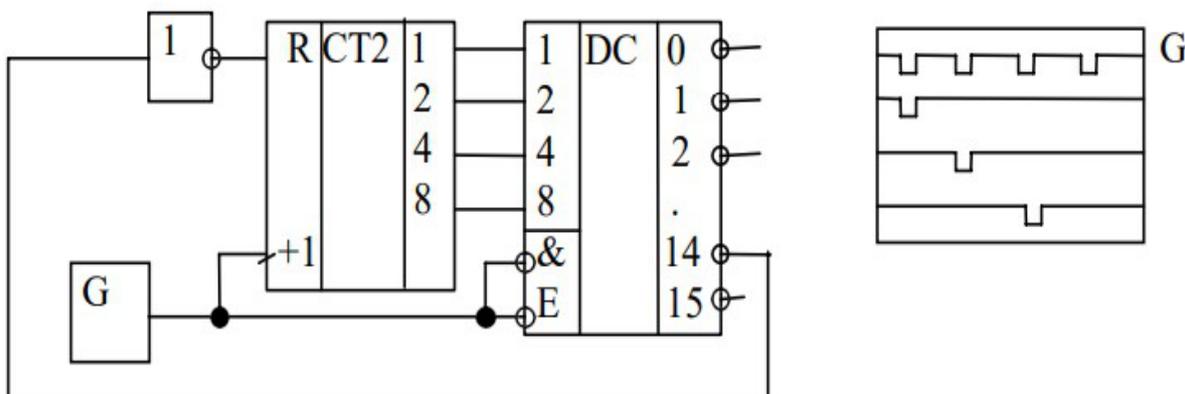
**Решение.** На вход  $C$   $D$ -триггера подана логическая 1. Следовательно, он работает как повторитель уровня, который подан на вход  $D$ . При этом на его прямом выходе — 0, инверсном — 1. На выходе логического элемента «Исключающее ИЛИ» формируется логический 0, так как уровни на входах одинаковые. Поскольку на всех адресных входах дешифратора (в данном случае он работает как демультиплексор) логические нули, входной сигнал  $A$  повторится на его нулевом выходе. На всех других выходах будет логическая 1.

**Задача 7.** Счетчик находился в состоянии 7, после чего на его вход поступило 125 импульсов. Какое число загорится на цифровом индикаторе?



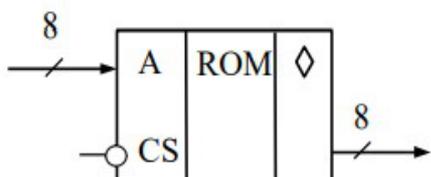
**Ответ:** 4. На схеме изображен четырехразрядный суммирующий двоичный счетчик с коэффициентом пересчета 16, меняющий состояния с 0 по 15. После поступления 16 импульсов на вход счетчика он снова окажется в 7-м состоянии. В этом же состоянии он будет через 112 импульсов (ближайшее целое число к 125, которое делится на 16). Еще через 13 импульсов он окажется в состоянии 4. Это число и загорится на цифровом индикаторе.

**Задача 8.** Оценить число каналов распределителя импульсов, показанного на рисунке?



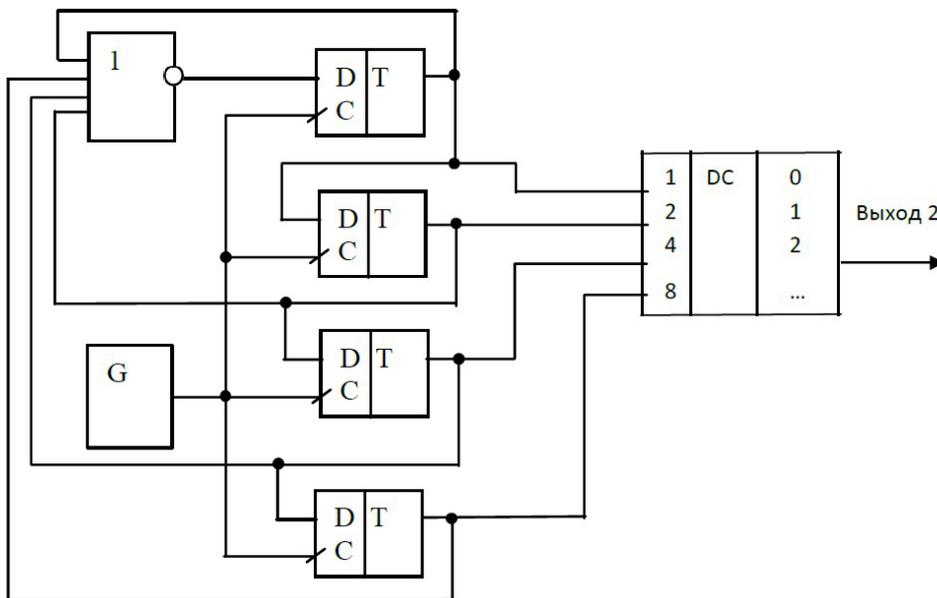
**Решение.** Как только суммирующий двоичный счетчик переходит в 14-е состояние (по фронту импульсов генератора  $G$ ), формируется логическая 1 на входе  $R$  и он сбрасывается в нулевое состояние. Таким образом, число каналов распределителя импульсов равно 14 (с 0-го по 13-й).

**Задача 9.** Указать емкость ПЗУ в битах.



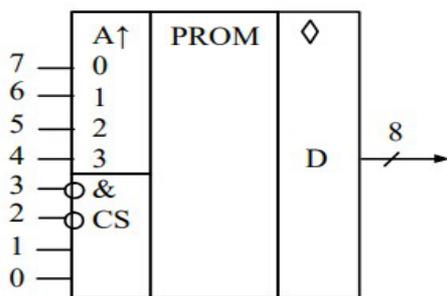
**Ответ:** 2048. Емкость ЗУ в битах определяется произведением количества хранящихся слов на разрядность. В данном ПЗУ хранится 256 восьмиразрядных слов.

**Задача 10.** Во сколько раз (указать число) частота выходных импульсов меньше частоты генератора.



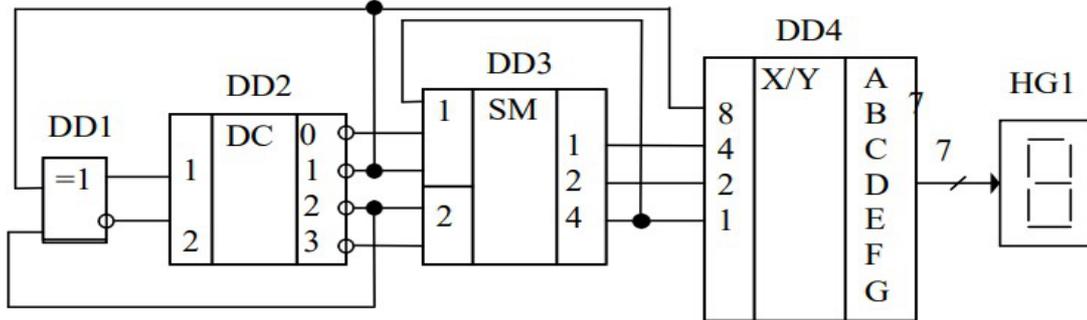
**Ответ: 5.** На рисунке показана схема кольцевого счетчика на регистре сдвига, к выходам которого подключен дешифратор. Коэффициент пересчета счетчика равен 5. По его пяти выходам при подаче импульсов генератора перемещается логическая 1 (пятый выход счетчика — это выход логического элемента ИЛИ-НЕ). Состояниям счетчика соответствует появление логической единицы на выходах дешифратора 0, 1, 2, 4 и 8. На выходе 2 частота импульсов будет в пять раз меньше частоты генератора. На некоторых других выходах, например третьем, импульсов не будет.

**Задача 11.** Указать уровни сигналов на входах ПЗУ при считывании информации из пятнадцатой ячейки.



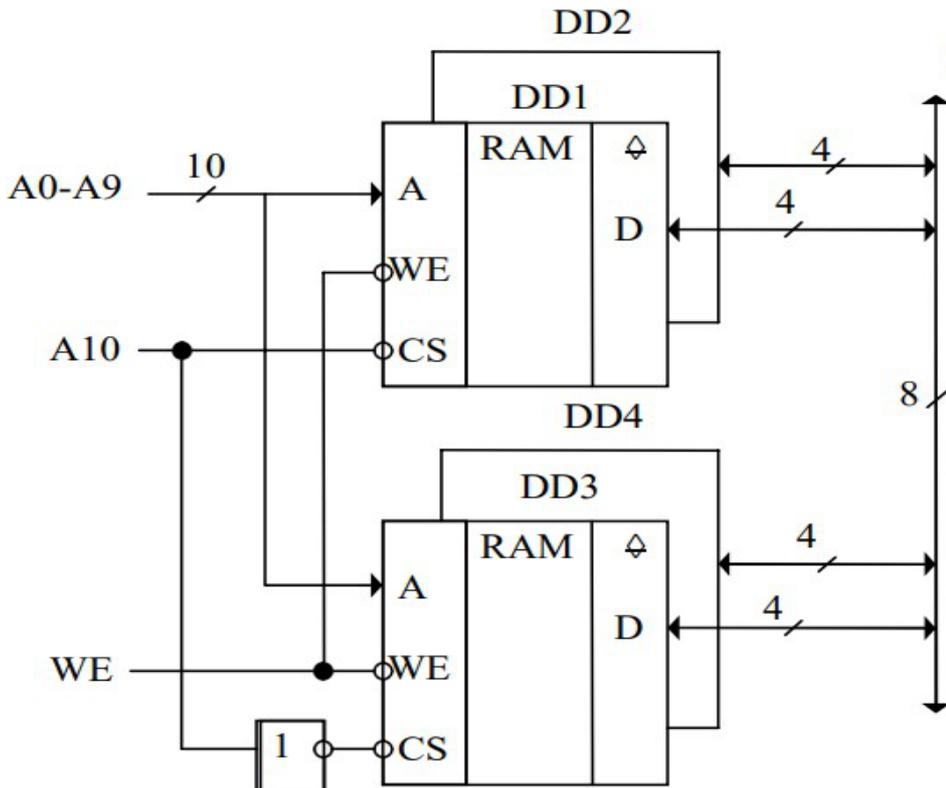
**Ответ: 11110011.** На рисунке приведена схема однократно программируемого ПЗУ емкостью 16 байт. Адрес ячейки (от 0 до 15) задается уровнями сигналов на адресных входах ПЗУ. Для считывания информации на двух нижних входах разрешения должны быть логические единицы, на двух верхних — логические нули.

**Задача 12.** Какое число загорится на цифровом индикаторе?



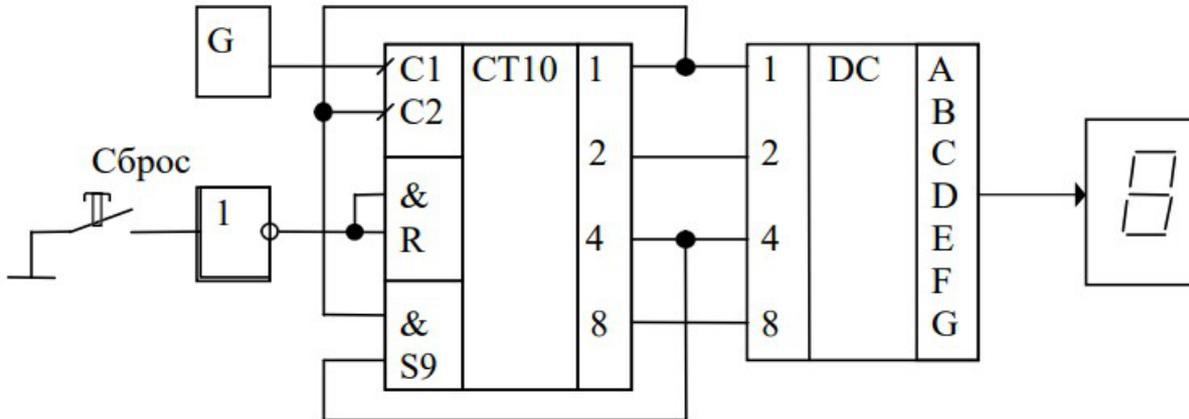
**Ответ:** 3. Анализируя схему, расставим логические уровни на входах и выходах элементов. На входах дешифратора уровни сигналов не совпадают. Следовательно, активным может быть либо первый, либо второй выход DD2. Во всяком случае, не совпадают и уровни сигналов на входах логического элемента DD1. Следовательно, на его прямом выходе — 1, инверсном — 0. При этом на всех выходах DD2, кроме первого, логические единицы. На входы DD3 поданы сигналы, сумма которых 5 или 6. Так как в любом из этих случаев по цепи обратной связи на вход младшего разряда сумматора поступает 1, то  $S = 6$  (логические единицы на выходах с весовыми коэффициентами 4 и 2). При этом на входах преобразователя DD4 логические уровни соответствуют коду цифры 3, которая и загорится на цифровом индикаторе HG1.

**Задача 13.** Организуйте ОЗУ 2К·8 на микросхемах K541PY2 (1К·4).



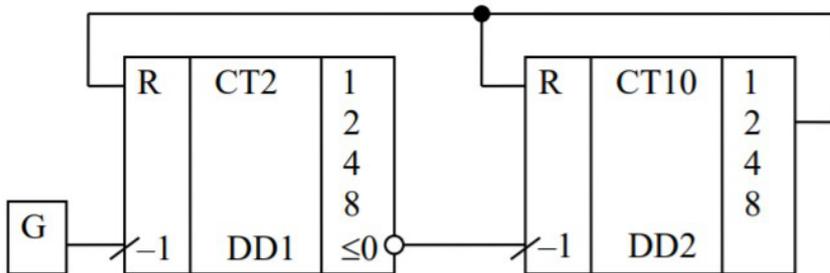
**Решение.** Для увеличения разрядности слов объединены все одноименные входы микросхем DD1, DD2 (и соответственно, DD3, DD4). При  $A_{10} = 0$  выбирается верхнее ОЗУ 1К·8, при  $A_{10} = 1$  — нижнее. Выходы микросхем связаны с восьмиразрядной двунаправленной шиной DB.

**Задача 14.** Какое число загорится на цифровом индикаторе после поступления на вход предварительно сброшенного счетчика ста импульсов?



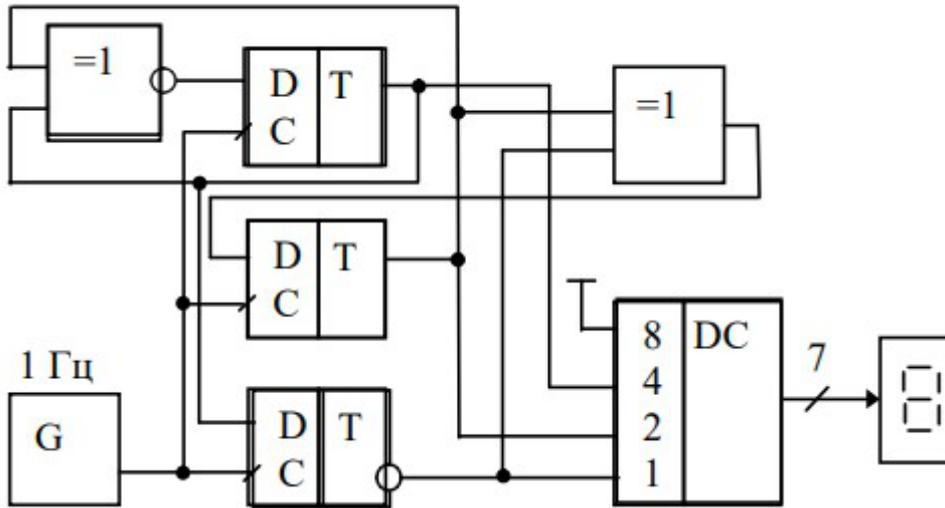
**Ответ:** 4. Микросхема (например, К155ИЕ2) работает как двоично-десятичный счетчик, считая в прямом направлении от нуля до девяти. Но из пятого состояния за счет обратных связей она перекидывается в девятое. Таким образом, в цикле реализуются состояния 9,0,1,2,3,4 и коэффициент пересчета счетчика равен 6. После 96 импульсов предварительно сброшенный счетчик будет находиться в нулевом состоянии, а еще через 4 импульса — в четвертом. Это число и загорится на индикаторе.

**Задача 15.** Определите коэффициент пересчета счетчика.



**Ответ:** 33. Первый каскад вычитающего счетчика собран на четырехразрядном двоичном счетчике DD1 (например, К155ИЕ7), второй — на двоично-десятичном счетчике DD2 (К155ИЕ6). Проведем анализ работы устройства при поступлении импульсов на вход предварительно обнуленного счетчика. Первый импульс, поступающий на счетный вход, повторяется на выходе заема ( $\leq 0$ ) DD1. По его положительному фронту микросхема DD1 переходит в 15-е состояние, микросхема DD2 — в 9-е. Последующие 15 импульсов будут менять состояние DD1, не меняя режим DD2. По окончании 17-го импульса DD1 перейдет в 15-е состояние, DD2 — в восьмое. Еще через 16 импульсов DD2 перейдет в седьмое состояние и появится логическая 1 на выходе 2, которая сбросит счетчик в нулевое состояние. Таким образом, коэффициент пересчета счетчика равен 33.

**Задача 16.** Записать последовательность чисел, которые загорятся в цикле на цифровом индикаторе.



**Решение.** Обозначим сигналы на входах дешифратора до подачи активного фронта тактового импульса весовыми коэффициентами 4, 2 и 1, после подачи тактового импульса —  $4^+$ ,  $2^+$  и  $1^+$ . Тогда логика смены состояний счетчика описывается системой уравнений:  $4^+ = 4 \oplus 2$ ;  $2^+ = 2 \oplus 1$ ;  $1^+ = \bar{4}$ .

Зафиксируем таблицу переходов после подачи очередного активного фронта тактового импульса  $n$ , предположив, что в исходном состоянии на индикаторе горит цифра  $N = 0$ .

$n$	4	2	1	$N$
0	0	0	0	0
1	1	0	1	5
2	0	1	0	2
3	0	1	1	3
4	0	0	1	1
5	1	1	1	7
6	1	0	0	4
7	0	0	0	0

Анализ смены состояний показывает, что в цикле семь состояний (все кроме шестого). Из шестого состояния счетчик переходит снова в шестое. Следовательно, у схемы два алгоритма работы. Если при включении или под действием помехи счетчик переходит в состояние 6, то оно в дальнейшем не меняется. Иначе реализуется цикл, зафиксированный в ответе.

**Ответ:** 0523174

### 3 Примеры решения задач по микропроцессорной технике

**Пример 1.** Заполнить массив 1100H-110FH внешнего ОЗУ данных константой со входов порта P1. Транслировать программу, начиная с адреса 100H.

```

1 0100          ORG      100H
2 0100 901100   MOV      DPTR,#1100H
3 0103 7910     MOV      R1,#16
4 0105 E590     MOV      A,P1
5 0107 F0       M1:     MOVX     @DPTR,A
6 0108 A3       INC      DPTR
7 0109 D9FC     DJNZ    R1,M1
8 010B         END

```

Для обращения к ВПД используется регистр указатель данных DPTR. Счетчик числа элементов массива выполнен на регистре R1. В четвертой строке программы второй байт берется равным прямому адресу порта P1. В седьмой строке второй байт равен относительному смещению от адреса следующей команды до адреса, соответствующего метке M1 (дополнительный код числа минус 4).

**Пример 2.** Произведение П цифр двухразрядного десятичного числа, находящегося в аккумуляторе в двоично-десятичном коде, возратить в аккумулятор также в двоично-десятичном коде. Транслировать программу, начиная с нулевой ячейки.

```

1 0000 75F010   MOV      B,#10H      ; Распаковка цифр числа
2 0003 84       DIV      AB        ; в регистры A и B
3 0004 A4       MUL      AB        ; Двоичный код П в A
4 0005 75F00A   MOV      B,#10      ; П делится на 10. A содержит
5 0008 84       DIV      AB        ; цифру десятков, B – остаток
6 0009 C4       SWAP     A          ; Цифры произведения в
7 000A 45F0     ORL      A,B        ; упакованном формате
8 000C         END

```

Сначала исходное число делится на 16 (процессор при выполнении команды деления считает, что содержимое аккумулятора соответствует двоичному числу). Старшая цифра числа попадает в A, младшая – в B. Затем в аккумуляторе формируется двоичный код их произведения. Далее делением на 10 реализуется преобразование произведения в двоично-десятичный формат. Второй байт команды в седьмой строке соответствует прямому адресу регистра B.

**Пример 3.** Скопировать массив РПД 20H-2FH на новое место, начиная с ячейки 40H.

```

MOV      R0,#20H      ; Начальный адрес первого массива
MOV      R1,#40H     ; Начальный адрес второго массива

```

	MOV	R2,#16	; Число элементов массива
M1:	MOV	A,@R0	; Пересылка очередного элемента
	MOV	@R1,A	; массива
	INC	R0	; Организация цикла
	INC	R1	; копирования
	DJNZ	R2,M1	; элементов массива

Для обработки элементов массива в цикле всегда удобно использовать косвенную адресацию, которая в данном примере реализуется с помощью регистров R0 и R1. Другие регистры общего назначения для этой цели использовать нельзя. Директива END в данном и последующих примерах опущена.

**Пример 4.** Наибольшее число массива 8-разрядных чисел без знака в РПД (20H-2FH) поместить в ячейку 30H.

MAX	EQU	30H	; Директива ассемблера
	MOV	R0,#20H	; Указатель памяти
	MOV	R1,#16	; Счетчик числа элементов
	MOV	MAX,#0	; Обнуление ячейки результата
M1:	MOV	A,@R0	; Сравнение очередного элемента
	CJNE	A,MAX,\$+3	; массива с ячейкой результата
	JC	M2	; Переход, если меньше или равно
	MOV	MAX,A	; Замена, если больше
M2:	INC	R0	; Организация цикла
	DJNZ	R1,M1	; просмотра элементов массива

С помощью директивы EQU ячейке резидентной памяти данных с адресом 30H присвоено символическое имя MAX, которое неоднократно используется в тексте программы, улучшая ее “читаемость”. Присвоение уникальных имен всем переменным, используемым при выполнении задачи – прием, широко используемый в практике программирования на языке ассемблера.

В данном примере результатом выполнения команды сравнения является установка или сброс флага переноса C. Команда тестирования этого флага выполняется не зависимо от того, равно содержимое аккумулятора содержимому ячейки MAX или нет.

**Пример 5.** Сравнить содержимое аккумулятора с константой 100 и выполнить следующие действия:

если A=100, то перейти на метку M1;

если  $A < 100$ , то перейти на метку M2;

если  $A > 100$ , то перейти на метку M3.

CJNE                    A,#100,\$+6

LJMP                    M1                    ; Переход, если  $A=100$

JC                    M2                    ; Переход, если  $A < 100$

M3:                    .....                    ; Выполнение условия  $A > 100$

Если содержимое аккумулятора не равно 100, то дополнительно тестируется флаг переноса C. Он устанавливается в единицу при выполнении условия  $A < 100$  и в ноль при  $A > 100$ .

**Пример 6.** Преобразовать двоичное число без знака, находящееся в аккумуляторе, в двоично-десятичное и поместить его в DPTR.

MOV                    B,#100                    ; Делим на 100 для  
DIV                    AB                    определения

; числа сотен

MOV                    DPH,A                    ;

MOV                    A,#10                    ; Делим на 10 для  
XCH                    A,B                    определения

; числа десятков

DIV                    AB

SWAP                    A                    ; Обмен полубайтов аккумулятора

ADD                    A,B                    ; Добавляем единицы

MOV                    DPL,A

**Пример 7.** Показать структуру построения программы, использующей аппаратное прерывание по фронту INT0.

```
ORG                    0                    ; Начало программы
SJMP                    MAIN                ; Переход к основной
ORG                    0003H                программе
; Вектор прерывания
MAIN:                    AJMP                    SUBR                ; Переход к п/п
MOV                    SP,#100                обслуживания
SETB                    EA                    ; Настройка указателя стека
; Сброс блокировки
; прерываний
SETB                    EX0                ; Разрешение прерывания
SETB                    IT0                INT0
; Бит прерывания по фронту
; Текст основной программы
.....
```

SUBR:	ORG	800H	; Начало п/п прерывания
	PUSH	PSW	; Сохранение в стеке
	PUSH	ACC	; содержимого
	PUSH	B	; регистров
	PUSH	DPL	;
	PUSH	DPH	;
	SETB	RS0	; Выбор первого банка РОН
	CLR	RS1	
	.....		; Текст подпрограммы
	POP	DPH	; Восстановление из стека
POP	DPL	; содержимого	
POP	B	; регистров	
POP	ACC	;	
POP	PSW	;	
RETI		; Возврат из п/п прерывания	

Предполагается регистры первого банка РОН использовать только в подпрограмме обслуживания аппаратного прерывания. Для этого необходимо модифицировать содержимое указателя стека, так как после сброса он настроен на область РПД, занимаемую банком РОН1.

Прерывание происходит всегда неожиданно для основной программы, поэтому при его обслуживании необходимо сохранить содержимое PSW и всех регистров, используемых подпрограммой. В данном примере предполагается использовать в подпрограмме регистры А, В и DPTR.

Текст подпрограммы обслуживания прерываний можно располагать в любом удобном месте программы. Так как при наличии запроса на прерывание управление передается ячейке с адресом 0003H, в этой ячейке записывается команда безусловного перехода к выбранному адресу подпрограммы обслуживания. Если после выполнения команды RETI на входе INTO сохраняется 0, подпрограмма обслуживания не будет выполняться повторно, так как установлен бит прерывания по фронту IT0.

**Пример 8.** На линии P1.0 – P1.3 поступают сигналы X, Y, Z и V от датчиков. Выдать на линию P1.4 этого порта сигнал в соответствии с логическим выражением  $F=X(Y+Z)+V$ .

1	0090	X	EQU	P1.0	
2	0091	Y	EQU	P1.1	
3	0092	Z	EQU	P1.2	
4	0093	V	EQU	P1.3	
5	0094	F	EQU	P1.4	
6	0000	A291		MOV	C,Y
7	0002	7292		ORL	C,Z
8	0004	8290		ANL	C,X
9	0006	A093		ORL	C,/

10 0008	9294	MOV	V F,C
11 000A		END	

Контроллер МК51 имеет битовый процессор и позволяет проводить логические операции и операции тестирования с отдельными битами. Прямую адресацию имеют 128 флагов пользователя в РПД и биты 11 регистров специальных функций. В РПД можно организовать карту опроса 128 датчиков и эффективно обрабатывать эту информацию с использованием команд битового процессора.

В контроллерах, не имеющих битового процессора (К580, МК48), каждая команда логической обработки бита требует загрузки байта в аккумулятор, выполнения команд логической обработки байтов, маскирования и команд условных переходов. Поэтому реализация булевых функций микроконтроллером МК51 осуществляется значительно проще и быстрее.

На языке ассемблера битовый операнд можно записывать, используя символические имена бита или регистра, либо прямые адреса бита или регистра. Вот примеры записи команды выбора первого банка РОН:

SETB	RS0	; Используется символическое имя бита
SETB	PSW.3	; Используется символическое имя регистра
SETB	0D0H.3	; Используется прямой адрес регистра
SETB	0D3H	; Используется прямой адрес бита

Независимо от формы записи команды второй байт при трансляции соответствует прямому адресу бита (D3).

Логические операции обработки битов реализуются с участием триггера переноса C, который для битовых операндов выполняет такую же роль, как аккумулятор в командах арифметических и логических операций с байтами.

**Пример 9.** Организовать задержку длительностью 50 мс на микроконтроллере К1830BE51 с использованием таймера, прерываний и режима холостого хода.

При использовании таймера в режиме 1 (кварц на 12 МГц) можно получать задержки до 65536 мкс. Включение и выключение таймера осуществляется установкой и сбросом бита TR0 (TCON.4). Чтобы переполнение таймера произошло через 50 мс, в его регистры необходимо загрузить дополнительный код числа 50000. Формирование дополнительного кода, загрузку младшего байта в TL0, а старшего в TH0, выполняет ассемблер.

Микроконтроллеры серии К1830, выполненные по КМОП- технологии, можно перевести в режим холостого хода установкой нулевого бита регистра PCON (IDL). В этом режиме блокируются функциональные узлы центрального процессора, что уменьшает энергопотребление до 4 мА. Сохраняется содержимое SP, PC, PSW, A и других регистров и РПД. Активизация любого разрешенного прерывания (а также аппаратный сброс микроконтроллера) заканчивает режим XX. После выполнения команды RETI будет исполнена команда, которая следует за командой, переведшей МК в режим холостого хода.

1 000B		ORG	000BH	; Вектор
			прерывания	
2 000B	C28C	CLR	TR0	; Останов T/C0

3 000D	32		RETI		; Возврат
4 0100			из п/п		
		ORG	100H		; Начало программы
5 0100	D2AF	MAIN:	SETB	EA	; Снятие
			блокировки		
6 0102	758901		MOV	TMOD,#01H	; Режим 1
			T/C0		
7 0105	758AB0		MOV	TL0,#LOW(NOT(50000)+1)	
8 0108	758C3C		MOV		
9 010B	D28C				
10 010D	D2A9		TH0,#HIGH(NOT(50000)+1)	SETB	
			TR0		; Старт T/C0
			SETB	IE.1	; Разрешение
			прерываний		
11 010F	758701		MOV	PCON,#01H	
Режим XX					
12 0112		NEXT:			
			.....		; Продолжение
			программы		

Установкой первого бита регистра PCON (PD) можно перевести МК в режим микропотребления (напряжение питания может быть уменьшено до 2 В, потребляемый ток – 50 мкА). В этом режиме прекращается работа всех узлов микроконтроллера, но сохраняется содержимое ОЗУ. Вывести МК из режима микропотребления можно только аппаратным сбросом (RST=1).

У микроконтроллеров серии K1816 используется только старший бит регистра PCON (SMOD). Установка этого бита удваивает скорость передачи при работе последовательного порта.

**Пример 10.** Подсчитать число импульсов, поступающих на вход T1 (P3.5) за заданный промежуток времени (10 мс). Текст программы разместить с адреса 1000H. Результат сформировать в DPTR.

1 D8F0		TIME	EQU	NOT(10000)+1
2 1000			ORG	1000H
3 1000	758951		MOV	TMOD,#01010001B
4 1003	E4		CLR	A
5 1004	F58D		MOV	TH1,A
6 1006	F58B		MOV	TL1,A
7 1008	758CD8		MOV	TH0,#HIGH(TIME)
8 100B	758AF0		MOV	TL0,#LOW(TIME)
9 100E	438850		ORL	TCON,#50H
10 1011	108D02		JBC	TF0,M2
	M1:			
11 1014	80FB		SJMP	M1
12 1016	858D83	M2:	MOV	DPH,TH1

```

13 1019    858B82          MOV    DPL,TL1
14 101C          END

```

Управляющее слово в третьей строке программы настраивает T/C0 на работу в режиме 16-битового таймера, T/C1 – в режиме 16-битового счетчика событий. В регистры таймера перед пуском загружается дополнительный код числа 10000, регистры счетчика событий обнуляются. Команда в девятой строке одновременно осуществляет старт T/C0 и T/C1. Счет импульсов происходит до переполнения T/C0, после чего содержимое T/C1 переписывается в DPTR. Команда JBC сбрасывает флаг TF0.

**Пример 11.** Разработать программу преобразования кода 10 элементов массива, находящегося в резидентной памяти микроконтроллера K1816BE51, из двоично-десятичного в двоичный (байтовая информация).

Считаем, что массив занимает ячейки памяти данных на кристалле с 50 по 59. Используем команду умножения байтов. Подробный комментарий помещен в текст программы.

```

MOV    R0,#50      ; Начальный адрес массива
MOV    R7,#10     ; Число элементов массива
M1:   MOV    A,@R0 ; Выделение числа
      SWAP  A      ; десятков
      ANL  A,#0FH
      MOV  B,#10   ; 10 в расширитель аккумулятора
      MUL  AB
      MOV  B,A
      MOV  A,@R0  ; Выделение единиц
      ANL  A,#0FH
      ADD  A,B     ; Формирование двоичного числа
      MOV  @R0,A  ; Замена элемента массива
      INC  R0     ; Нарастивание адреса
      DJNZ R7,M1 ; Цикл обработки массива

```

**Пример 12.** Сформировать на выходе WR импульс логического нуля длительностью 50 мкс.

Ниже приводится фрагмент программы, реализующий эту функцию, с указанием времени выполнения отдельных команд (при частоте кварца 12 МГц один машинный цикл составляет 1 мкс):

```

CLR    WR          ; 1 машинный цикл
MOV    R7,#24     ; 1 машинный цикл
DJNZ  R7,$        ; 2 машинных цикла
SETB  WR          ; 1 машинный цикл

```

Время между сбросом и установкой бита WR (P3.6) составит примерно 50 мкс, так как команда DJNZ выполнится 24 раза.

**Пример 13.** Разрешить внешние прерывания по фронту сигналов на входах INT0 и INT1 и прерывание по переполнению таймера/счетчика T/C1. Прерыванию по T/C1 присвоить высший приоритет. Стек организовать начиная с сотой ячейки РПД.

Инициализацию системы прерываний выполним установкой в 1 соответствующих битов регистров IE, IP, TCON. После системного сброса эти регистры обнулены.

MOV	IE,#10001101B ; Установка битов EA, ET1, EX1, EX0	
		; регистра масок прерываний
SETB	PT1	; Высший приоритет T/C1
		установкой
		; бита IP.3 регистра приоритетов
SETB	IT0	; Прерывания по фронту INT0
SETB	IT1	; Прерывания по фронту INT1
MOV	SP,#99	; Модификация указателя стека

Установка бита EA снимает общую блокировку прерываний, которая действует после системного сброса. После выполнения первой команды разрешены три прерывания (в порядке убывания приоритетов): внешнее аппаратное по входу INT0 с вектором 03H, внешнее аппаратное по входу INT1 с вектором 13H и прерывание по таймеру/счетчику T/C1 с вектором 1BH. После выполнения второй команды прерыванию от T/C1 присваивается высокий уровень приоритета. Подпрограмма обслуживания этого прерывания не может быть прервана другим источником прерываний. В то же время установка флага TF1 вызовет переход к вектору 1BH даже при выполнении подпрограмм обслуживания внешних прерываний. Все подпрограммы прерываний должны заканчиваться командой RETI (при обслуживании прерываний действует блокировка прерываний такого же уровня, которую команда RET не снимает). После системного сброса в указатель стека заносится число 7H. Последняя команда модифицирует его так, что при вызове подпрограммы адрес возврата запишется в сотую (младший байт) и сто первую ячейки РПД (старший байт).

**Пример 14.** Получить на линейке светодиодов, подключенных к линиям порта P1, световой эффект бегущего огонька.

Нагрузочной способности порта P1 недостаточно для непосредственного включения светодиодов (выходной ток высокого уровня сигналов – 80 мкА, выходной ток низкого уровня сигналов – 1,6 мА, в то время как нормальное свечение светодиода обеспечивается при токе порядка 5-10 мА). В качестве усилителя тока можно использовать преобразователи уровня или логические элементы НЕ.

Алгоритм получения эффекта «бегущая единица» состоит в организации цикла последовательных операций: вывод 1 в одну из линий порта, организация задержки, смена адреса активной линии порта. Смену адреса проще всего получить путем сдвига кодовой единицы в аккумуляторе. Задержка должна составлять доли секунды. При частоте переключений больше 24 Гц глазу человека будет казаться, что все светодиоды горят непрерывно. С учетом этих замечаний построена приводимая ниже программа.

	MOV	A,#1	; 1
M1:	MOV	P1,A	; 2
DELAY:	DJNZ	R0,\$	; 2
	DJNZ	R1,DELAY	; 2
	RL	A	; 1
	SJMP	M1	; 2

В поле примечания приведено время выполнения команд программы в машинных циклах (при частоте кварцевого резонатора 12 МГц один машинный цикл равен 1 мкс).

После окончания временной задержки регистры R0 и R1 обнулены. Таким образом, каждый раз при реализации временной задержки команда DJNZ R1,DELAY выполняется 256 раз, а команда DJNZ R0,\$ – 256<sup>2</sup> раз.

Суммарное время задержки составляет  $5+2(256+256^2) = 131589$  мкс.

## 4 Примеры контрольных работ по цифровой схемотехнике

### 4.1 Контрольная работа №1

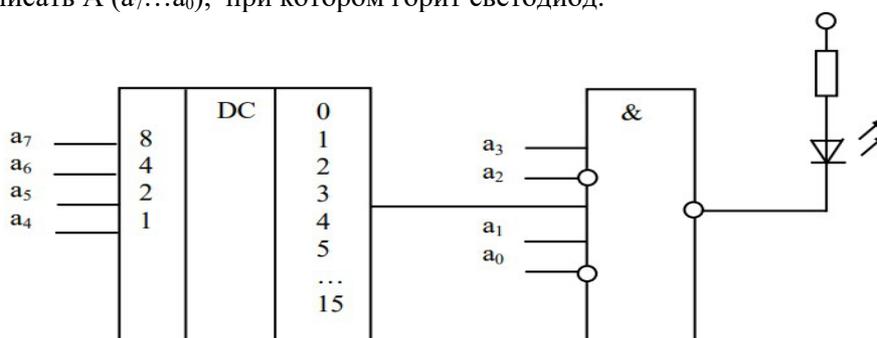
<i>ЦМПП-1 КР1</i>	
1. Запишите двоичный код числа 137Q	
2. Запишите дополнительный код числа плюс 93	
3. Представьте число 78 в двоично-десятичном коде	
4. Укажите прямой код числа, дополнительный код которого 9EH	
5. Определите сумму дополнительных кодов чисел минус 55 и +95	
6. Указать соотношения, в которых допущена ошибка.	
1) $\overline{xy}(x+y) = x \oplus y$ 2) $\overline{x} = x \oplus 0$ 3) $\overline{xy} = \overline{x} + \overline{y}$ 4) $xy(x \oplus \overline{y}) = xy$ 5) $xy(x+y) = \overline{xy}$	

### 4.2 Контрольная работа №2

1. В приведенном списке ИМС указать цифровые интегральные микросхемы

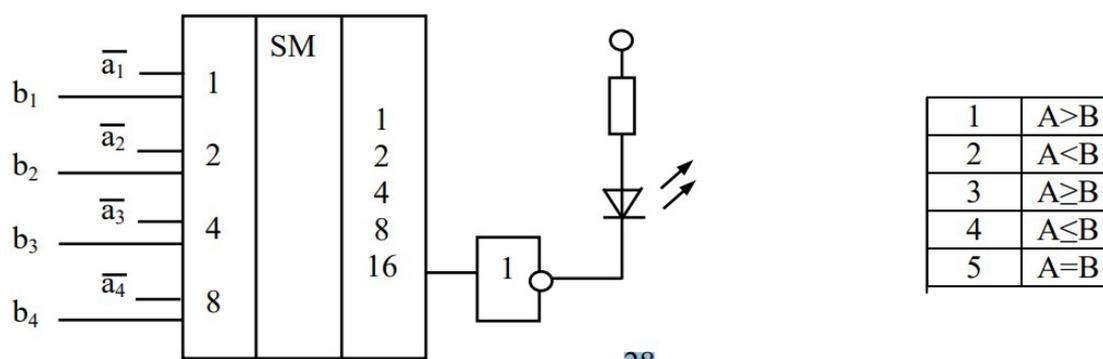
К555ИД1	К556РТ7
К140УД7	К561ТМ2
К564 ИМ5	К252ПА2
К142 ЕН6	К133ЛА4

2. Записать A (a<sub>7</sub>...a<sub>0</sub>), при котором горит светодиод.



4. Определите сумму дополнительных кодов чисел минус 67 и +13

5. Какую функцию сравнения фиксирует горящий светодиод

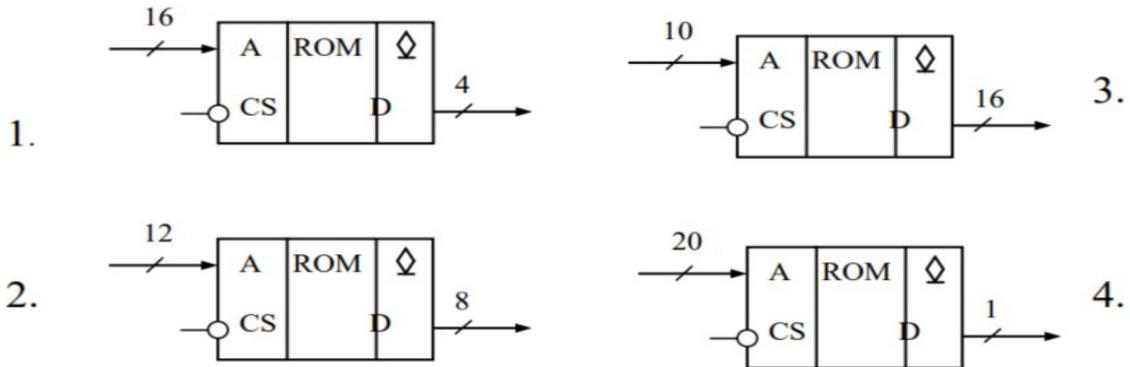


### 4.3 Контрольная работа №3

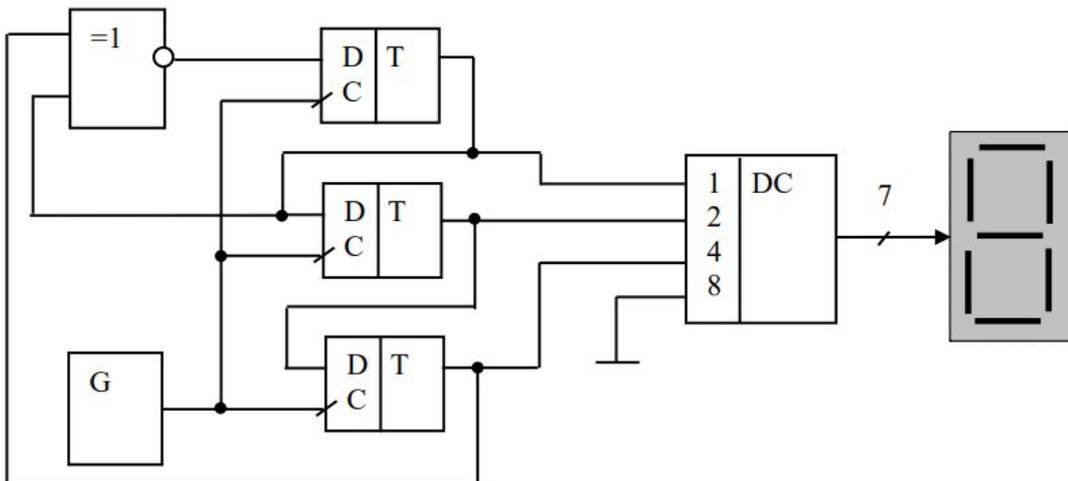
1. В предложенном списке микросхем отметить JK-триггер

K555TP2 K555TM5 K555TL2 K555TM2 K555TB6

2. Указать ПЗУ с наибольшим объемом памяти

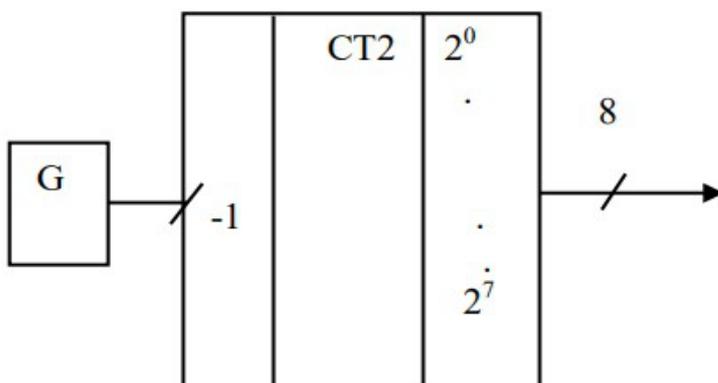


3. Записать числа, которые последовательно загорятся на цифровом индикаторе в цикле при работе счетчика и соответствуют его состояниям, начиная с нулевого



4. Определите сумму дополнительных кодов чисел минус 67 и +13

5. Счетчик находился в 29 состоянии, после чего на его вход поступило 200 импульсов. В каком состоянии будет находиться счетчик?



## 5 Примеры контрольной работы по микропроцессорной технике

### 5.1 Контрольная работа №1

**Задача 1.** Транслировать команду SJMP \$-5.

**Решение.** Команда «короткий безусловный переход» выполняет безусловный переход в программе по указанному адресу. Знак \$ в поле операндов используется для обозначения текущего содержимого программного счетчика PC (он равен адресу первого байта рассматриваемой команды). Команда двухбайтовая, первый байт равен 80H. Второй байт команды — относительное смещение со знаком (rel) от начального адреса следующей команды до адреса, указанного в команде. В данном примере  $rel = -(5+2) = -7 = F9H$  (rel записывается в дополнительном коде и находится в диапазоне от -128 до +127).

**Ответ:** 80F9H

В случае SJMP \$+5, ответ: 8003H.

**Задача 2.** Определить частоту следования импульсов (кГц) на выводе микроконтроллера P1.0 при выполнении программы на частоте кварцевого резонатора 12 МГц:

ORG 0

```
mov TMOD,#2           ; перевести таймер T0 во второй режим работы, а
                       ; T1 – в нулевой.
mov TH0, #156         ;Загрузить старший байт таймера
mov TL0, #156         ;Загрузить младший байт таймера
setb TR0              ;Запуск таймера T0
M1:jnb TF0, M1        ;Подождать пока не переполнится таймер
cpl P1.0              ;Проинвертировать сигнал на P1.0
sjmp M1               ;снова перейти к ожиданию окончания временного
                       ; интервала
```

**Решение.** Первая команда переводит таймер T0 во второй режим работы – режим автоперезагрузки старшего байта таймера TH0 в младший TL0, с тактированием от внутренних синхроимпульсов. До переполнения таймера необходимо  $2^8 - 156 = 256 - 156 = 100$  машинных циклов т.е 100 мкс.

При выполнении команды jnb TF0, M1 флаг таймера TF0 сбрасывается. Учитывая, что сигнал на выводе P1.0 инвертируется каждые 100 мкс, тогда период следования импульсов равен 200 мкс. Отсюда частота следования импульсов:  $F = 1 / (200 * 10^{-6}) = 5$  кГц.

**Ответ:** 5кГц.

**Задача 3.** Определить время выполнения (мкс) подпрограммы, частота кварцевого резонатора 12 МГц:

```
1   DELAY:   MOV     R0, #2     ;1
2           MOV     R1, #10    ;1
3   M1:      DJNZ   R0, $      ;2
4           DJNZ   R1, M1     ;2
5           RET                    ;2
```

**Решение.** Справа указаны номера команд подпрограммы, слева – время выполнения в машинных циклах. При расчете времени выполнения подпрограммы необходимо учесть время вызова подпрограммы командой CALL DELAY (два машинных цикла). При частоте кварцевого резонатора 12 МГц, один машинный цикл составляет 1 мкс. Команды в строках 1 и 2 выполнятся за два машинных цикла. В строке 3 внутренний цикл подпрограммы, будет выполняться 2\*2 раз, после этого содержимое регистра R0 обнулится и управление программой перейдет на строку 4. В строке 4 находится внешний цикл подпрограммы, на каждый декремент R1 внутренний цикл будет выполняться 2\*256 раз. Внешний цикл будет выполняться (10-1) раз. Кроме того команда в строке 4 выполнится 10 раз, что составит 2\*10 машинных циклов. Команда возврата в строке 5 выполнится за два машинных цикла. Тогда время выполнения подпрограммы с учетом вызова составит:

$$t = 10^{-6}(2+1+1+2*2+(2*256)*(10-1)+2*10+2)=4638 \text{ мкс.}$$

**Ответ:** 4638 мкс.

**Задача 4.** Определить содержимое аккумулятора (шестнадцатеричный код) после выполнения программы:

```

                                ORG 0
1                               MOVC  A,@A+PC
2                               SETB  C
3                               ADD   A, #0AH
4                               DA     A
5                               JMP    $

```

**Решение.** Справа пронумерованы строки. В первой строке стоит команда косвенной адресации к памяти программ. При выполнении этой команды в РС содержится адрес следующей команды, а в аккумуляторе 0. Значит после выполнения команды MOVC A,@A+PC, в аккумуляторе будет код команды SETB C, т.е. D3H. Команда в строке 3 складывает содержимое аккумулятора (D3H) с непосредственными данными (0AH). В результате сложения в младшей и старшей тетраде результата будет код больше числа 9. Команда двоично-десятичной коррекции (строка 4) добавит в младшую и старшую тетраду код 6. Тогда в аккумуляторе будет число 43H. Перенос из старшей тетрады установит бит C в PSW.

**Ответ:** 43H.

**Задача 5.** Оценить содержимое DPTR (четыре шестнадцатеричных символа) после выполнения команд:

```

1  MOV    DPTR,#1234
2  XCH   A,DPL
3  RLC   A
4  XCH   A,DPL
5  XCH   A,DPH
6  RLC   A
7  XCH   A,DPH

```

**Решение.** Справа пронумерованы строки. После выполнения первой строки программы в младший байт указателя данных DPL будет загружено число D2H, в старший DPH – число 04H. После обмена содержимым аккумулятора и DPL (вторая строка), содержимое аккумулятора сдвигается влево на один разряд через бит переноса C. При этом C=1, A=101001000B=A4H. Учитывая, что до выполнения команды бит C=0 и он был сдвинут в младший разряд аккумулятора.

В 5-7 строках происходят те же действия, но с участием старшего байта DPH. До 6 строки A=04H=0000 0100B.

С учетом, что бит C=1, DPH = 0000 1001B=09H.

**Ответ:** 09A4H.

## 5.2 Контрольная работа №2

1. n-разрядный регистр, в котором сохраняется результат арифметической или логической операции называется .....

2. Укажите признаки Принстонской архитектуры МПС

1. данная архитектура относится к комплексу аппаратных средств для пользователя
2. раздельное адресное пространство памяти программ и данных
3. общее адресное пространство памяти программ и данных
4. раздельная системная шина (ША, ШД, ШУ)
5. данная архитектура относится к комплексу программных средств для пользователя
6. общая системная шина (ША, ШД, ШУ)

3. Дайте определение аккумулятору. Какова разрядность аккумулятора в МК51.

4. Определить время задержки (мкс) при частоте тактового генератора 12 МГц., если  $R1=5FH$ :

```
DJNZ R1,$
```

5. После системного сброса выполнен фрагмент программы. Укажите содержимое РОН.

```
MOV R6,#25H
```

```
MOV R1,#12H
```

```
MOV R4,#0F3H
```

```
PUSH 6
```

```
PUSH 1
```

```
PUSH 4
```

```
POP 4
```

```
POP 1
```

```
POP 6
```

## 6 Список рекомендуемой литературы

6.1 Шарапов А.В. Микроэлектроника. Цифровая схемотехника: Учебное пособие / А.В. Шарапов. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2007. – 162 с.: ил.,табл. – (Приоритетные национальные проекты. Образование). – ISBN 978-5-86889-400- 8 (наличие в библиотеке ТУСУР - 90 экз.)

6.2 Основы микропроцессорной техники: Учебное пособие / Шарапов А. В. - 2008. 240 с. [Электронный ресурс] - Режим доступа: <https://edu.tusur.ru/publications/834>, дата обращения: 30.05.2018.

6.3 Сайт Цифровая и микропроцессорная техника-1 [Электронный ресурс]. - <https://sdo.tusur.ru/course/view.php?id=88> (доступ авторизованный).

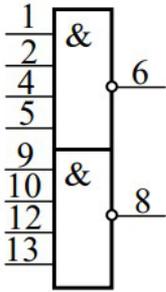
6.4 Сайт Цифровая и микропроцессорная техника-2 [Электронный ресурс]. - <https://sdo.tusur.ru/course/view.php?id=427> (доступ авторизованный).

6.5 Маловичко, Ю. В. Основы микропроцессорной техники : учебное пособие / Ю. В. Маловичко. — Норильск : НГИИ, 2015. — 171 с. — ISBN 978-5-89009-635-7.— Текст: электронный// Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/155906> (дата обращения: 26.03.2018). — Режим доступа: для авториз. пользователей.

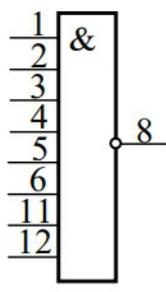
6.6 Водовозов, А. М. Микроконтроллеры для систем автоматики : учебное пособие / А. М. Водовозов. — 2-е изд. — Вологда : ВоГУ, 2015. — 164 с. — ISBN 978-5-87851-599-3.— Текст: электронный// Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/93084> (дата обращения: 27.03.2018). — Режим доступа: для авториз. пользователей.

УСЛОВНЫЕ ГРАФИЧЕСКИЕ ОБОЗНАЧЕНИЯ ИКРОСХЕМ

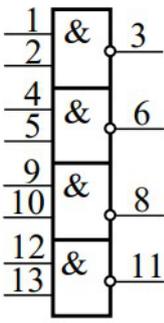
К555ЛА1



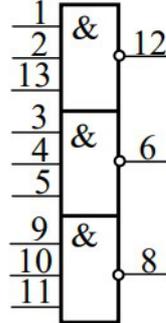
К555ЛА2



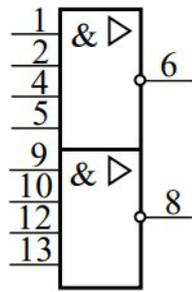
К555ЛА3



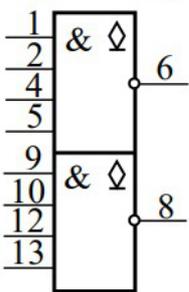
К555ЛА4



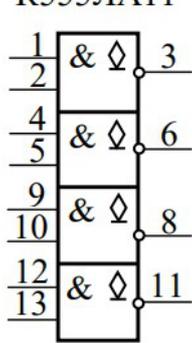
К555ЛА6



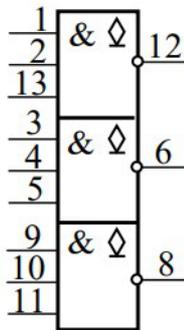
К555ЛА7



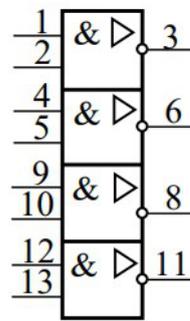
К555ЛА9  
К555ЛА11



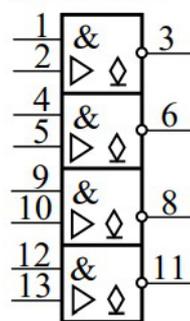
К555ЛА10



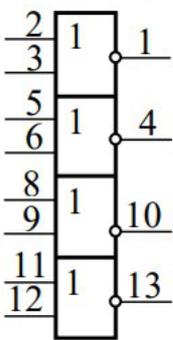
К555ЛА12



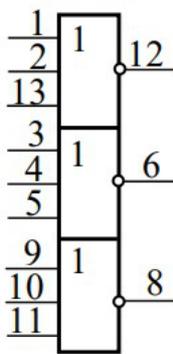
К555ЛА13



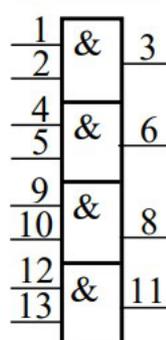
К555ЛЕ1



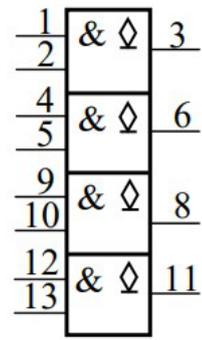
К555ЛЕ4



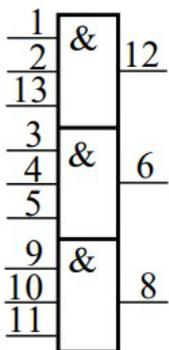
К555ЛИ1



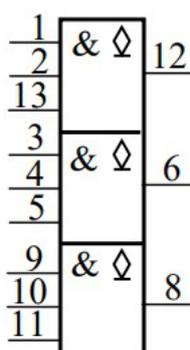
К555ЛИ2



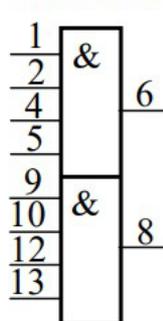
К555ЛИ3



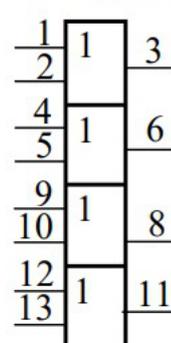
К555ЛИ4



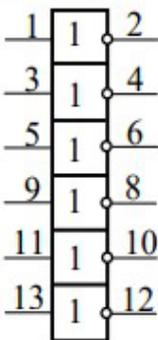
К555ЛИ6



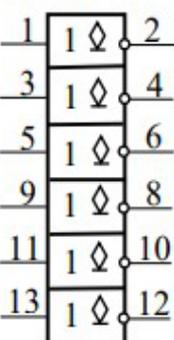
К555ЛИ1



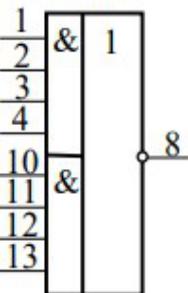
K555JIH1



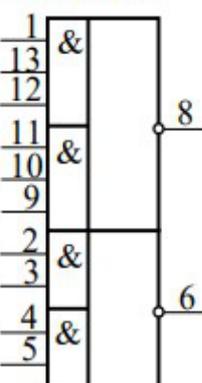
K555JIH2



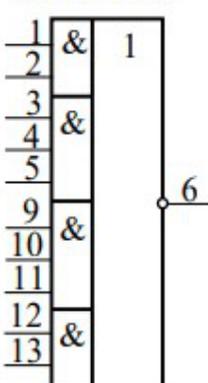
K555JIP4



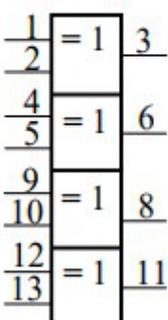
K555JIP11



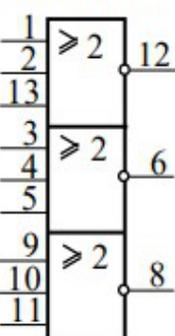
K555JIP13



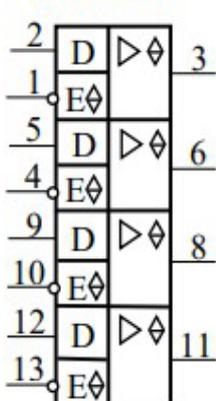
K555JII5



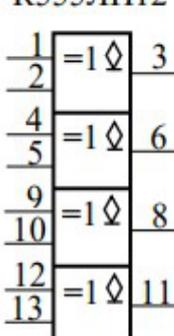
K555JII3



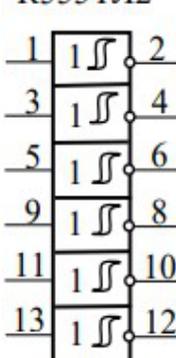
K555JII8



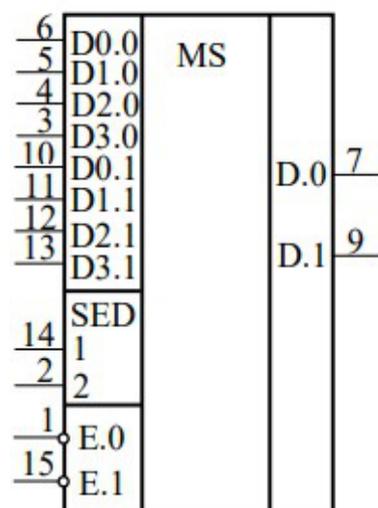
K555JII12



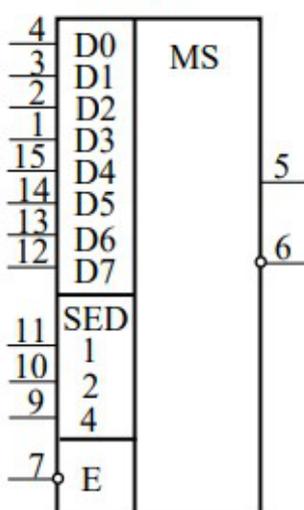
K555TJI2



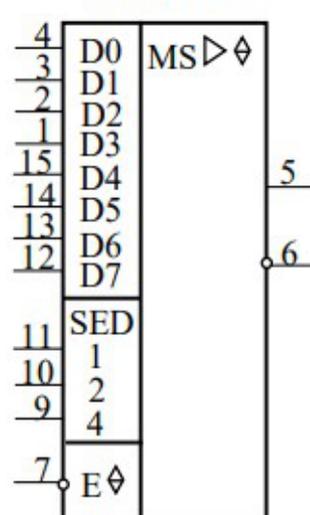
K555KII2

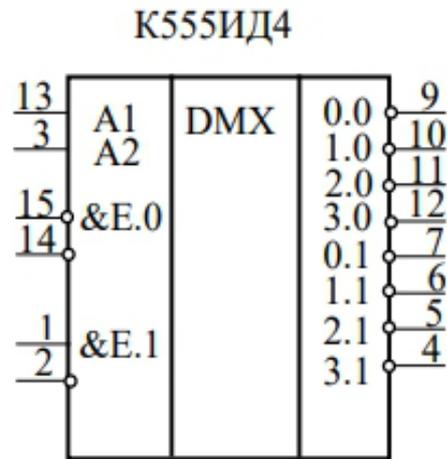
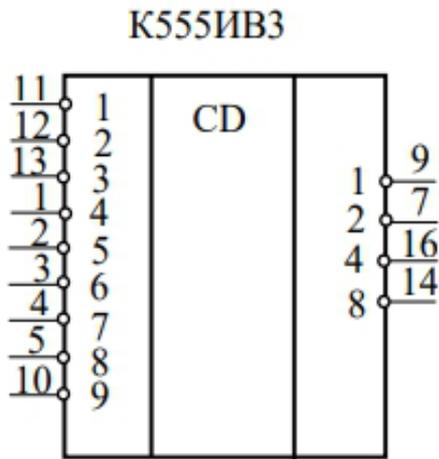
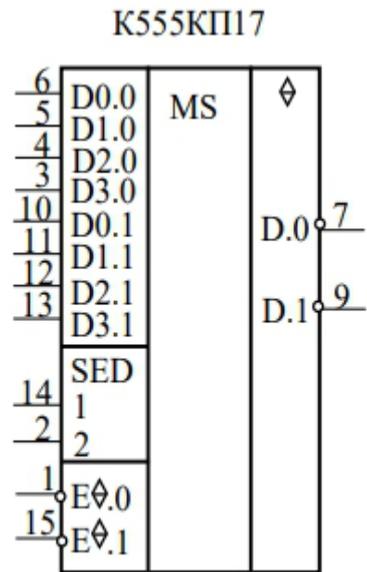
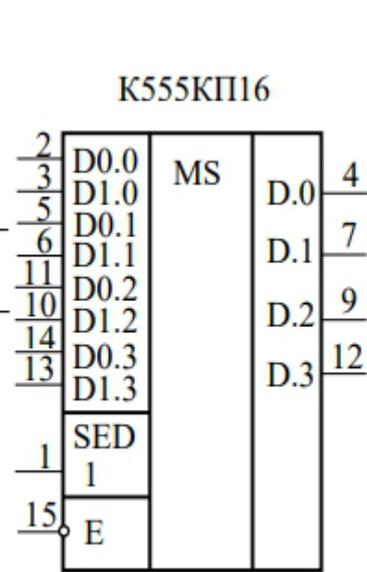
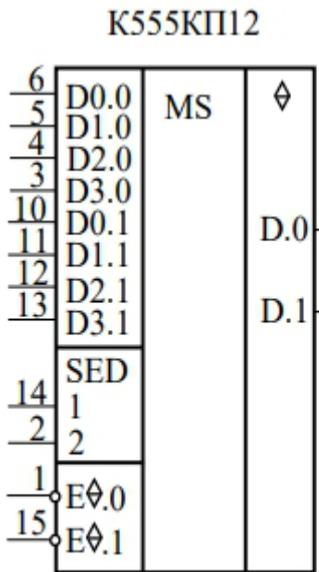
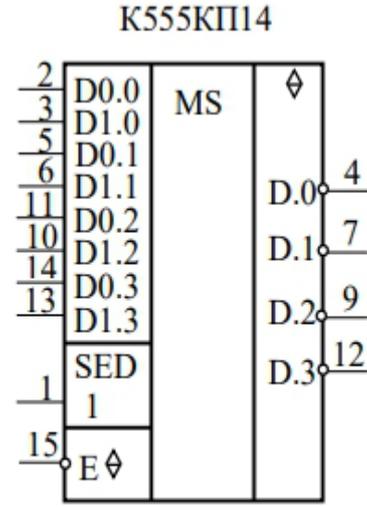
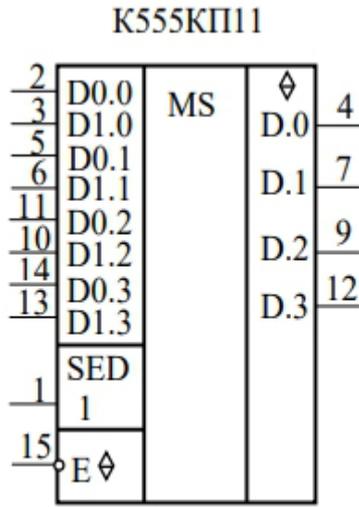
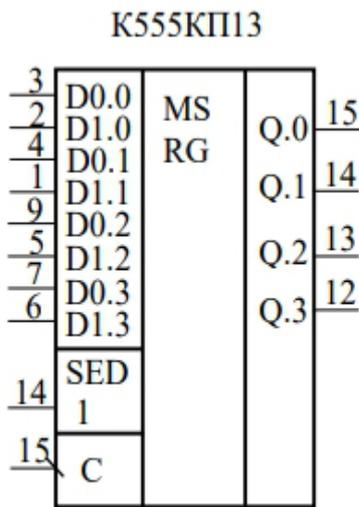


K555KII7

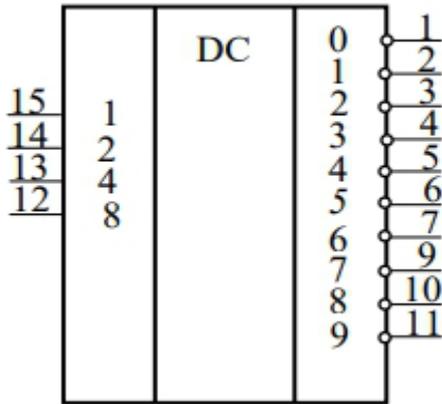


K555KII5

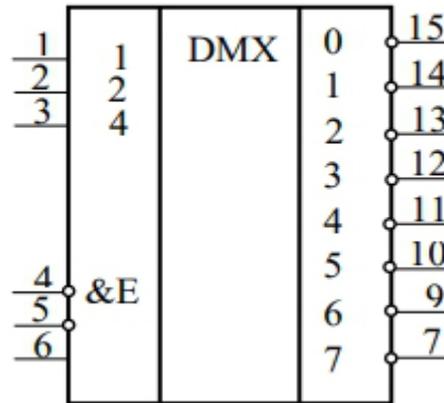




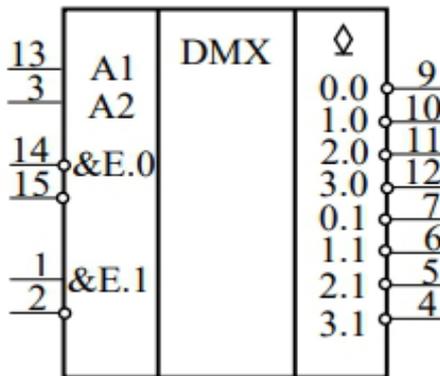
К555ИД6



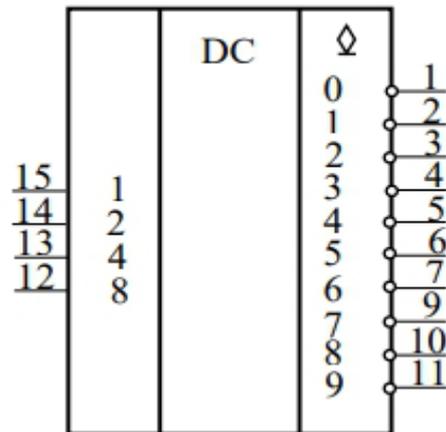
К555ИД7



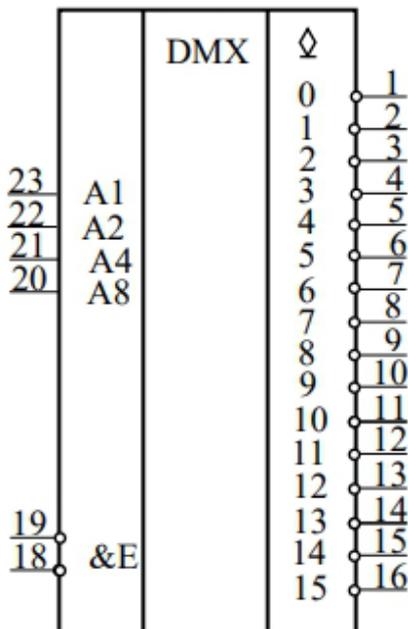
К555ИД5



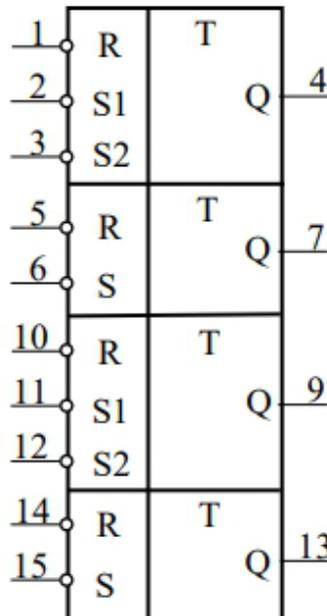
К555ИД10



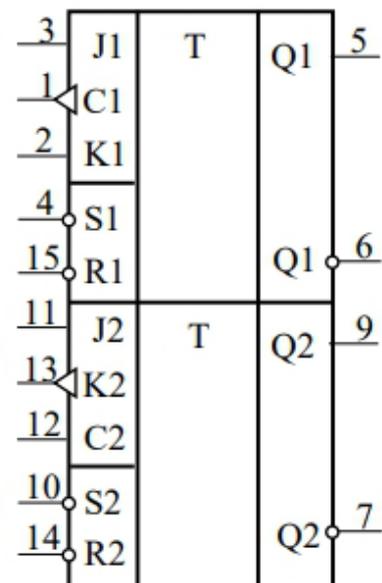
К555ИД19



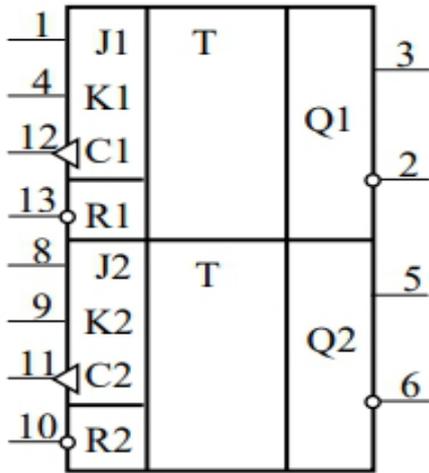
К555ТP2



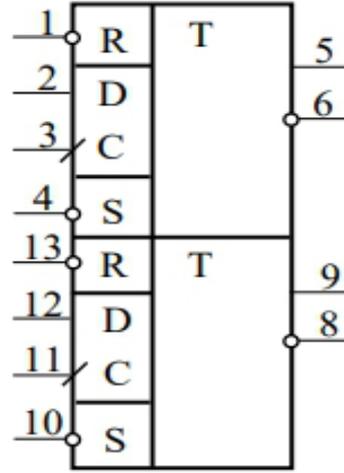
К555ТВ9



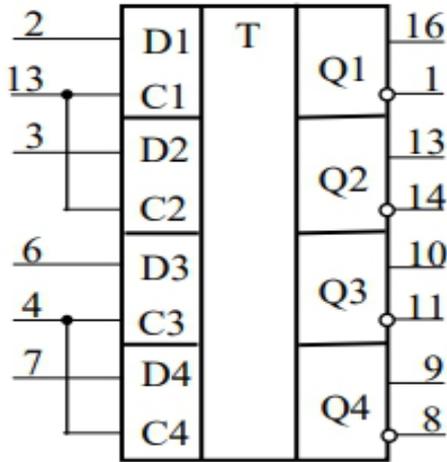
K555TB6



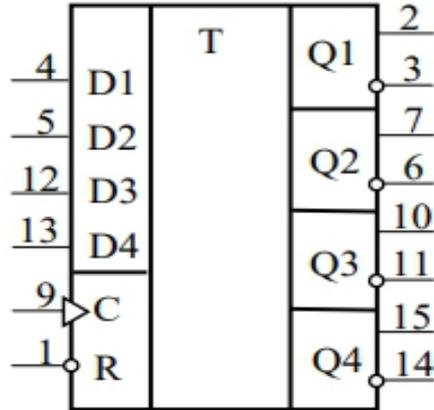
K555TM2



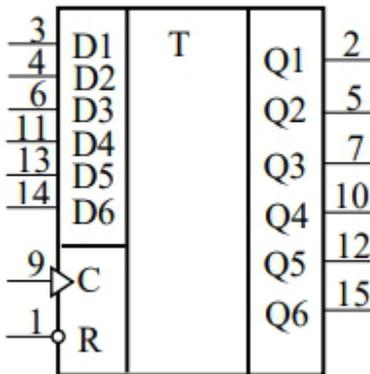
K555TM7



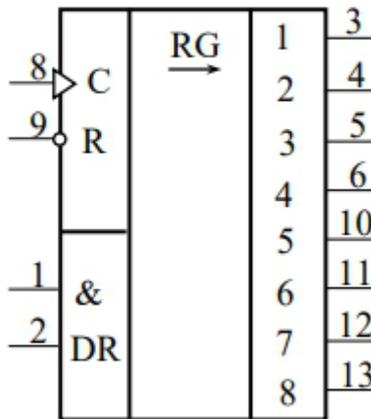
K555TM8



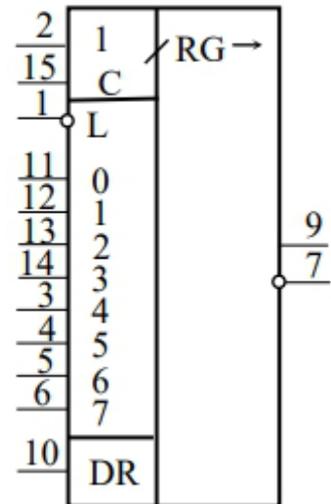
K555TM9

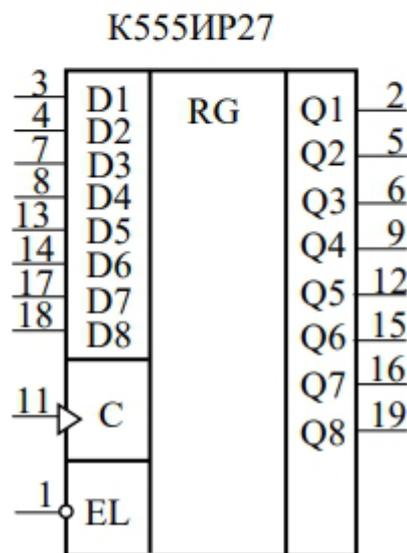
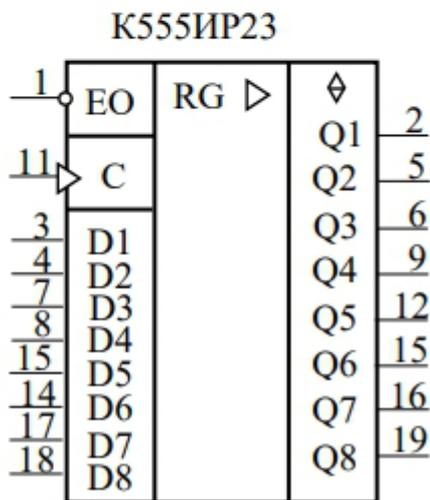
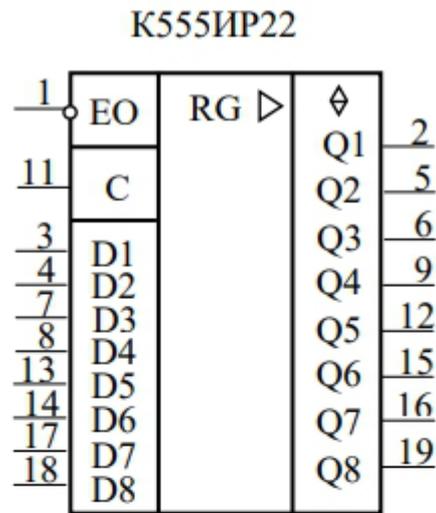
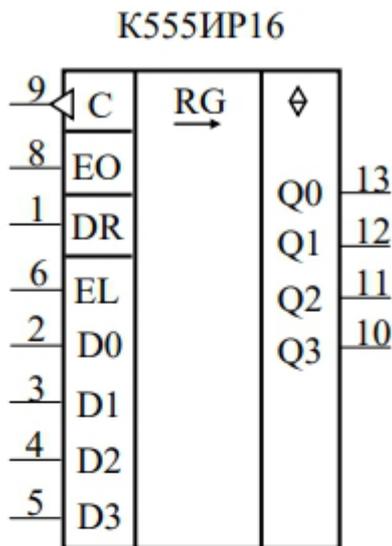
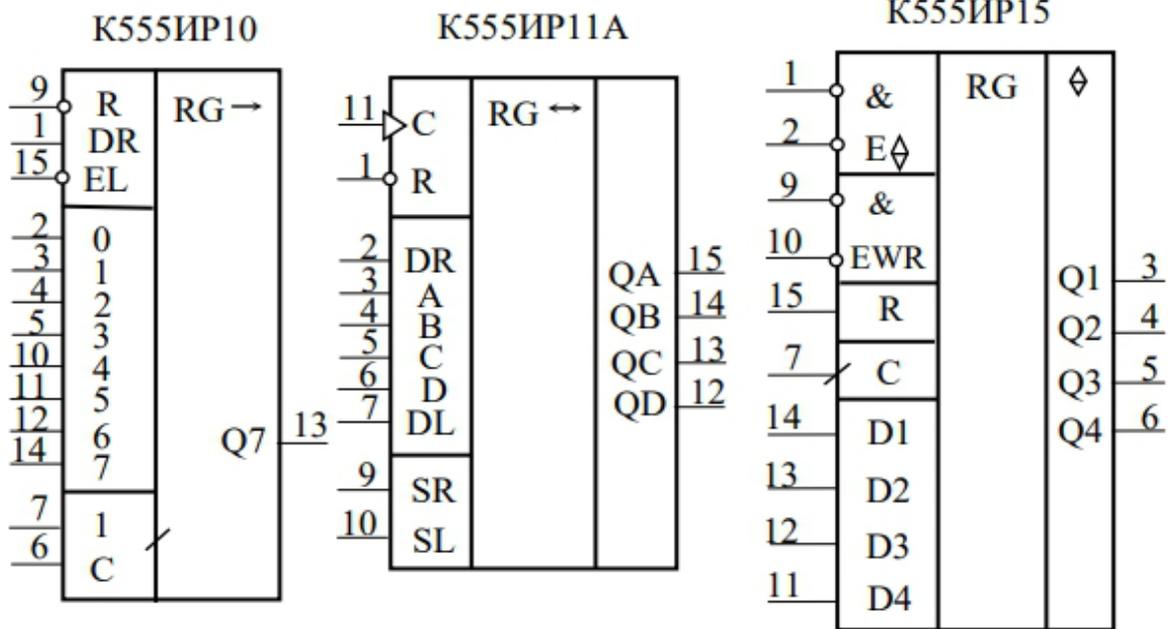


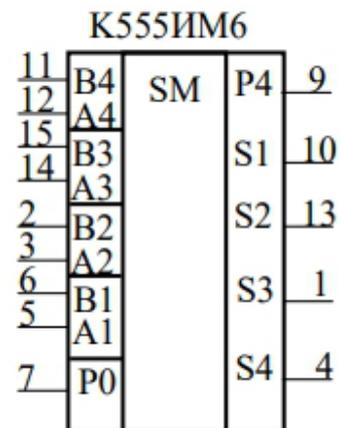
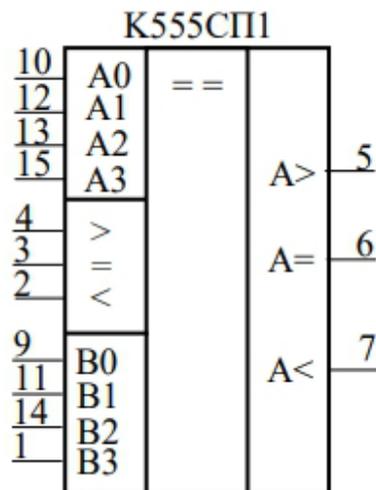
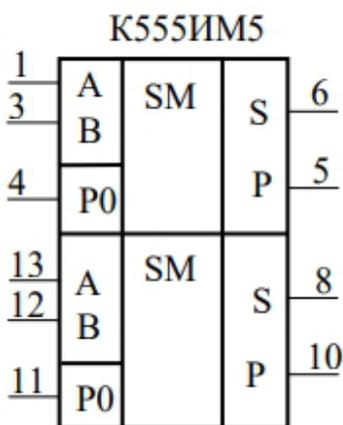
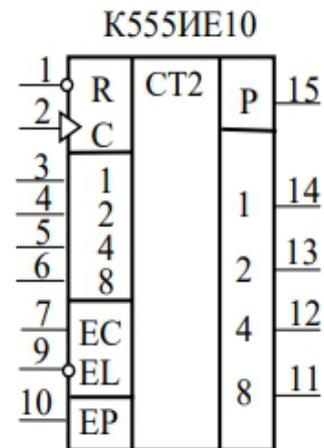
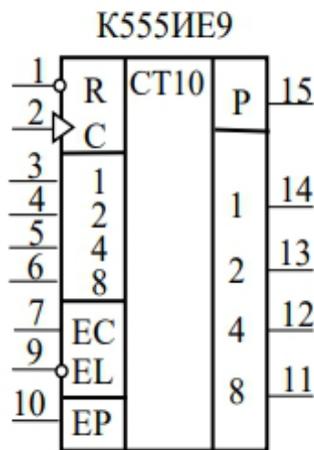
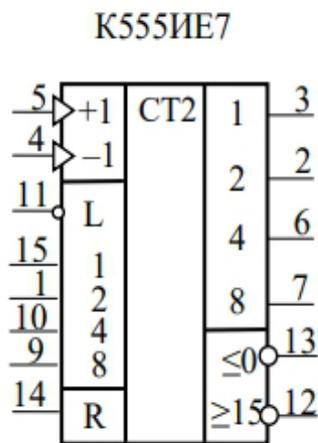
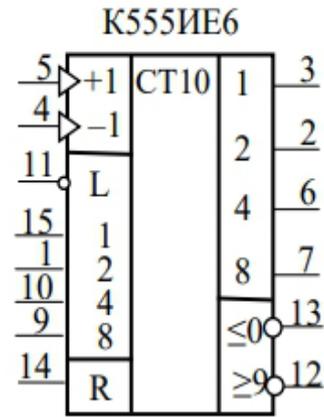
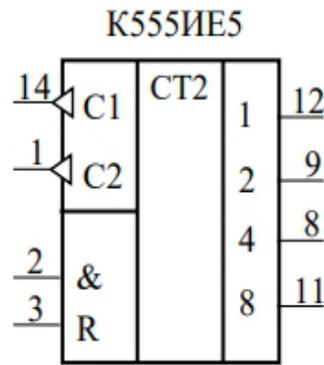
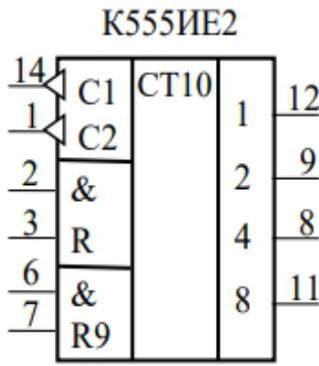
K555IP8



K555IP9







**ПРИЛОЖЕНИЕ Б (справочное)**

**Система команд МК51**

**Группа команд передачи данных**

Таблица 1

Название команды	Мнемокод	КОП	Б	Ц	Операция
Пересылка в аккумулятор из регистра ( $n = 0 - 7$ )	MOV A, Rn	11101rrr	1	1	(A) = (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	11100101	2	1	(A) = (ad)
Пересылка в аккумулятор байта из РДП ( $i = 0, 1$ )	MOV A, @Ri	1110011i	1	1	(A) = ((Ri))
Загрузка в аккумулятор константы	MOV A, #d	01110100	2	1	(A) = #d
Пересылка в регистр из аккумулятора	MOV Rn, A	11111rrr	1	1	(Rn) = (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	10101rrr	2	2	(Rn) = (ad)
Загрузка в регистр константы	MOV Rn, #d	01111rrr	2	1	(Rn) = #d
Пересылка по прямому адресу аккумулятора	MOV ad, A	11110101	2	1	(ad) = (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	10001rrr	2	2	(ad) = (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	10000101	3	2	(add) = (ads)
Пересылка байта из РДП по прямому адресу	MOV ad, @Ri	1000011i	2	2	(ad) = ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	01110101	3	2	(ad) = #d
Пересылка в РДП из аккумулятора	MOV @Ri, A	1111011i	1	1	((Ri)) = (A)
Пересылка в РДП прямоадресуемого байта	MOV @Ri, ad	0110011i	2	2	((Ri)) = (ad)

Окончание таблицы 1

Название команды	Мnemonic	КОП	Б	Ц	Операция
Загрузка указателя данных	MOV DPTR, #d16	10010000	3	2	(DPTR) = #d16
Пересылка в аккумулятор байта из ПП	MOVC A, @A + DPTR	10010011	1	2	(A) = ((A) + (DPTR))
Пересылка в аккумулятор байта из ПП	MOVC A, @A + PC	10000011	1	2	(PC) = (PC) + 1 (A) = ((A) + (PC))
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	1110001i	1	2	(A) = ((Ri))
Пересылка в аккумулятор байта из расширенной ВПД	MOVX A, @DPTR	11100000	1	2	(A) = ((DPTR))
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	1111001i	1	2	((Ri)) = (A)
Пересылка в расширенную ВПД из аккумулятора	MOVX @DPTR, A	11110000	1	2	((DPTR)) = (A)
Загрузка в стек	PUSH ad	11000000	2	2	(SP) = (SP) + 1 ((SP)) = (ad)
Извлечение из стека	POP ad	11010000	2	2	(ad) = (SP) (SP) = (SP) - 1
Обмен аккумулятора с регистром	XCH A, Rn	11001rrr	1	1	(A) <-> (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	11000101	2	1	(A) <-> (ad)
Обмен аккумулятора с байтом из РДП	XCH A, @Ri	1100011i	1	1	(A) <-> ((Ri))
Обмен младшей тетрады аккумулятора с младшей тетрадой байта РДП	XCHD A, @Ri	1101011i	1	1	(A <sub>0-3</sub> ) <-> ((R))

## Группа команд арифметических

Группу образуют 24 команды, выполняющие операции сложения, десятичной коррекции, инкремента/декремента байтов. Дополнительно по сравнению с МК48 введены команды вычитания, умножения и деления байтов.

Таблица 2

Название команды	Мнемокод	КОП	Б	Ц	Операция
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	2	1	$(A) = (A) + (ad)$
Сложение аккумулятора с байтом из РПД ( $i = 0, 1$ )	ADD A, @Ri	0010011i	1	1	$(A) = (A) + ((Ri))$
Сложение аккумулятора с константой	ADD A, #d	00100100	2	1	$(A) = (A) + \#d$
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	00111rrr	1	1	$(A) = (A) + (Rn) + (C)$
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	2	1	$(A) = (A) + (ad) + (C)$
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @Ri	0011011i	1	1	$(A) = (A) + ((Ri)) + (C)$
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	2	2	$(A) = (A) + \#d + (C)$
Десятичная коррекция аккумулятора	DA A	11010100	1	1	Если $(A_{0-3}) > 9 \vee ((AC) = 1)$ , то $(A_{0-3}) = (A_{0-3}) + 6$ , затем если $(A_{4-7}) > 9 \vee ((C) = 1)$ , то $(A_{4-7}) = (A_{4-7}) + 6$
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	10011rrr	1	1	$(A) = (A) - (C) - (Rn)$

Название команды	Мнемокод	КОП	Б	Ц	Операция
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	1001011i	1	1	$(A) = (A) - (C) - ((Ri))$
Вычитание из аккумулятора константы и заема	SUBB A, #d	10010100	2	2	$(A) = (A) - (C) - #d$
Инкремент аккумулятора	INC A	00000100	1	1	$(A) = (A) + 1$
Инкремент регистра	INC Rn	00001rrr	1	1	$(Rn) = (Rn) + 1$
Инкремент прямоадресуемого байта	INC ad	00000101	3	2	$(ad) = (ad) + 1$
Инкремент байта в РПД	INC @Ri	0000011i	1	1	$((Ri)) = ((Ri)) + 1$
Инкремент указателя данных	INC DPTR	10100011	1	1	$(DPTR) = (DPTR) + 1$
Декремент аккумулятора	DEC A	00010100	1	1	$(A) = (A) - 1$
Декремент регистра	DEC Rn	00011rrr	1	1	$(Rn) = (Rn) - 1$
Декремент прямоадресуемого байта	DEC ad	00010101	3	2	$(ad) = (ad) - 1$
Декремент байта в РПД	DEC @Ri	0001011i	1	1	$((Ri)) = ((Ri)) - 1$
Умножение аккумулятора на регистр B	MUL AB	10100100	1	1	$(B)(A) = (A)*(B)$
Деление аккумулятора на регистр B	DIV AB	10000100	1	1	$(A).(B) = (A)/(B)$

Команды ADD и ADDC аналогичны командам сложения МК48, но допускают сложение аккумулятора с большим числом операндов.

Аналогично командам ADDC существуют четыре команды SUBB, что позволяет более просто, чем в МК48, производить вычитание байтов и многобайтных двоичных чисел. В МК51 реализуется расширенный (по сравнению с МК48) список команд инкремента/декремента байтов, введена команда инкремента 16-битного операнда.

### Группа команд логических операций

Данную группу образуют 25 команд, реализующих те же логические операции над байтами, что и в МК48. Однако в МК51 значительно расширено число типов операндов, участвующих в операциях.

Таблица 3

Название команды	Мнемокод	КОП	Б	Ц	Операция
Логическое И аккумулятора и регистра	ANL A, Rn	01011rrr	1	1	$(A) = (A) \wedge (Rn)$
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	01010101	2	1	$(A) = (A) \wedge (ad)$

Продолжение таблицы 3

Название команды	Мнемокод	КОП	Б	Ц	Операция
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	0101011i	1	1	$(A) = (A) \wedge ((Ri))$
Логическое И аккумулятора и константы	ANL A, #d	01010100	2	1	$(A) = (A) \wedge \#d$
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	01010010	2	1	$(ad) = (ad) \wedge (A)$
Логическое И прямоадресуемого байта и константы	ANL ad, #d	01010011	3	2	$(ad) = (ad) \wedge \#d$
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	01001rrr	1	1	$(A) = (A) \vee (Rn)$
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	01000101	2	1	$(A) = (A) \vee (ad)$
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	0100011i	1	1	$(A) = (A) \vee ((Ri))$
Логическое ИЛИ аккумулятора и константы	ORL A, #d	01000100	2	1	$(A) = (A) \vee \#d$
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	01000010	2	1	$(ad) = (ad) \vee (A)$
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #d	01000011	3	2	$(ad) = (ad) \vee \#d$
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	01101rrr	1	1	$(A) = (A) \nabla (Rn)$
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	01100101	2	1	$(A) = (A) \nabla (ad)$
Исключающее ИЛИ аккумулятора и байта из РПД	XRL A, @Ri	0110011i	1	1	$(A) = (A) \nabla ((Ri))$
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	01100100	2	1	$(A) = (A) \nabla \#d$
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	01100010	2	1	$(ad) = (ad) \nabla (A)$
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #d	01100011	3	2	$(ad) = (ad) \nabla \#d$
Сброс аккумулятора	CLR A	11100100	1	1	$(A) = 0$
Инверсия аккумулятора	CPL A	11110100	1	1	$(A) = (\neg A)$

Окончание таблицы 3

Название команды	Мнемокод	КОП	Б	Ц	Операция
Сдвиг аккумулятора влево циклически	RL A	00100011	1	1	$(A_{n+1}) = (A_n)$ , $n = 0 \dots 6, (A_0) = (A_7)$
Сдвиг аккумулятора влево через перенос	RLC A	00110011	1	1	$(A_{n+1}) = (A_n)$ , $n = 0 \dots 6, (A_0) = (C)$ , $(C) = (A_7)$
Сдвиг аккумулятора вправо циклически	RR A	00000011	1	1	$(A_n) = (A_{n+1})$ , $n = 0 \dots 6, (A_7) = (A_0)$
Сдвиг аккумулятора вправо через перенос	RRC A	00010011	1	1	$(A_n) = (A_{n+1})$ , $n = 0 \dots 6, (A_7) = (C)$ , $(C) = (A_0)$
Обмен местами тетрады в аккумуляторе	SWAP A	11000100	1	1	$(A_{0-3}) \leftrightarrow (A_{4-7})$

### Группа команд операции с битами

Отличительной особенностью данной группы команд является то, что они оперируют с однобитными операндами. В качестве таких операндов могут выступать отдельные биты некоторых регистров специальных функций (РСФ) и портов, а также 128 программных флагов пользователя.

Таблица 4

Название команды	Мнемокод	КОП	Б	Ц	Операция
Сброс переноса	CLR C	11000011	1	1	$(C) = 0$
Сброс бита	CLR bit	11000010	2	1	$(b) = 0$
Установка переноса	SETB C	11010011	1	1	$(C) = 1$
Установка бита	SETB bit	11010010	2	1	$(b) = 1$
Инверсия переноса	CPL C	10110011	1	1	$(C) = (\bar{C})$
Инверсия бита	CPL bit	10110010	2	1	$(b) = (\bar{b})$
Логическое И бита и переноса	ANL C, bit	10000010	2	2	$(C) = (C) \wedge (b)$
Логическое И инверсии бита и переноса	ANL C, /bit	10110000	2	2	$(C) = (C) \wedge (\bar{b})$
Логическое ИЛИ бита и переноса	ORL C, bit	01110010	2	2	$(C) = (C) \vee (b)$

Продолжение таблицы 4

Название команды	Мнемокод	КОП	Т	Б	Операция
Логическое ИЛИ инверсии бита и переноса	ORL C, /bit	10100000	4	2	$(C) = (C) \vee (\bar{b})$
Пересылка бита в перенос	MOV C, bit	10100010	4	2	$(C) = (b)$
Пересылка переноса в бит	MOV bit, C	10010010	4	2	$(b) = (C)$

Существуют команды сброса (CLR), установки (SETB) и инверсии (CPL) бит, а также конъюнкции и дизъюнкции бита и флага переноса. Для адресации бит используется прямой восьмиразрядный адрес (bit). Косвенная адресация бит невозможна.

### Группа команд передачи управления

К данной группе команд относятся команды, обеспечивающие условное и безусловное ветвление, вызов подпрограмм и возврат из них, а также команда пустой операции NOP. В большинстве команд используется прямая адресация, т.е. адрес перехода целиком (или его часть) содержится в самой команде передачи управления. Можно выделить три разновидности команд ветвления по разрядности указываемого адреса перехода.

Таблица 5

Название команды	Мнемокод	КОП	Б	Ц	Операция
Длинный переход в полном объеме памяти в программ	LJMP ad16	00000010	3	2	$(PC) = ad16$
Абсолютный переход внутри страницы в 2 Кбайта	AJMP ad11	$a_{10}a_9a_800001$	2	2	$(PC) = (PC) + 2$ $(PC_{0-10}) = ad11$
Короткий относительный переход внутри страницы в 256 байт	SJMP rel	10000000	2	2	$(PC) = (PC) + 2$ $(PC) = (PC) + rel$
Косвенный относительный переход	JMP @A+DPTR	01110011	1	2	$(PC) = (A) + (DPTR)$
Переход, если аккумулятор равен нулю	JZ rel	01100000	2	2	$(PC) = (PC) + 2$ , если $(A) = 0$ , то $(PC) = (PC) + rel$
Переход, если аккумулятор не равен нулю	JNZ rel	01110000	2	2	$(PC) = (PC) + 2$ , если $(A) \neq 0$ ,

Название команды	Мнемокод	КОП	Б	Ц	Операция
Переход, если перенос равен единице	JC rel	01000000	2	2	$(PC) = (PC) + 2$ , если $(C) = 1$ , то $(PC) = (PC) + rel$
Переход, если перенос равен нулю	JNC rel	01010000	2	2	$(PC) = (PC) + 2$ , если $(C) = 0$ , то $(PC) = (PC) + rel$
Переход, если бит равен единице	JB bit, rel	00100000	3	2	$(PC) = (PC) + 3$ , если $(b) = 1$ , то $(PC) = (PC) + rel$
Переход, если бит равен нулю	JNB bit, rel	00110000	3	2	$(PC) = (PC) + 3$ , если $(b) = 0$ , то $(PC) = (PC) + rel$
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	00010000	3	2	$(PC) = (PC) + 3$ , если $(b) = 1$ , то $(b) = 0$ и $(PC) = (PC) + rel$
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	11011rrr	2	2	$(PC) = (PC) + 2$ , $(Rn) = (Rn) - 1$ , если $(Rn) = 0$ , то $(PC) = (PC) + rel$
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	11010101	3	2	$(PC) = (PC) + 2$ , $(ad) = (ad) - 1$ , если $(ad) \neq 0$ , то $(PC) = (PC) + rel$

Название команды	Мнемокод	КОП	Б	Ц	Операция
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	10110101	3	2	$(PC) = (PC) + 3$ , если $(A) ? (ad)$ , то $(PC) = (PC) + rel$ , если $(A) < (ad)$ , то $(C) = 1$ , иначе $(C) = 0$
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	10110100	3	2	$(PC) = (PC) + 3$ , если $(A) ? \#d$ , то $(PC) = (PC) + rel$ , если $(A) < \#d$ , то $(C) = 1$ , иначе $(C) = 0$
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	10111rrr	3	2	$(PC) = (PC) + 3$ , если $(Rn) ? \#d$ , то $(PC) = (PC) + rel$ , если $(Rn) < \#d$ , то $(C) = 1$ , иначе $(C) = 0$
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri, #d, rel	1011011i	3	2	$(PC) = (PC) + 3$ , если $((Ri)) ? \#d$ , то $(PC) = (PC) + rel$ , если $((Ri)) < \#d$ , то $(C) = 1$ , иначе $(C) = 0$
Длинный вызов подпрограммы	LCALL ad16	00010010	3	2	$(PC) = (PC) + 3$ , $(SP) = (SP) + 1$ , $((SP)) = (PC_{0-7})$ , $(SP) = (SP) + 1$ , $((SP)) = (PC_{8-15})$ , $(PC) = ad16$
Возврат из подпрограммы	RET	00100010	1	2	$(PC_{8-15}) = ((SP))$ , $(SP) = (SP) - 1$ , $(PC_{0-7}) = ((SP))$ , $(SP) = (SP) - 1$

Окончание таблицы 5

Название команды	Мнемокод	КОП	Б	Ц	Операция
Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	ACALL ad11	$a_{10}a_9a_810001$	2	2	$(PC) = (PC) + 2,$ $(SP) = (SP) + 1,$ $((SP)) = (PC_{0-7}),$ $(SP) = (SP) + 1,$ $((SP)) = (PC_{8-15}),$ $(PC_{0-10}) = ad11$
Возврат из подпрограммы	RET	00100010	1	2	$(PC_{8-15}) = ((SP)),$ $(SP) = (SP) - 1,$ $(PC_{0-7}) = ((SP)),$ $(SP) = (SP) - 1$
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	2	$(PC_{8-15}) = ((SP)),$ $(SP) = (SP) - 1,$ $(PC_{0-7}) = ((SP)),$ $(SP) = (SP) - 1$
Холостая команда	NOP	00000000	1	1	$(PC) = (PC) + 1$

**Длинный переход.** Переход по всему адресному пространству ПП. В команде содержится полный 16-битный адрес перехода (ad 16). Трех байтные команды длинного перехода содержат в мнемокоде букву L (Long). Всего существует две такие команды: LJMP - длинный переход и LCALL - длинный вызов подпрограммы. На практике редко возникает необходимость перехода в пределах всего адресного пространства и чаще используются укороченные команды перехода, занимающие меньше места в памяти.

**Абсолютный переход.** Переход в пределах одной страницы памяти программ размером 2048 байт. Такие команды содержат только 11 младших бит адреса перехода (ad 11). Команды абсолютного перехода имеют формат 2 байта. Начальная буква мнемокода - A (Absolute). При выполнении команды в вычисленном адресе следующей по порядку команды  $((PC) = (PC) + 2)$  11 младших бит заменяются на ad11 из тела команды абсолютного перехода.

**Относительный переход.** Короткий относительный переход позволяет передать управление в пределах -128 - +127 байт относительно адреса следующей команды (команды, следующей по порядку за командой относительного перехода). Существует одна команда безусловного короткого перехода SJMP (Short). Все команды условного перехода используют данный метод адресации. Относительный адрес перехода (rel) содержится во втором байте команды.

**Косвенный переход.** Команда JMP @A + DPTR позволяет передавать управление по косвенному адресу. Эта команда удобна тем, что предоставляет возможность организации перехода по адресу, вычисляемому самой программой и неизвестному при написании исходного текста программы.

**Условные переходы.** Развитая система условных переходов предоставляет возможность осуществлять ветвление по следующим условиям: аккумулятор содержит нуль (JZ); содержимое аккумулятора не равно нулю (JNZ); перенос равен единице (JC); перенос равен нулю (JNC); адресуемый бит равен единице (JB); адресуемый бит равен нулю (JNB).

Для организации программных циклов удобно пользоваться командой DJNZ, которая работает аналогично соответствующей команде МК48. Однако в качестве счетчика циклов в МК51 может использоваться не только регистр, но и прямоадресуемый байт (например, ячейка РПД).

Команда CJNE эффективно используется в процедурах ожидания какого-либо события. Например, команда **WAIT: CJNE A,R0,WAIT** будет выполняться до тех пор, пока на линиях порта 0 не установится информация, совпадающая с содержимым аккумулятора. Все команды данной группы, за исключением CJNE и JBC, не оказывают воздействия на флаги. Команда CJNE устанавливает флаг C, если первый операнд оказывается меньше второго. Команда JBC сбрасывает флаг C в случае перехода.

**Подпрограммы.** Для обращения к подпрограммам необходимо использовать команды вызова подпрограмм (LCALL, ACALL). Эти команды в отличие от команд перехода (LJMP, AJMP) сохраняют в стеке адрес возврата в основную программу. Для возврата из подпрограммы необходимо выполнить команду RET. Команда RETI отличается от команды RET тем, что разрешает прерывания обслуженного уровня.