	Харьковский национальный университет радиоэлектроники Кафедра ЭВМ ТОРБА Александр Алексеевич Кандидат технических наук, профессор	Украина, 61166, г.Харьков, пр. Ленина 14, ауд. 221
		Раб.тел: (8-057) 702-13-54 Email: august.al@yandex.ua

КОНСПЕКТ ЛЕКЦИЙ **И** **МЕТОДИЧЕСКИЕ УКАЗАНИЯ** к лабораторной работе по курсу «Проектирование цифровых устройств на микроконтроллерах семейства **MCS–51»**

Утверждено на заседании каф. ЭВМ
 Протокол №__ от «__» ____ 2017

Зав. кафедрой ЭВМ,

проф.

Коваленко А.А.

2021

ПРИНЯТЫЕ СОКРАЩЕНИЯ:

8ADR	– младшие 8 бит адреса Программной Памяти (ПП);
11ADR	– 11–ти битовый адрес в текущей странице Программной Памяти (ПП) по 2 Кбайта;
16ADR	– 16–ти битовый адрес Программной Памяти;
ВПД	– Внешняя (расширенная) память данных
РПД	– Резидентная (внутренняя) память данных
bit	– 8–ми битовый адрес прямоадресуемого бита в Резидентной Памяти Данных (РПД) или в блоке Регистров Специальных функций (РСФ);
#DAT	– 8–ми битовый непосредственный операнд;
#D16	– 16–ти битовый непосредственный операнд;
dir	– 8–ми битовый адрес прямоадресуемого байта в РПД или РСФ;
DPTR	– Регистр указатель данных
Ft	– Частота кварцевого генератора;
PC	– Программный счетчик;
P0, P1, P2, P3	– 8–ми битовые двунаправленные Порты ввода/вывода;
PSW	– Слово состояния процессора;
rel	– 8–ми битовое смещение адреса Программной Памяти (со знаком в дополнительном коде);
Ri	– Регистры косвенной адресации (R0, R1);
i	– Двоичный код регистра косвенной адресации;
Rn	– Регистры Общего Назначения (РОНы) – (R0_R7);
r r r	– Двоичный код РОНа;
SFR	– Блок Регистров Специальных Функций (РСФ);
S0_S7	– Страницы Программной Памяти (ПП) по 256 байт;
sss	– (A10,A9,A8 – три бита адреса) – двоичный код номера страницы ПП по 256 байт;
SP	– Указатель стека;
&	– Логическая команда «И»
∨	– Логическая команда «ИЛИ»
(+)	– Логическая команда «ИСКЛЮЧАЮЩЕЕ ИЛИ»
~	– Логическая команда «ИНВЕРСИЯ»
<>	– Неравенство
[]	– В скобках указаны номера битов;
()	– В скобках указан адрес операнда в Резидентной Памяти Данных (РПД) или Внешней Памяти Данных (ВПД);
(())	– В скобках указан адрес операнда в Памяти Программ (ПП) (Резидентной или Внешней);

МИКРОКОНТРОЛЛЕРЫ СЕМЕЙСТВА MCS-51

Встраиваемый или однокристалльный микроконтроллер (embedded microcontroller), который в отечественной литературе часто называют однокристалльная микро-ЭВМ (ОМЭВМ), представляет собой изготовленную на одном кристалле микропроцессорную систему, ориентированную на реализацию алгоритмов цифрового управления различными объектами и процессами.

Микроконтроллер содержит центральный процессор, внутреннюю постоянную и оперативную память, параллельные и последовательные порты ввода-вывода данных, набор периферийных устройств: таймеры, аналого-цифровые преобразователи, широтно-импульсные модуляторы, контроллер прерываний, модули обработки сигналов (событий) в реальном времени.

Архитектура семейства MCS-51 фирмы Intel была в свое время определена настолько удачно, что является сегодня, по существу, одним из стандартов «де-факто» на мировом рынке 8-разрядных микроконтроллеров.

Понятие «архитектура» семейства микроконтроллеров далее трактуется как совокупность внутренних и внешних программно доступных ресурсов, системы команд, системы прерываний, функций ввода/вывода и протоколов обмена по магистрали. Архитектура семейства воплощается производителем в виде набора связанных функционально-топологических модулей. Конкретный микропроцессор семейства представляет собой определенную комбинацию этих модулей, основой которой является операционное ядро («core» у фирмы Intel).

Семейство MCS-51 фирмы Intel насчитывает в настоящее время около полусотни микроконтроллеров (табл. 1), разбитых производителем на несколько групп (product lines).

Новое семейство микроконтроллеров 8xC151Sx (MCS-151) по системе команд, набору программно доступных ресурсов, системе прерываний, набору блоков ввода-вывода и функциям выводов корпуса совместимы с микроконтроллерами 8xC51Fx. Усовершенствования коснулись, в основном, операционного ядра. Введены: конвейер команд, режим страничной адресации памяти и др.

В результате при конвейерной выборке в пределах одной страницы время выполнения команды составляет два периода частоты задающего кварцевого генератора (вместо 12 периодов у предыдущего семейства MCS-51).

Микроконтроллеры семейства MCS-251 являются развитием архитектуры семейств MCS-51 и MCS-151. В основу положена «старая» система команд и устоявшийся набор блоков ввода/вывода: три таймера-счетчика, последовательный порт, блок PCA и сторожевой таймер.

Центральный процессор микроконтроллеров MCS-251 построен с использованием конвейера команд (время выполнения команд – 2 периода ча-

стоты кварцевого генератора) и регистрового файла. Система команд дополнена инструкциями, оперирующими 16–ти и 32–х разрядными операндами.

Таблица 1 – *Отличительные особенности микроконтроллеров семейства MCS-51*

Контроллер	ROM/ EPROM (кбайт)	RAM (байт)	T/C	Макс. Ft (МГц)	Особенности группы
8031AH	–	128	2	12	n–MOS технология, Базовая конфигурация, 4 порта
8051AH	4K ROM	128	2	12	
8751H	4K EPROM	128	2	12	
80C31BH	–	128	2	12, 16	CMOS технология, Режим понижен. энерго–потребл., 3 бита защиты
80C51BH	4K ROM	128	2	12, 16	
87C51BH	4K EPROM	128	2	16, 20	
8032AH	–	256	3	12	n–MOS технология, 4 порта, 3 бита защиты
8052AH	8K ROM	256	3	12	
8752BH	8K EPROM	256	3	12	
80C32	–	256	3	20, 24	CMOS технология, Таймер/счетчик с прямым и обратным счетом, 3 бита защиты
80C52	8K ROM	256	3	20, 24	
87C52	8K EPROM	256	3	20, 24	
80C54	16K ROM	256	3	20, 24	
87C54	16K EPROM	256	3	20, 24	
80C58	32K ROM	256	3	20, 24	
87C58	32K EPROM	256	3	20, 24	
80L52	8K ROM	256	3	16, 20	Контроллеры с пониженным напряжением питания 2,7...3,6 Вольт
87L52	8K EPROM	256	3	16, 20	
80L54	16K ROM	256	3	16, 20	
87L54	16K EPROM	256	3	16, 20	
80L58	32K ROM	256	3	16, 20	
87L58	32K EPROM	256	3	16, 20	
80C31FA	–	256	3	20, 24	Модуль PCA, T/C с прямым и обратным счетом
80C51FA	8K ROM	256	3	20,24	
87C51FA	8K EPROM	256	3	20,24	
83C51FB	16K ROM	256	3	20, 24	Сторожевой таймер, 3 бита защиты
87C51FB	16K EPROM	256	3	20, 24	
83C51FC	32K ROM	256	3	20, 24	
87C51FC	32K EPROM	256	3	20, 24	
80C51GB	–	256	3	20, 24	АЦП (8 кан/8 разрядов), 2 PCA, 6 портов I/O, Сторожевой таймер
83C51GB	8K ROM	256	3	20, 24	
87C51GB	8K EPROM	256	3	20, 24	

1 СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS-51

Основой микроконтроллера (см. рис. 1) является 8-ми битовое Арифметическо-Логическое устройство (АЛУ) с аппаратной реализацией операций умножения, деления и десятичной коррекции.

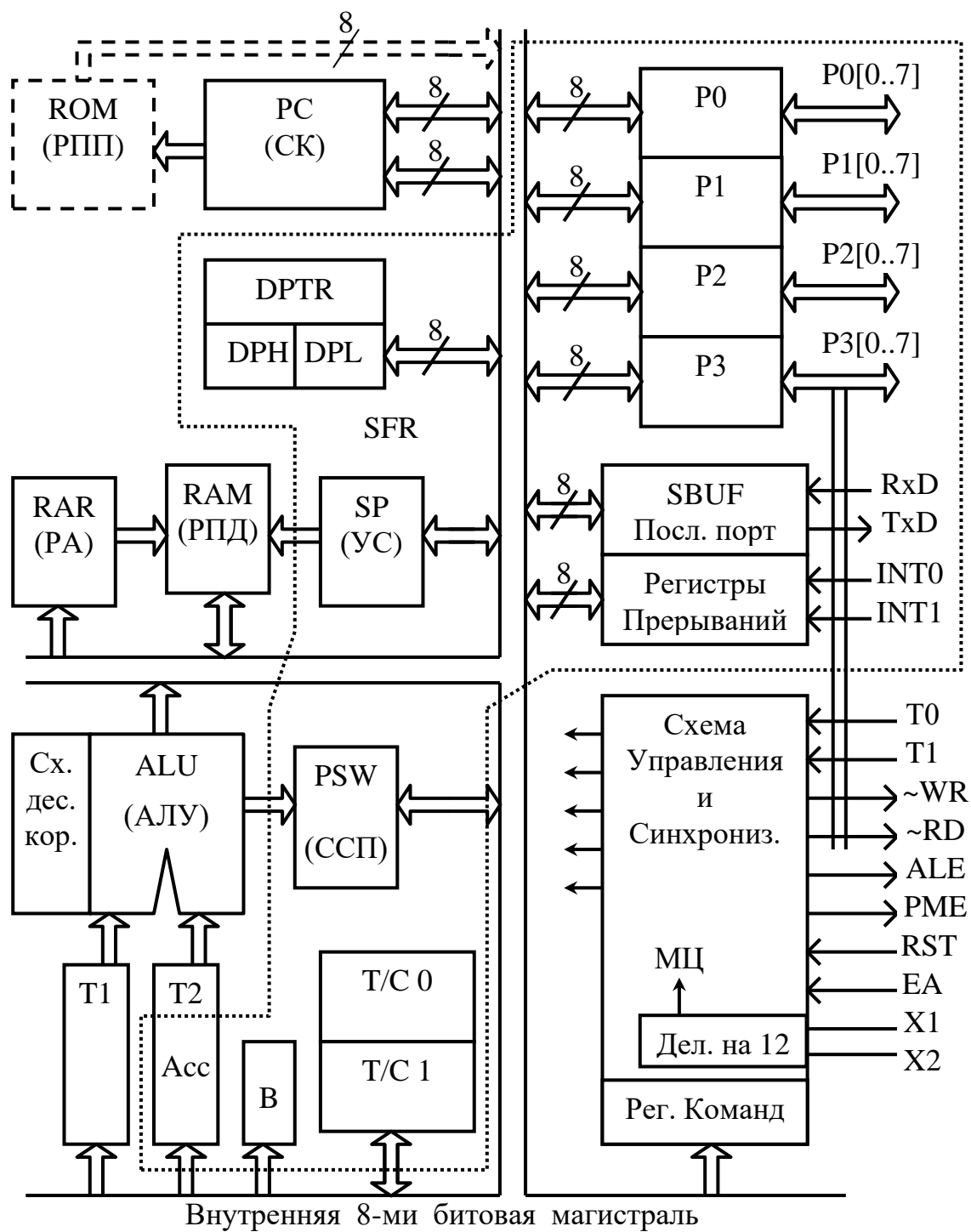


Рис. 1 – Структурная схема микроконтроллеров семейства MCS-51

Память МК имеет Гарвардскую архитектуру, т.е. логически разделена: на память программ – ПП (внутреннюю или внешнюю), адресуемую 16-ти битовым счетчиком команд (СК) и память данных – внутреннюю (Резидентная память данных – РПД) 128 (или 256) байт, а также внешнюю (Внешняя память данных – ВПД) с максимальным размером – до 64 Кбайт.

Физически память программ реализована на ПЗУ (доступна только по чтению), а память данных – на ОЗУ (возможна запись и чтение данных).

Резидентная память данных (РПД) в первых моделях микроконтроллеров семейства MCS-51 имела объем 128 байт. Младшие 32 байта РПД являются одновременно и регистрами общего назначения – РОН (4 банка по 8 РОНов). Программа может обратиться к одному из 8-ми РОНов активного банка – R0...R7. Выбор активного банка РОНов осуществляется программированием двух бит в регистре состояния процессора – PSW.

Переключение банков РОНов упрощает выполнение подпрограмм и обработку прерываний, т.к. не нужно пересылать в стек содержимое РОНов основной программы при вызове подпрограммы (достаточно в подпрограмме перейти в другой активный банк РОНов).

Обращение к РПД возможно с использованием косвенной или прямой байтовой адресации (прямая байтовая адресация позволяет обратиться только к первым 128-ми байтам РПД).

Расширенная область РПД (у микроконтроллеров семейства MCS-52 и последующих семейств) с адреса 128 (80h) до 255 (FFh) может адресоваться только с использованием косвенного метода адресации.

Обращение к регистрам специальных функций – РСФ (SFR – на рис. 1 они обведены пунктирной линией) возможно только с использованием прямой байтовой адресации в диапазоне адресов от 128 (80h) и более (см. табл. 3).

Прием и выдача внешних сигналов осуществляется через 4 универсальных восьмибитовых порта P0..P3. Каждый порт может быть запрограммирован на ввод или вывод байта или одного бита.

При обращении к внешней памяти программ (ВПП) или памяти данных (ВПД) порты P0 и P2 используются как мультиплексированная внешняя шина Адрес/Данные. Линии порта P3 могут выполнять также альтернативные функции:

Таблица 2 – Альтернативные функции порта P3

P3.0	Вход приемника последовательного порта – RxD;
P3.1	Выход передатчика последовательного порта – TxD;
P3.2	Вход внешнего прерывания 0 – \sim INT0;
P3.3	Вход внешнего прерывания 1 – \sim INT1;
P3.4	Внешний вход таймера/счетчика 0 – T0;
P3.5	Внешний вход таймера/счетчика 1 – T1;
P3.6	Выход строб. сигнала при записи в ВПД – \sim WR;
P3.7	Выход строб. сигнала при чтении из ВПД – \sim RD.

Дуплексная (двухсторонняя) передача последовательных данных между микроконтроллерами или ПЭВМ осуществляется через последовательный порт.

Два 16-ти разрядных программируемых таймера-счетчика (Т/С 0, Т/С 1) предназначены для формирования программно управляемых интервалов или генерации прямоугольных импульсов с программно задаваемой частотой.

16-ти битовый регистр DPTR формирует адрес ВПД или базовый адрес Памяти программ в команде преобразования Аккумулятора. Регистр DPTR может также использоваться как два независимых 8-ми битовых регистра (DPL и DPH) для хранения операндов.

8-ми битовый внутренний регистр команд (РК) принимает код выполняемой команды; этот код дешифрируется схемой управления, которая генерирует управляющие сигналы (см. рис. 1).

Таблица 3 – Блок Регистров Специальных Функций (SFR)

Адрес dir	Мнемо- код	Наименование
0E0h	* ACC	Аккумулятор
0F0h	* B	Регистр расширитель аккумулятора
0D0h	* PSW	Слово состояния процессора
0B0h	* P3	Порт 3
0A0h	* P2	Порт 2
90h	* P1	Порт 1
80h	* P0	Порт 0
0B8h	* IP	Регистр приоритетов прерываний
0A8h	* IE	Регистр маски прерываний
99h	SBUF	Буфер последовательного приемо-передатчика
98h	* SCON	Регистр управления/статуса последовательного порта
89h	TMOD	Регистр режимов таймеров/счетчиков
88h	* TCON	Регистр управления/статуса таймеров/счетчиков
8Dh	TH1	Таймер 1 (старший байт)
8Bh	TL1	Таймер 1 (младший байт)
8Ch	TH0	Таймер 0 (старший байт)
8Ah	TL0	Таймер 0 (младший байт)
83h	DPH	Регистр-указатель данных (DPTR) (старший байт)
82h	DPL	Регистр-указатель данных (DPTR) (младший байт)
81h	SP	Регистр-указатель стека
87h	PCON	Регистр управления мощностью потребления

* – Отмеченные регистры допускают адресацию отдельных бит (см. программную модель MCS-51)

2 ПРОГРАММНАЯ МОДЕЛЬ MCS-51

2.1 КАРТА ПРЯМОАДРЕСУЕМЫХ БИТ

В Резидентной Памяти Данных

В блоке Регистров Спец. Функций

7Fh								
Адреса РПД								
30h								
2Fh	7F	7E	7D	7C	7B	7A	79	78
2Eh	77	76	75	74	73	72	71	70
2Dh	6F	6E	6D	6C	6B	6A	69	68
2Ch	67	66	65	64	63	62	61	60
2Bh	5F	5E	5D	5C	5B	5A	59	58
2Ah	57	56	55	54	53	52	51	50
29h	4F	4E	4D	4C	4B	4A	49	48
28h	47	46	45	44	43	42	41	40
27h	3F	3E	3D	3C	3B	3A	39	38
26h	37	36	35	34	33	32	31	30
25h	2F	2E	2D	2C	2B	2A	29	28
24h	27	26	25	24	23	22	21	20
23h	1F	1E	1D	1C	1B	1A	19	18
22h	17	16	15	14	13	12	11	10
21h	0F	0E	0D	0C	0B	0A	09	08
20h	07	06	05	04	03	02	01	00
1Fh	R7							
	Банк РОНов 3							
18h	R0							
17h	R7							
	Банк РОНов 2							
10h	R0							
	R7							
	Банк РОНов 1							
08h	R0							
07h	R7							
	Банк РОНов 0							
00h	R0							

Адреса SFR								
	Регистр В							
0F0h	F7	F6	F5	F4	F3	F2	F1	F0
0E0h	Регистр ACC							
	E7	E6	E5	E4	E3	E2	E1	E0
0B8h	IP							
				BC	BB	В	B9	B8
						A		
0B0h	Порт P3							
	B7	B6	B5	B4	B3	B2	B1	B0
0A8h	IE							
	AF			A	A	A	A9	A8
				C	B	A		
0A0h	Порт P2							
	A7	A6	A5	A4	A3	A2	A1	A0
98h	SCON							
	9F	9E	9D	9C	9B	9A	99	98
90h	Порт P1							
	97	96	95	94	93	92	91	90
88h	TCON							
	8F	8E	8D	8C	8B	8A	89	88
80h	Порт P0							
	87	86	85	84	83	82	81	80

2.2 ТИПЫ КОМАНД MCS-51

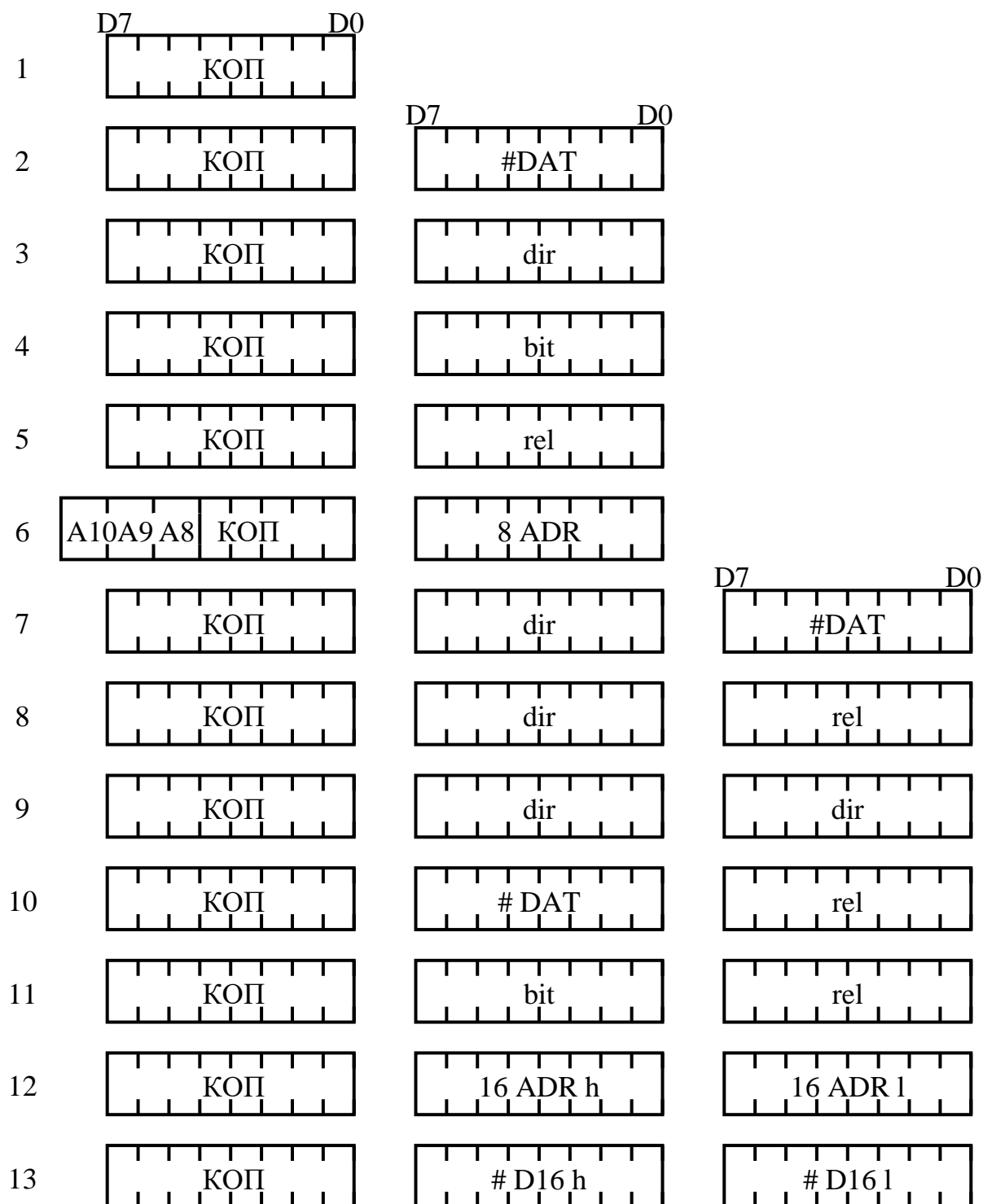


Рис. 2 – Типы команд MCS-51

Почти половина команд выполняется за 1 машинный цикл (МЦ) (см. таблицы 8..12). При частоте кварцевого генератора 12 МГц время выполнения такой команды – 1 мкс. Остальные команды выполняются за 2 машинных цикла, т.е. за 2 мкс (см. таблицы 8..12). Только команды умножения (MUL) и деления (DIV) выполняются за 4 машинных цикла.

За время одного машинного цикла происходит два обращения к Памяти Программ (внутренней или внешней) для считывания двух байтов команды или одно обращение к Внешней Памяти Данных (ВПД).

2.3 МЕТОДЫ (СПОСОБЫ) АДРЕСАЦИИ MCS-51

1. РЕГИСТРОВАЯ АДРЕСАЦИЯ – 8–ми битовый операнд находится в РОНе выбранного (активного) банка регистров;
2. НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ (обозначается знаком – #) – операнд находится во втором (а для 16–ти битового операнда и в третьем) байте команды;
3. КОСВЕННАЯ АДРЕСАЦИЯ (обозначается знаком – @) – операнд находится в Памяти Данных (РПД или ВПД), а адрес ячейки памяти содержится в одном из РОНов косвенной адресации (R0 или R1); в командах PUSH и POP адрес содержится в указателе стека SP; регистр DPTR может содержать адрес ВПД объемом до 64К;
4. ПРЯМАЯ БАЙТОВАЯ АДРЕСАЦИЯ – (dir) – используется для обращения к ячейкам РПД (адреса 00h...7Fh) и к регистрам специальных функций SFR (адреса 80h...0FFh);
5. ПРЯМАЯ БИТОВАЯ АДРЕСАЦИЯ – (bit) – используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках РПД по адресам 20H...2FH и к отдельно адресуемым битам регистров специальных функций (см. табл. 3 и программную модель);
6. КОСВЕННАЯ ИНДЕКСНАЯ АДРЕСАЦИЯ (обозначается знаком – @) – упрощает просмотр таблиц в Памяти Программ, адрес ПП определяется по сумме базового регистра (РС или DPTR) и индексного регистра (Аккумулятора);
7. НЕЯВНАЯ (ВСТРОЕННАЯ) АДРЕСАЦИЯ – в коде команды содержится неявное (по умолчанию) указание на один из операндов (чаще всего на Аккумулятор).

2.4 ФОРМАТ СЛОВА СОСТОЯНИЯ ПРОЦЕССОРА (PSW)

Прямой байтовый адрес PSW : dir – 0D0H.
 Допускается адресация отдельных бит PSW : bit – 0D0H_0D7H.

	7	6	5	4	3	2	1	0
Адрес : bit	0D7h	0D6h	0D5h	0D4h	0D3h	0D2h	0D1h	0D0h
PSW	C	AC	F0	RS1	RS0	OV		P

C – флаг переноса (CARY) или заема, выполняет также функции «булевого Аккумулятора» в командах, оперирующих с битами;

АС – флаг вспомогательного (дополнительного) переноса – устанавливается в «1», если в команде сложения (ADD, ADDC) был перенос из младшей тетрады в старшую (т.е. из 3-го бита в 4-й бит);

F0 – флаг пользователя – устанавливается, сбрасывается и проверяется программно;

RS1,RS0 – Выбор банка регистров:

RS1	RS0	Банк	Адрес (dir)
0	0	0	00h..07h
0	1	1	08h..0Fh
1	0	2	10h..17h
1	1	3	18h..1Fh

OV – Флаг арифметического переполнения; его значение определяется операцией "Исключающее ИЛИ" сигналов входного и выходного переносов старшего разряда АЛУ; единичное значение этого флага указывает на то, что результат арифметической операции в дополнительном коде вышел за допустимые пределы: $-128...+127$; при выполнении операции деления флаг OV сбрасывается, а в случае деления на ноль – устанавливается; при умножении флаг OV устанавливается, если результат больше 255 (0FFh);

Разряд PSW[1] – Резервный, содержит триггер, доступный по записи или чтению;

P – флаг паритета – является дополнением количества единичных битов в аккумуляторе до четного; формируется комбинационной схемой (программно доступен только по чтению).

В микроконтроллерах MCS-51 отсутствует флаг «Z». Но в командах условного перехода (JZ, JNZ) проверяется комбинационной схемой текущее (нулевое или ненулевое) содержимое Аккумулятора.

2.5 СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРОВ 8051

Все команды пересылок и обмена операндов могут осуществляться через Аккумулятор (см. рис. 3). Причем пересылки из/в Внешней Памяти (Памяти Программ или Памяти Данных) могут осуществляться **только** через Аккумулятор.

Большинство пересылок могут осуществляться также через прямоадресуемый байт (**dir**). Существуют даже пересылки **dir – dir** (см. рис. 3).

Отсутствующие пересылки из РОНа в РОН могут быть реализованы как пересылки из РОНа в прямоадресуемый байт **dir** (с учетом того, что РО-НЫ расположены в начальной области Резидентной Памяти Данных, ячейки которой могут адресоваться как **dir**).

Команды обмена **XCH** позволяют пересылать байты без разрушения обоих операндов.

Команда **DA A** – десятичной коррекции выполняет действия над содержимым Аккумулятора **после сложения** BCD-чисел в процессоре (числа складывались по законам шестнадцатеричной арифметики) следующим образом (см. пример):

- если содержимое младшей тетрады Аккумулятора больше 9 или установлен флаг вспомогательного переноса ($AC = 1$), то к содержимому Аккумулятора добавляется 6 (т.е. недостающие шесть цифр в hex-формате);
- если после этого содержимое старшей тетрады Аккумулятора больше 9 или установлен флаг C , то число 6 добавляется к старшей тетраде Аккумулятора.

	$C=0$	$AC=1$		
+ 58	+	0101 1000		; Сложение по законам шестнадцатеричной арифметики
+ 79	+	0111 1001		
<u>137</u>		<u>1101 0001</u>		
	+	0110		; Добавление 6, потому что $AC=1$
		<u>1101 0111</u>		
	+	0110		; Добавление 60h, потому что
		<u>1101 1001</u>		; старшая тетрада больше 9
	$C=1$	1 <u>3</u> <u>7</u>		; Результат

Команду десятичной коррекции **DA A** не применяют после команды инкремента (**INC**), потому что команда инкремента не влияет (не изменяет) на флаги C и AC .

Логические команды:

- логическое «И» – **ANL**,
- логическое «ИЛИ» – **ORL**,
- логическая команда «ИСКЛЮЧАЮЩЕЕ ИЛИ» – **XRL** –

выполняются в Аккумуляторе (как и арифметические), но имеется возможность выполнить логические команды также и в прямоадресуемом байте (dir). При этом второй операнд может быть:

- в Аккумуляторе или
- непосредственный операнд в команде.

Команды вращения (**RR A**, **RL A**) и команды вращения через флаг $CARY$ (**RRC A**, **RLC A**) циклически сдвигают (вращают) содержимое Аккумулятора на 1 бит.

Пересылки битовых операндов осуществляются только через флаг C .

Таблица 5 — Таблица ассемблера MCS-51
КОМАНДЫ ПЕРЕСЫДКИ, ОБМЕНА И ЗАГРУЗКИ

Rn	0	1	2	3	4	5	6	7		@Ri	R0	R1		Прямая адресация			Непосредственная		
MOV A,Rn	E8	E9	EA	EB	EC	ED	EE	EF		MOV A,@Ri	E6	E7		MOV A	E5	, dir	MOV A	74	,#DAT
MOV Rn,A	F8	F9	FA	FB	FC	FD	FE	FF		MOV @Ri,A	F6	F7		MOV dir	F5	, A	MOV dir	75	,#DAT
MOV Rn	78	79	7A	7B	7C	7D	7E	7F	,#DAT	MOV @Ri	76	77	,#DAT	MOV dir	B5	, dir			
MOV Rn	A8	A9	AA	AB	AC	AD	AE	AF	, dir	MOV @Ri	A6	A7	, dir	PUSH	C0	, dir	MOV DPTR	90	,#D16
MOV dir	B8	B9	BA	BB	BC	BD	BE	BF	, Rn	MOV dir	B6	B7	, @Ri	POP	D0	, dir			
XCH A,Rn	C8	C9	CA	CB	CC	CD	CE	CF		XCH A,@Ri	C6	C7		XCH A	C5	, dir			
SWAP A	C4	←Обмен тетрад Аккумулятора								XCHD A,@Ri	D6	D7	←Обмен младших тетрад						
Пересылки		MOVX A,@DPTR						E0	MOVX A,@Ri			E2	E3	Пересылки из		MOVC A,@A+DPTR			93
ВПД с Акк		MOVX @DPTR,A						F0	MOVX @Ri,A			F2	F3	Пам. Прогр. в Акк		MOVC A,@A+PC			B3

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ КОМАНДЫ

Комментарии	Rn	0	1	2	3	4	5	6	7	@Ri	R0	R1	Прям. адресация			Непосредственная		
A←A+()	ADD A,Rn	28	29	2A	2B	2C	2D	2E	2F	ADD A,@Ri	26	27	ADD A	25	,dir	ADD A,	24	,#DAT
A←A+()+C	ADDC A,Rn	38	39	3A	3B	3C	3D	3E	3F	ADDC A,@Ri	36	37	ADDC A	35	,dir	ADDC A	34	,#DAT
A←A-()-C	SUBB A,Rn	98	99	9A	9B	9C	9D	9E	9F	SUBB A,@Ri	96	97	SUBB A	95	,dir	SUBB A	94	,#DAT
()←()+1	INC Rn	08	09	0A	0B	0C	0D	0E	0F	INC @Ri	06	07	INC	05	,dir	INC A	04	
()←()-1	DEC Rn	18	19	1A	1B	1C	1D	1E	1F	DEC @Ri	16	17	DEC	15	,dir	DEC A	14	
A←A & ()	ANL A,Rn	58	59	5A	5B	5C	5D	5E	5F	ANL A,@Ri	56	57	ANL A	55	,dir	ANL A	54	,#DAT
()←() & A													ANL dir	52	,A	ANL dir	53	,#DAT
A←A ∨ ()	ORL A,Rn	48	49	4A	4B	4C	4D	4E	4F	ORL A,@Ri	46	47	ORL A	45	,dir	ORL A	44	,#DAT
()←() ∨ A													ORL dir	42	,A	ORL dir	43	,#DAT
A←A (+) ()	XRL A,Rn	68	69	6A	6B	6C	6D	6E	6F	XRL A,@Ri	66	67	XRL A	65	,dir	XRL A	64	,#DAT
()←() (+) A													XRL dir	62	,A	XRL dir	63	,#DAT
BA ← A × B	MUL AB	A4	INC DPTR				52	Сброс		CLR A	E4	Циклический сдвиг			RL A	23	RR A	03
A . B ← A / B	DIV AB	B4	DA A				D4	Инверсия		CPL A	F4	Сдвиг через перенос			RLC A	33	RRC A	13

Продолжение Таблицы 5

ОПЕРАЦИИ С БИТАМИ

Сброс	CLR C	C3		CLR	C2	bit	ANL C	82	,bit	ANL C	B0	,/bit
Установка	SETB C	D3		SETB	D2	bit	ORL C	72	,bit	ORL C	A0	,/bit
Инверсия	CPL C	B3		CPL	B2	bit	MOV C	A2	,bit	MOV bit,C	92	

КОМАНДЫ УСЛОВНЫХ ПЕРЕХОДОВ														БЕЗУСЛОВНЫЕ ПЕРЕХОДЫ								
														LJMP		02	16ADR					
	Z	NZ	C	NC				B	NB	BC			\Ri	R0	R1			AJMP	NS+01	11ADR		
J	60	70	40	50	rel		J	20	30	10	bit,rel		CJNE @Ri	B6	B7	, #DAT,rel		SJMP		80	rel	
\Rn			0	1	2	3	4	5	6	7			CJNE A	B4	, #DAT,rel		JMP @A+DPTR		73			
CJNE Rn			B8	B9	BA	BB	BC	BD	BE	BF	, #DAT,rel		CJNE A	B5	, dir,rel		LCALL	12	16ADR	RET	22	
DJNZ Rn			D8	D9	DA	DB	DC	DD	DE	DF	, rel		DJNZ		D5	, dir,rel		ACALL	NS+11	11ADR	RETI	32

Таблица 6 – КОМАНДЫ, МОДИФИЦИРУЮЩИЕ ФЛАГИ

Команда	Флаги			Команда	Флаг C	Команда	Флаг C
	C	AC	OV				
ADD, ADDC	+	+	+	CLR C	0	ANL C,bit	+
SUBB	+	+	+	SETB C	1	ANL C,/bit	+
DA A	+	–	–	CPL C	+	ORL C,bit	+
MUL, DIV	0	–	–	RRC	+	ORL C,/bit	+
CJNE	+	–	–	RLC	+	MOV C,bit	+

Флаг OV при выполнении операции деления – сбрасывается, а в случае деления на ноль – устанавливается; при умножении флаг OV устанавливается, если результат больше 255 (0FFH)

Флаги : P (четности Аккумулятора) и Z (нулевого содержимого Аккумулятора) формируются комбинационными схемами. Эти флаги модифицируются любыми командами, изменяющими содержимое Аккумулятора

Таблица 7 - ТАБЛИЦА ДИЗАССЕМБЛЕРА MCS-51

Младшие разряды								
	0	1	2	3	4	5	6	7
С т а р ш и е р а з р я д ы	0	NOP	AJMP S0+8ADR	LJMP 16ADR	RR A	INC A	INC dir	INC @R0
	1	JBC bit,rel	ACALL S0+8ADR	LCALL 16ADR	RRC A	DEC A	DEC dir	DEC @R0
	2	JB bit,rel	AJMP S1+8ADR	RET	RL A	ADD A,#DAT	ADD A,dir	ADD A,@R0
	3	JNB bit,rel	ACALL S1+8ADR	RETI	RLC A	ADDC A,#DAT	ADDC A,dir	ADDC A,@R0
	4	JC rel	AJMP S2+8ADR	ORL dir,A	ORL dir, #DAT	ORL A,#DAT	ORL A,dir	ORL A,@R0
	5	JNC rel	ACALL S2+8ADR	ANL dir,A	ANL dir, #DAT	ANL A,#DAT	ANL A,dir	ANL A,@R0
	6	JZ rel	AJMP S3+8ADR	XRL dir,A	XRL dir, #DAT	XRL A,#DAT	XRL A,dir	XRL A,@R0
	7	JNZ rel	ACALL S3+8ADR	ORL C,bit	JMP @A +DPTR	MOV A,#DAT	MOV dir, #DAT	MOV @R0, #DAT
	8	SJMP rel	AJMP S4+8ADR	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV dir,dir	MOV dir,@R0
	9	MOV DPTR, #D16	ACALL S4+8ADR	MOV bit,C	MOVC A,@A+ DPTR	SUBB A,#DAT	SUBB A,dir	SUBB A,@R0
	A	ORL C,/bit	AJMP S5+8ADR	MOV C,bit	INC DPTR	MUL AB		MOV @R0,dir
	B	ANL C,/bit	ACALL S5+8ADR	CPL bit	CPL C	CJNE A,#DAT ,rel	CJNE A,dir,rel	CJNE @R0,#D ,rel
	C	PUSH Dir	AJMP S6+8ADR	CLR bit	CLR C	SWAP A	XCH A,dir	XCH A,@R0
	D	POP dir	ACALL S6+8ADR	SETB bit	SETB C	DA A	DJNZ dir,rel	XCHD A,@R0
	E	MOVX A, @DPTR	AJMP S7+8ADR	MOVX A,@R0	MOVX A,@R1	CLR A	MOV A,dir	MOV A,@R0
	F	MOVX @DPTR,A	ACALL S7+8ADR	MOVX @R0,A	MOVX @R1,A	CPL A	MOV dir,A	MOV @R0,A
		0	1	2	3	4	5	6
Младшие разряды								
	0	1	2	3	4	5	6	7

ТАБЛИЦА ДИЗАССЕМБЛЕРА MCS-51

Младшие разряды								
8	9	A	B	C	D	E	F	
INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7	0
DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7	1
ADD A,R0	ADD A,R1	ADD A,R2	ADD A,R3	ADD A,R4	ADD A,R5	ADD A,R6	ADD A,R7	2
ADDC A,R0	ADDC A,R1	ADDC A,R2	ADDC A,R3	ADDC A,R4	ADDC A,R5	ADDC A,R6	ADDC A,R7	3
ORL A,R0	ORL A,R1	ORL A,R2	ORL A,R3	ORL A,R4	ORL A,R5	ORL A,R6	ORL A,R7	4
ANL A,R0	ANL A,R1	ANL A,R2	ANL A,R3	ANL A,R4	ANL A,R5	ANL A,R6	ANL A,R7	5
XRL A,R0	XRL A,R1	XRL A,R2	XRL A,R3	XRL A,R4	XRL A,R5	XRL A,R6	XRL A,R7	6
MOV R0, #DAT	MOV R1, #DAT	MOV R2, #DAT	MOV R3, #DAT	MOV R4, #DAT	MOV R5, #DAT	MOV R6, #DAT	MOV R7, #DAT	7
MOV dir,R0	MOV dir,R1	MOV dir,R2	MOV dir,R3	MOV dir,R4	MOV dir,R5	MOV dir,R6	MOV dir,R7	8
SUBB A,R0	SUBB A,R1	SUBB A,R2	SUBB A,R3	SUBB A,R4	SUBB A,R5	SUBB A,R6	SUBB A,R7	9
MOV R0,dir	MOV R1,dir	MOV R2,dir	MOV R3,dir	MOV R4,dir	MOV R5,dir	MOV R6,dir	MOV R7,dir	A
CJNE R0,#D, rel	CJNE R1,#D, rel	CJNE R2,#D, rel	CJNE R3,#D, rel	CJNE R4,#D, rel	CJNE R5,#D, rel	CJNE R6,#D, rel	CJNE R7,#D, rel	B
XCH A,R0	XCH A,R1	XCH A,R2	XCH A,R3	XCH A,R4	XCH A,R5	XCH A,R6	XCH A,R7	C
DJNZ R0,rel	DJNZ R1,rel	DJNZ R2,rel	DJNZ R3,rel	DJNZ R4,rel	DJNZ R5,rel	DJNZ R6,rel	DJNZ R7,rel	D
MOV A,R0	MOV A,R1	MOV A,R2	MOV A,R3	MOV A,R4	MOV A,R5	MOV A,R6	MOV A,R7	E
MOV R0,A	MOV R1,A	MOV R2,A	MOV R3,A	MOV R4,A	MOV R5,A	MOV R6,A	MOV R7,A	F
8	9	A	B	C	D	E	F	
Младшие разряды								

С
т
а
р
ш
и
е

р
а
з
р
я
д
ы

Команды **CLR** сбрасывают бит в нуль. Команды **SETB** устанавливают бит в единицу. Команды **CPL** инвертируют значение выбранного бита.

Битовые логические команды выполняются во флаге C (CARY), как в Булевом Аккумуляторе.

Логические битовые команды **ANL C,/bit**, **ORL C,/bit** выполняются над содержимым флага CARY и инверсным значением выбранного бита. При этом значение самого бита не изменяется.

Команды безусловных переходов **JMP** или вызова подпрограмм **CALL** могут осуществлять переход в пределах:

- **LJMP met**, **LCALL met** – переход по абсолютному адресу в пределах всего адресного пространства Памяти Программ (64 Кбайта);
- **AJMP met**, **ACALL met** – переход по абсолютному адресу в пределах одной страницы Памяти Программ размером 2 Кбайта;
- **SJMP met** – относительный переход на 128 байт назад или на 127 байт вперед (однобайтовое относительное смещение в дополнительном коде указывается в коде команды).

Все команды условных переходов выполняют относительный переход по однобайтовому смещению в дополнительном коде.

Команды **JB bit,met** выполняют переход на метку, если выбранный бит установлен. Команда **JNB bit,met** выполняет переход на метку, если выбранный бит очищен (сброшен). Команды **JBC bit,met** выполняют переход на метку, если выбранный бит установлен, а после перехода очищают (сбрасывают) бит.

Команды цикла **DJNZ** (аналог команды **LOOP** процессоров x86) выполняют сначала декремент счетчика циклов (который указывается в команде) и переход на метку, если результат декремента не равен нулю.

Команды сравнения с условным переходом **CJNE** вычитают из первого операнда второй, результат никуда не записывают (не разрушают операнды) и осуществляют переход на метку, если операнды не равны. Но в этих командах выставляется также флаг **Cary**, который можно использовать для последующих условных переходов по условию: меньше – больше.

Таблица 8 – КОМАНДЫ ПЕРЕСЫЛКИ, ОБМЕНА И ЗАГРУЗКИ

Комментарии	Мнемокод	К О П	ТК	Б	МЦ
$A \leftarrow R_n$; Пересылка из РОНа в Акк	MOV A,Rn	11101rrr	1	1	1
$A \leftarrow (R_i)$; Пересылка из РПД в Акк	MOV A,@Ri	1110011i	1	1	1
$A \leftarrow (dir)$; Пересылка из РПД в Акк	MOV A,dir	11100101	3	2	1
$A \leftarrow \#DAT$; Загрузка байта в Акк	MOV A,#DAT	01110100	2	2	1
$R_n \leftarrow A$; Пересылка из Акк. в РОН	MOV Rn,A	11111rrr	1	1	1
$(R_i) \leftarrow A$; Пересылка из Акк. в РПД	MOV @Ri,A	1111011i	1	1	1
$(dir) \leftarrow A$; Пересылка из Акк. в РПД	MOV dir,A	11110101	3	2	1
$(dir) \leftarrow \#DAT$; Загрузка байта в РПД	MOV dir,#DAT	01110101	7	3	2
$R_n \leftarrow \#DAT$; Загрузка байта в РОН	MOV Rn,#DAT	01111rrr	2	2	1
$(R_i) \leftarrow \#DAT$; Загрузка байта в РПД	MOV @Ri,#DAT	0111011i	2	2	1
$(dir) \leftarrow (dir)$; Пересыл. из РПД в РПД	MOV dir,dir	10000101	9	3	2
$R_n \leftarrow (dir)$; Пересылка из РПД в РОН	MOV Rn,dir	10101rrr	3	2	2
$(R_i) \leftarrow (dir)$; Пересылка из РПД в РПД	MOV @Ri,dir	1010011i	3	2	2
$(dir) \leftarrow R_n$; Пересылка из РОНа в РПД	MOV dir,Rn	10001rrr	3	2	2
$(dir) \leftarrow (R_i)$; Пересылка из РПД в РПД	MOV dir,@Ri	1000011i	3	2	2
$DPTR \leftarrow \#D16$; Загрузка слова в DPTR	MOV DPTR,#D16	10010000	13	3	2
$A \leftrightarrow R_n$; Обмен Акк. с РОНом	XCH A,Rn	11001rrr	1	1	1
$A \leftrightarrow (R_i)$; Обмен Акк. с РПД	XCH A,@Ri	1100011i	1	1	1
$A \leftrightarrow (dir)$; Обмен Акк. с РПД	XCH A,dir	11000101	3	2	1
Обмен младших тетрад Акк. с РПД	XCHD A,@Ri	1101011i	1	1	1
Обмен тетрад Аккумулятора (Циклический сдвиг Акк. на 4 бита)	SWAP A	11000100	1	1	1
$SP \leftarrow SP+1$; $(SP) \leftarrow (dir)$; Запись в стек	PUSH dir	11000000	3	2	2
$(dir) \leftarrow (SP)$; $SP \leftarrow SP-1$; Чтение из стека	POP dir	11010000	3	2	2
$A \leftarrow (DPTR)$; Пересылка из ВПД в Акк.	MOVX A,@DPTR	11100000	1	1	2
$(DPTR) \leftarrow A$; Пересылка из Акк. в ВПД	MOVX @DPTR,A	11110000	1	1	2
$A \leftarrow (R_i)$; Пересылка из ВПД в Акк.	MOVX A,@Ri	1110001i	1	1	2
$(R_i) \leftarrow A$; Пересылка из Акк. в ВПД	MOVX @Ri,A	1111001i	1	1	2
$A \leftarrow ((A+DPTR))$; Пересылка байта из ПП в Акк.	MOVC A,@A+DPTR	10010011	1	1	2
$A \leftarrow ((A+PC))$; Пересылка из ПП в Акк.	MOVC A,@A+PC	10000011	1	1	2

Таблица 9 – АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ КОМАНДЫ

Комментарии	Мнемокод	К О П	ТК	Б	МЦ
$A \leftarrow A+R_n$; Сложение Акк. с РОНом	ADD A,Rn	00101rrr	1	1	1
$A \leftarrow A+(R_i)$; Сложение Акк. с РПД	ADD A,@Ri	0010011i	1	1	1
$A \leftarrow A+(dir)$; Сложение Акк. с РПД	ADD A,dir	00100101	3	2	1
$A \leftarrow A+\#DAT$; Сложение Акк. с #DAT	ADD A,#DAT	00100100	2	2	1
$A \leftarrow A + R_n + C$;	ADDC A,Rn	00111rrr	1	1	1
$A \leftarrow A + (R_i) + C$;	ADDC A,@Ri	0011011i	1	1	1
$A \leftarrow A + (dir) + C$;	ADDC A,dir	00110101	3	2	1
$A \leftarrow A + \#DAT + C$;	ADDC A,#DAT	00110100	2	2	1

Десятичная коррекция Аккумулятора после сложения	DA A	11010100	1	1	1
$A \leftarrow A - R_n - C;$	SUBB A,Rn	10011rrr	1	1	1
$A \leftarrow A - (R_i) - C;$	SUBB A,@Ri	1001011i	1	1	1
$A \leftarrow A - (dir) - C;$	SUBB A,dir	10010101	3	2	1
$A \leftarrow A - \#DAT - C;$	SUBB A,#DAT	10010100	2	2	1
$A \leq A + 1;$ Инкремент Аккумулят.	INC A	00000100	1	1	1
$R_n \leftarrow R_n + 1;$ Инкремент РОНа	INC Rn	00001rrr	1	1	1
$(R_i) \leftarrow (R_i) + 1;$ Инкремент РПД	INC @Ri	0000011i	1	1	1
$(dir) \leftarrow (dir) + 1;$ Инкремент РПД	INC dir	00000101	3	2	1
$DPTR \leftarrow DPTR + 1;$ Инкремент DPTR	INC DPTR	10100011	1	1	2
$A \leftarrow A - 1;$ Декремент Аккумулят.	DEC A	00010100	1	1	1
$R_n \leftarrow R_n - 1;$ Декремент РОНа	DEC Rn	00011rrr	1	1	1
$(R_i) \leftarrow (R_i) - 1;$ Декремент РПД	DEC @Ri	0001011i	1	1	1
$(dir) \leftarrow (dir) - 1;$ Декремент РПД	DEC dir	00010101	3	2	1
$BA \leftarrow A * B;$ Умножение A на B, $B \leftarrow$ старш. байт, $A \leftarrow$ младш. байт	MUL AB	10100100	1	1	4
$A.B \leftarrow A / B;$ Деление A на B, $A \leftarrow$ байт частного, $B \leftarrow$ остаток	DIV AB	10000100	1	1	4
$A \leftarrow A \& R_n;$ Лог. "И" Акк. и РОНа	ANL A,Rn	01011rrr	1	1	1
$A \leftarrow A \& (R_i);$ Лог."И" Акк. и РПД	ANL A,@Ri	0101011i	1	1	1
$A \leftarrow A \& (dir);$ Лог."И" Акк. и РПД	ANL A,dir	01010101	3	2	1
$A \leftarrow A \& \#DAT;$ Лог."И" Акк. и #DAT	ANL A,#DAT	01010100	2	2	1
$(dir) \leftarrow (dir) \& A;$ Лог."И" РПД и Акк	ANL dir,A	01010010	3	2	1
$(dir) \leftarrow (dir) \& \#DAT;$	ANL dir,#DAT	01010011	7	3	2
$A \leftarrow A \vee R_n;$ Лог."ИЛИ" Акк. и РОНа	ORL A,Rn	01001rrr	1	1	1
$A \leftarrow A \vee (R_i);$ Лог."ИЛИ" Акк. и РПД	ORL A,@Ri	0100011i	1	1	1
$A \leftarrow A \vee (dir);$ Лог."ИЛИ" Акк. и РПД	ORL A,dir	01000101	3	2	1
$A \leftarrow A \vee \#DAT;$	ORL A,#DAT	01000100	2	2	1
$(dir) \leftarrow (dir) \vee A;$	ORL dir,A	01000010	3	2	1
$(dir) \leftarrow (dir) \vee \#DAT;$	ORL dir,#DAT	01000011	7	3	2
$A \leftarrow A (+) R_n;$ "Искл.ИЛИ" Акк. и РОН	XRL A,Rn	01101rrr	1	1	1
$A \leftarrow A (+) (R_i);$	XRL A,@Ri	0110011i	1	1	1
$A \leftarrow A (+) (dir);$	XRL A,dir	01100101	3	2	1
$A \leftarrow A (+) \#DAT;$	XRL A,#DAT	01100100	2	2	1
$(dir) \leftarrow (dir) (+) A;$	XRL dir,A	01100010	3	2	1
$(dir) \leftarrow (dir) (+) \#DAT;$	XRL dir,#DAT	01100011	7	3	2
$A \leftarrow 0;$ Сброс Аккумулятора	CLR A	11100100	1	1	1
$A \leftarrow \sim A;$ Инверсия Аккумулятора	CPL A	11110100	1	1	1
Циклический сдвиг влево Акк.	RL A	00100011	1	1	1
Цикл.сдвиг влево Акк. через перенос	RLC A	00110011	1	1	1
Циклический сдвиг вправо Акк.	RR A	00000011	1	1	1
Цикл.сдвиг вправо Акк.через перенос	RRC A	00010011	1	1	1

Таблица 10 – ОПЕРАЦИИ С БИТАМИ

Комментарии	Мнемокод	К О П	ТК	Б	МЦ
$C \leftarrow 0$; Сброс флага C	CLR C	11000011	1	1	1
$C \leftarrow 1$; Установка флага C	SETB C	11010011	1	1	1
$C \leftarrow \sim C$; Инверсия флага C	CPL C	10110011	1	1	1
$(bit) \leftarrow 0$; Сброс прямоадресуем.бита	CLR bit	11000010	4	2	1
$(bit) \leftarrow 1$; Установка "бита"	SETB bit	11010010	4	2	1
$(bit) \leftarrow \sim (bit)$; Инверсия "бита"	CPL bit	10110010	4	2	1
$C \leftarrow C \& (bit)$; Лог."И" C и "бита"	ANL C,bit	10000010	4	2	2
$C \leftarrow C \& \sim (bit)$; "И" C и инверс.бита	ANL C,/bit	10110000	4	2	2
$C \leftarrow C \vee (bit)$; Лог."ИЛИ" C и бита	ORL C,bit	01110010	4	2	2
$C \leftarrow C \vee \sim (bit)$; Лог."ИЛИ" C и инверсии прямоадресуемого бита	ORL C,/bit	10100000	4	2	2
$C \leftarrow (bit)$; Пересылка из прямоадресуемого бита во флаг C	MOV C,bit	10100010	4	2	1
$(bit) \leftarrow C$; Пересылка из C в "бит"	MOV bit,C	10010010	4	2	2

Таблица 11 – КОМАНДЫ УСЛОВНЫХ ПЕРЕХОДОВ

Комментарии	Мнемокод	К О П	ТК	Б	МЦ
$PC \leftarrow PC + 2 + rel$, если $A = 0$	JZ rel	01100000	5	2	2
$PC \leftarrow PC + 2 + rel$, если $A \neq 0$	JNZ rel	01110000	5	2	2
$PC \leftarrow PC + 2 + rel$, если флаг C = 1	JC rel	01000000	5	2	2
$PC \leftarrow PC + 2 + rel$, если флаг C = 0	JNC rel	01010000	5	2	2
$PC \leftarrow PC + 3 + rel$, если $(bit) = 1$	JB bit,rel	00100000	11	3	2
$PC \leftarrow PC + 3 + rel$, если $(bit) = 0$	JNB bit,rel	00110000	11	3	2
$PC \leftarrow PC + 3 + rel$, если $(bit) = 1$, $(bit) \leftarrow 0$	JBC bit,rel	00010000	11	3	2
$PC \leftarrow PC + 3 + rel$, если $A \neq \#DAT$, $C \leftarrow 1$, если $A < \#DAT$, $C \leftarrow 0$, если $A > \#DAT$	CJNE A, #DAT, rel	10110100	10	3	2
$PC \leftarrow PC + 3 + rel$, если $R_n \neq \#DAT$, $C \leftarrow 1$, если $R_n < \#DAT$, $C \leftarrow 0$, если $R_n > \#DAT$	CJNE R_n , #DAT, rel	10111rrr	10	3	2
$PC \leftarrow PC + 3 + rel$, если $(@R_i) \neq \#DAT$, $C \leftarrow 1$, если $(@R_i) < \#DAT$, $C \leftarrow 0$, если $(@R_i) > \#DAT$	CJNE $@R_i$, #DAT, rel	1011011i	10	3	2
$PC \leftarrow PC + 3 + rel$, если $A \neq (dir)$, $C \leftarrow 1$, если $A < (dir)$, $C \leftarrow 0$, если $A > (dir)$	CJNE A, dir, rel	10110101	8	3	2
$PC \leftarrow PC + 2 + rel$, если $R_n - 1 \neq 0$	DJNZ R_n , rel	11011rrr	5	2	2
$PC \leftarrow PC + 3 + rel$, если $(dir) - 1 \neq 0$	DJNZ dir,rel	11010101	8	3	2

Таблица 12 – КОМАНДЫ БЕЗУСЛОВНЫХ ПЕРЕХОДОВ

Комментарии	Мнемокод	К О П	ТК	Б	МЦ
$PC[0_15] \leftarrow 16ADR$; Длинный переход	LJMP 16ADR	00000010	12	3	2
$PC[0_11] \leftarrow 11ADR$; Переход внутри страницы в 2 Кбайта	AJMP 11ADR	sss00001	6	2	2
$PC \leftarrow PC + 2 + rel$; Относит.переход	SJMP rel	10000000	5	2	2
$PC \leftarrow A + DPTR$; Косвенный относительный переход	JMP @A+DPTR	01110011	1	1	2
$(SP) \leftarrow PC + 3$; $PC[0_15] \leftarrow 16ADR$; Длинный вызов подпрограммы	LCALL 16ADR	00010010	12	3	2
$(SP) \leftarrow PC + 2$; $PC[0_10] \leftarrow 11ADR$; Вызов подпр. внутри страницы в 2Кб	ACALL 11ADR	sss10001	6	2	2
$PC \leftarrow (SP)$; Возврат из подпрограммы	RET	00100010	1	1	2
$PC \leftarrow (SP)$; Возврат из подпро- граммы обработки прерываний	RETI	00110010	1	1	2
$PC \leftarrow PC + 1$; Холостая команда	NOP	00000000	1	1	1

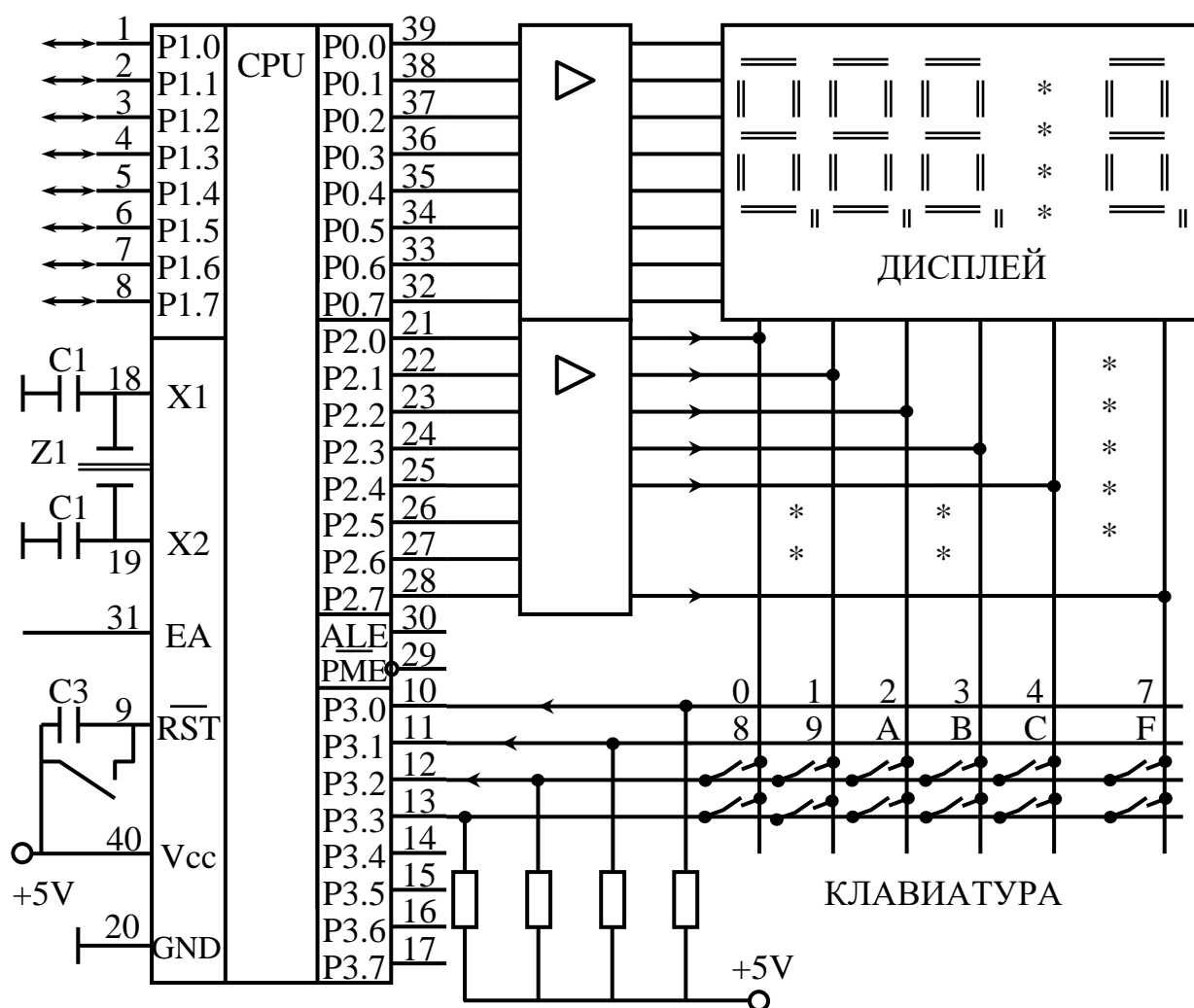


Рис. 4 – Ввод и отображение информации в МКС на основе MCS-51

В таблицах 8...12 введены обозначения:

- КОП – бинарный код операции (команды);
- ТК – тип команды (см. рис. 2);
- Б – количество байт в команде;
- МЦ – количество машинных циклов ($T_{\text{МЦ}} = 12 / F_t$) для выполнения каждой команды

3 АППАРАТНАЯ РЕАЛИЗАЦИЯ МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ

Сложные микроконтроллерные системы могут состоять только из одного микроконтроллера (см. рис.4). Дополнительные элементы выполняют функции усилителей мощности (для управления светодиодными семисегментными индикаторами) или преобразователей уровней сигналов.

На выходе порта P2 формируются сигналы типа «бегущий ноль» для динамического управления индикаторами и сканирования столбцов матрицы клавиатуры.

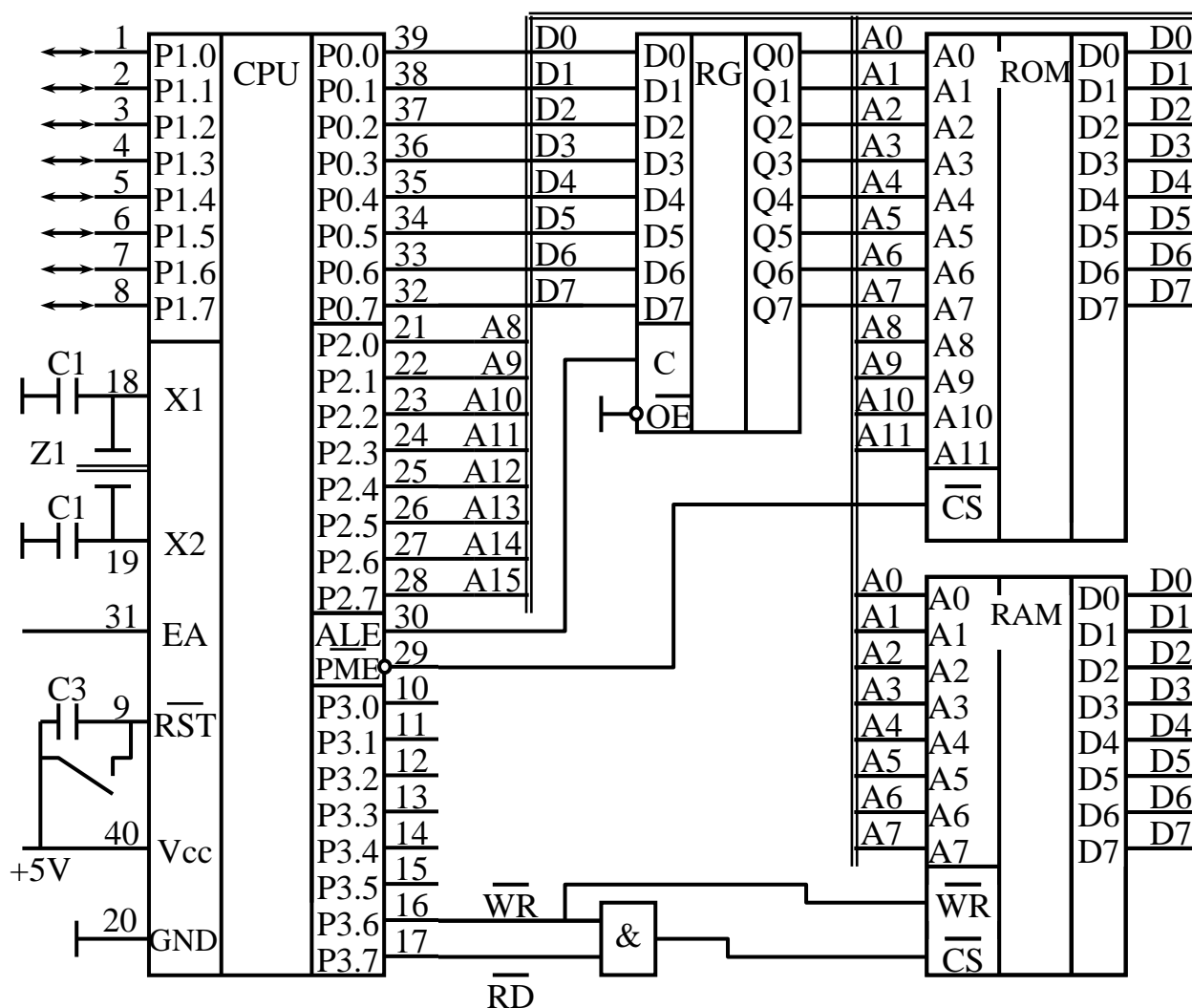


Рис. 5 – Расширение памяти программ и памяти данных MCS-51

При расширении памяти программ или памяти данных, кроме собственно чипов памяти, необходим дополнительный параллельный регистр (см. рис. 5) для запоминания младшего байта адреса с выходов порта P0. После этого микроконтроллер обменивается с внешней памятью одним байтом данных по линиям порта P0. Запоминать старший байт адреса с выходов порта P2 не нужно, потому что адресная информация на этих выводах не изменяется во время всего цикла обмена

4.1 ТАЙМЕРЫ / СЧЕТЧИКИ ВНЕШНИХ СОБЫТИЙ

Два программируемых 16-ти битовых таймера/счетчика (Т/С 0 и Т/С 1) могут быть использованы в качестве таймеров или счетчиков внешних событий. При работе в качестве таймера содержимое Т/С инкрементируется в каждом машинном цикле, т. е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое Т/С инкрементируется под воздействием перехода "1-0" внешнего входного сигнала, подаваемого на соответствующий вывод (Т0 или Т1). Максимальная входная частота счетчиков: $F_t / 24$.

РЕЖИМЫ РАБОТЫ Т/С определяются кодом, записанным в РЕГИСТР РЕЖИМОВ Т/С (TMOD):

РЕГИСТР РЕЖИМОВ ТАЙМЕРОВ / СЧЕТЧИКОВ (TMOD)

Прямой адрес TMOD: dir – 89H.

	7	6	5	4	3	2	1	0
TMOD :	GATE 1	C/~T 1	M1.1	M0.1	GATE 0	C/~T 0	M1.0	M0.0

Таблица 13 – Выбор режимов таймеров / счетчиков (TMOD)

Биты TMOD	Обозначение	Выбор режима															
0, 1 4, 5	M0, M1	<p>Определяют один из 4-х режимов работы, отдельно для Т/С 1 и Т/С 0 :</p> <table> <tr> <th>M1</th><th>M0</th><th>Режим</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>2</td></tr> <tr> <td>1</td><td>1</td><td>3</td></tr> </table>	M1	M0	Режим	0	0	0	0	1	1	1	0	2	1	1	3
M1	M0	Режим															
0	0	0															
0	1	1															
1	0	2															
1	1	3															
2 6	C/~T 0 C/~T 1	<p>Определяет работу отдельно для каждого счетчика в режиме :</p> <p>C/~T = 0 – таймера; C/~T = 1 – счетчика внешних событий.</p>															
3 7	GATE 0 GATE 1	<p>Разрешает управлять счетчиком от внешнего вывода (~INT0 – для Т/С 0, ~INT1 – для Т/С 1) :</p> <p>GATE = 0 – управление запрещено, GATE = 1 – управление разрешено.</p>															

РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА Т/С и внешними прерываниями (TCON) предназначен для приема и хранения кодов управляющего слова.

РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА Т/С И ВНЕШНИМИ ПЕРЕРЫВАНИЯМИ (TCON)

Прямой байтовый адрес TCON: dir – 88h.

Допускается адресация отдельных бит TCON : bit – 88h_8Fh.

	7	6	5	4	3	2	1	0
Адрес : bit	8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h
TCON	TF 1	TR 1	TF 0	TR 0	IE 1	IT 1	IE 0	IT 0

Таблица 14 – Назначение битов TCON

Биты TMOD	Обоз- на- чение	Назначение разрядов TCON
5 7	TF 0 TF 1	Флаги переполнения Т/С, устанавливаются аппаратно при переполнении соответствующего Т/С (переходе из состояния «все единицы» в состояние «все нули»). Если прерывание от соответствующего Т/С разрешено, то установка флага TF вызовет прерывание. Флаги TF 0 или TF 1 сбрасываются аппаратно при передаче управления подпрограмме обработки соответствующего прерывания
4 6	TR 0 TR 1	Разрешение счета отдельно для каждого Т/С : TR = 0 – счет остановлен, TR = 1 – разрешение счета.
1 3	IE 0 IE 1	Флаги запроса внешних прерываний по входам \sim INT0 и \sim INT1 соответственно; устанавливаются аппаратно (от внешних устройств) или программно и вызывают подпрограмму обработки прерываний. Если прерывание вызвано по фронту сигнала, эти флаги сбрасываются аппаратно при переходе к подпрограмме. Если прерывание было вызвано низким уровнем на входе \sim INT0 (\sim INT1), то сброс флага должна выполнять подпрограмма обслуживания прерывания, воздействуя на источник прерывания для снятия запроса.
0 2	IT 0 IT 1	Управление видом прерывания отдельно по входам \sim INT 0 или \sim INT 1 : IT = 0 – прерывание по уровню (низкому), IT = 1 – прерывание по фронту «1–0»

РЕЖИМ РАБОТЫ «0» ($M0=0, M1=0$) функционально совместим с таймером/счетчиком микроконтроллера MCS-48. Деление импульсов Машинных Циклов (МЦ) на 32 выполняют 5 младших разрядов регистров TL 0, TL 1.

Логика работы в РЕЖИМЕ 0 на примере Т/С 0 показана на рис. 6. Для Т/С 1 логика работы аналогична.

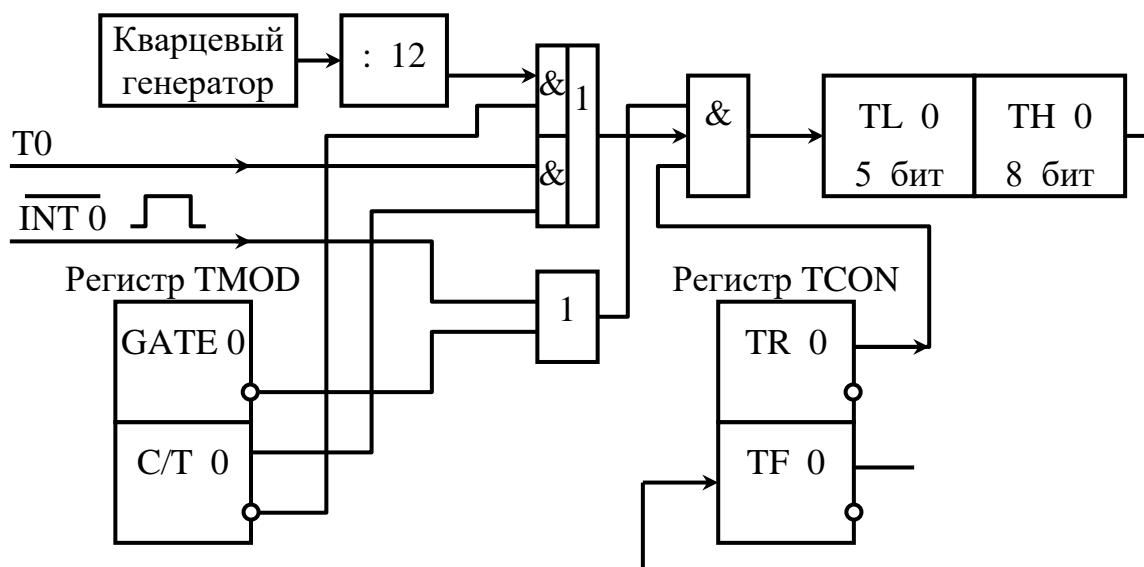


Рис. 6 – Логика работы Т/С 0 в РЕЖИМЕ 0 (в РЕЖИМЕ 1 TL 0 – 8 бит)

Счет начинается при установке бита TR 0 регистра TCON в состояние «1». (Если бит TR = 0, то регистры соответствующих таймеров/ счетчиков TH и TL могут использоваться как дополнительные РОНЫ).

Установка бита GATE в единичное состояние позволяет в режиме внутреннего таймера измерять длительность импульсного сигнала, подаваемого на вход внешнего прерывания \overline{INT} .

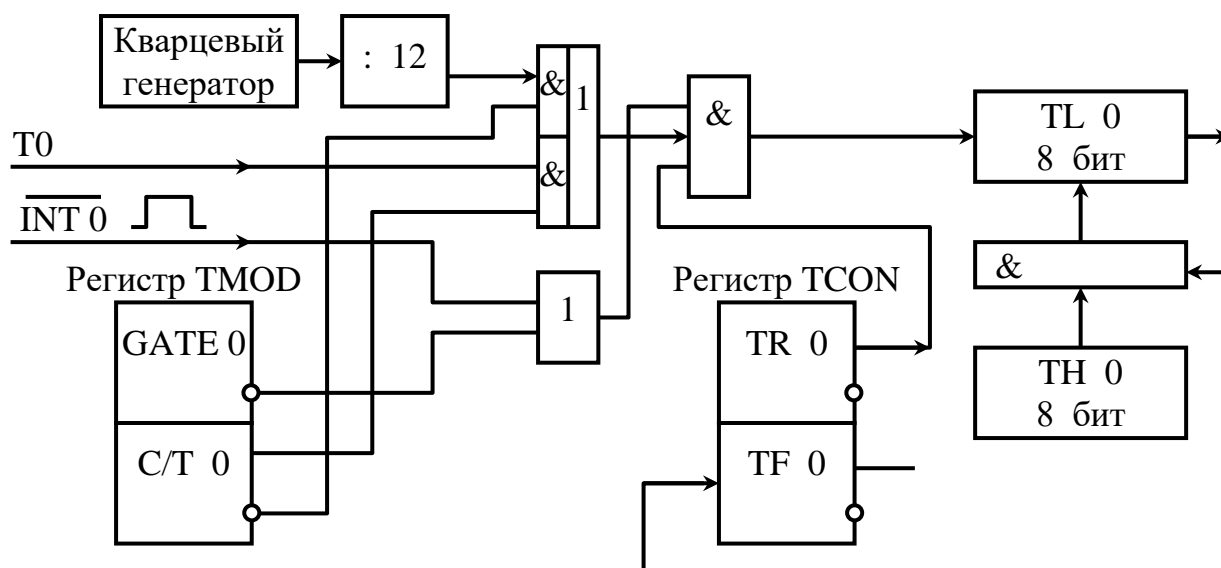


Рис. 7 – Логика работы Т/С 0 в РЕЖИМЕ «2»

РЕЖИМ РАБОТЫ «1» ($M0=1$, $M1=0$) аналогичен РЕЖИМУ «0». Отличие состоит в том, что таймерные регистры TL, TH - 16-ти разрядные.

РЕЖИМ РАБОТЫ «2» ($M0=0$, $M1=1$) представляет собой 8-ми разрядный делитель TL 0 (или TL 1) с переменным (программируемым) коэффициентом деления. При каждом переполнении 8-ми разрядного счетчика TL 0 устанавливается флаг TF 0 и происходит перезагрузка счетчика TL 0 из регистра TH 0 (рис. 7). Для Т/С 1 логика работы аналогична.

РЕЖИМ РАБОТЫ «3» различный для Т/С 0 и Т/С 1.

Счетчик Т/С 1 бессмысленно программировать в режиме «3», потому что он будет заблокирован (сохраняет свое текущее значение).

Счетчик Т/С 0 в РЕЖИМЕ «3» представляет собой два независимых 8-ми разрядных счетчика TL 0 и TH 0.

TL 0 может работать в режиме таймера и в режиме счетчика. За ним сохраняются все биты управления Т/С 0 и входные сигналы T0, $\overline{INT0}$ (см. рис. 8). TH 0 может работать только в режиме таймера, использует бит включения TR 1 и выставляет флаг переполнения TF 1 (рис. 8).

Этот режим позволяет реализовать два восьмибитовых таймера из Т/С 0, если Т/С 1 уже занят – формирует частоту обмена для последовательного интерфейса (последовательного порта).

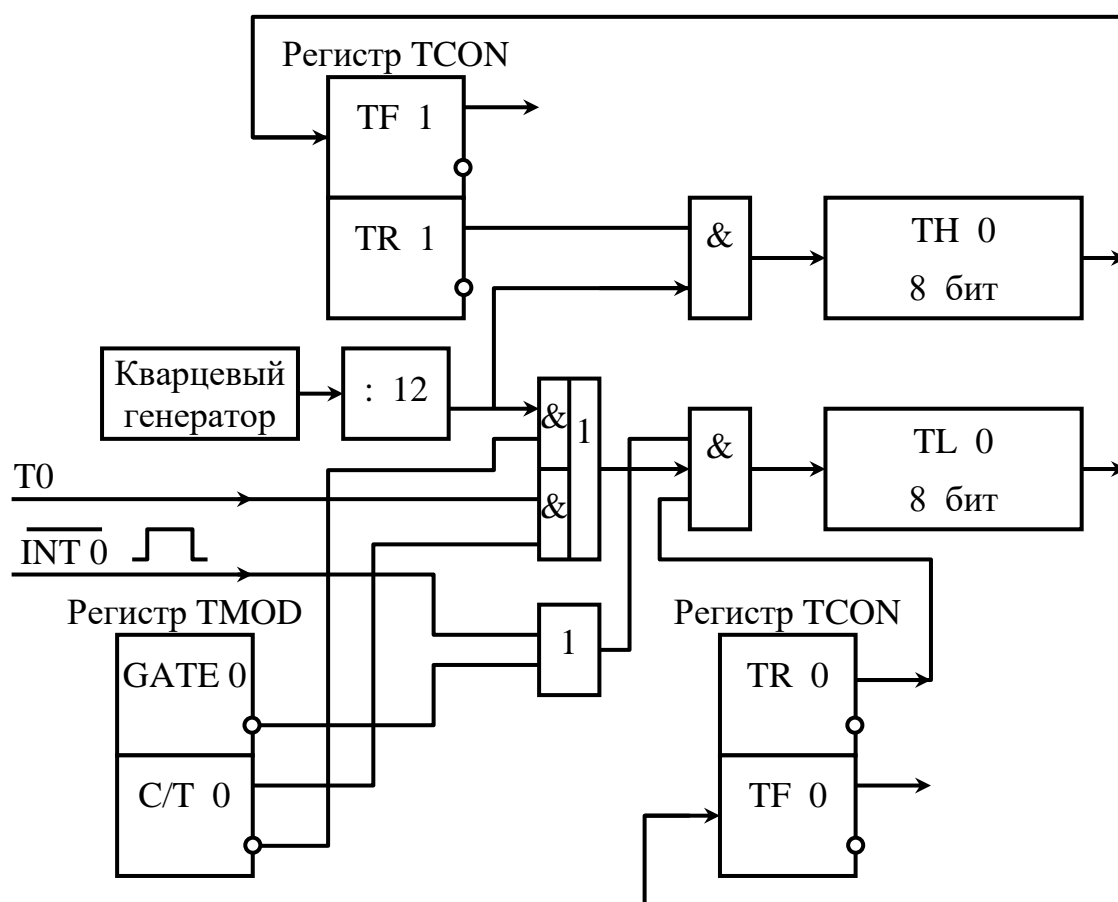


Рис. 8 – Логика работы Т/С 0 в РЕЖИМЕ «3»

4.2 РЕЖИМЫ ПРЕРЫВАНИЯ МИКРОКОНТРОЛЛЕРОВ MCS-51

Запросы от внешних прерываний $\sim\text{INT0}$, $\sim\text{INT1}$ фиксируются в триггерах IE0 , IE1 Регистра Управления Т/С и внешними прерываиями (TCON). Установка этих триггеров осуществляется низким уровнем на входах $\sim\text{INT0}$, $\sim\text{INT1}$ (если сброшены биты $\text{IT0} = 0$, $\text{IT1} = 0$ регистра TCON), или по фронту «1-0» (если биты установлены: $\text{IT0} = 1$, $\text{IT1} = 1$).

Запросы прерываний от Таймеров/Счетчиков фиксируются в триггерах TF0 , TF1 регистра управления TCON .

Запрос прерывания последовательного порта вызывается установкой флага прерывания приемника RI или флага прерывания передатчика TI в регистре SCON . В отличие от всех остальных флагов, RI и TI сбрасываются только программным путем (обычно в пределах подпрограммы обработки прерывания, где определяется: какому из флагов RI или TI соответствует прерывание).

Все перечисленные флаги прерываний : IE0 , IE1 , TF0 , TF1 , RI , TI – могут быть установлены (или сброшены) программно и вызвать соответствующие прерывания.

Прерывание по каждому из перечисленных источников может быть разрешено или запрещено установкой или сбросом соответствующего бита в РЕГИСТРЕ МАСКИ (Разрешения) ПРЕРЫВАНИЙ – (IE) :

РЕГИСТР МАСКИ (РАЗРЕШЕНИЯ) ПРЕРЫВАНИЙ – (IE)

Прямой байтовый адрес IE : $\text{dir} - 0\text{A8h}$.

Допускается адресация отдельных бит IE : $\text{bit} - 0\text{A8h_0AFh}$.

	7	6	5	4	3	2	1	0
Адрес : bit	0AFh			0ACH	0ABh	0AAh	0A9h	0A8h
IE	EA			ES	ET 1	EX 1	ET 0	EX 0

- EA – управление всеми источниками прерываний; если $\text{EA} = 0$ – все прерывания запрещены; если $\text{EA} = 1$ – прерывания могут быть разрешены индивидуально;
- $\text{ES} = 1$ – разрешение прерывания от последовательного порта, при $\text{ES} = 0$ – запрещение прерывания;
- $\text{ET1} = 1$ – разрешение прерывания от Т/С 1; $\text{ET1} = 0$ – запрет;
- $\text{EX1} = 1$ – разрешение прерывания от внешнего источника $\sim\text{INT1}$, при $\text{EX1} = 0$ – запрет;
- $\text{ET0} = 1$ – разрешение прерывания от Т/С 0; $\text{ET0} = 0$ – запрет;
- $\text{EX0} = 1$ – разрешение прерывания от внешнего источника $\sim\text{INT0}$, при $\text{EX0} = 0$ – запрет.

РЕГИСТР ПРИОРИТЕТОВ ПРЕРЫВАНИЙ – IP предназначен для установки уровня приоритета прерывания для каждого из пяти источников прерываний :

- PS – установка уровня приоритета прерывания от последовательного порта;
- PT1 – установка уровня приоритета прерывания от Т/С 1;
- PX1 – установка уровня приоритета прерывания от внешнего источника \sim INT1;
- PT0 – установка уровня приоритета прерывания от Т/С 0;
- PX0 – установка уровня приоритета прерывания от внешнего источника \sim INT0.

РЕГИСТР ПРИОРИТЕТОВ ПРЕРЫВАНИЙ – IP

Прямой байтовый адрес IP: dir – 0B8h.

Допускается адресация отдельных бит IP : bit – 0B8h_0BFh.

	7	6	5	4	3	2	1	0
Адрес : bit				0BCh	0BBh	0BAh	0B9h	0B8h
IP				PS	PT 1	PX 1	PT 0	PX 0

Наличие в разряде регистра IP «1» устанавливает для соответствующего источника высокий уровень приоритета, а наличие «0» – низкий уровень приоритета.

Программа обработки прерывания с низким уровнем приоритета может быть прервана запросом прерывания с высоким уровнем приоритета, но не может быть прервана другим запросом прерывания с низким уровнем приоритета. Программа обработки прерывания с высоким уровнем приоритета не может быть прервана никаким другим запросом прерывания.

Если два запроса с разными уровнями приоритета приняты одновременно, сначала будет обслужен запрос с высоким уровнем приоритета.

Если одновременно приняты запросы с одинаковым уровнем приоритета, обработка их будет производиться в порядке, задаваемом последовательностью внутреннего опроса флагов прерываний. Таким образом, в пределах одного приоритетного уровня существует еще одна структура приоритетов (табл. 15):

При переходе по вектору на подпрограмму обработки прерывания аппаратно запрещаются все прерывания с уровнем приоритета, равным (или меньшим) уровню приоритета обслуживаемого прерывания.

Таблица 15 – Приоритеты внутри одного уровня и векторы прерываний

Источник прерывания	Приоритет внутри уровня	Векторы прерываний в адресном пространстве Прогр. Памяти
Внешнее прерывание \sim INT 0	высший	0003h
Таймер/Счетчик T/C 0		000Bh



Внешнее прерывание	\sim INT 1	низший	0013h
Таймер/Счетчик	T/C 1		001Bh
Последовательный порт			0023h

Подпрограмма обслуживания прерывания должна заканчиваться выполнением команды RETI, которая восстанавливает состояние логики прерывания и загружает из стека в счетчик команд (PC) адрес возврата в исходную программу. При использовании команды RET восстанавливается только счетчик команд (PC) из стека. Состояние логики прерывания команда RET не меняет, т. е. сохраняется запрет на прерывания с равным (или меньшим) приоритетом.

4.3 ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС МИКРОКОНТРОЛЛЕРОВ MCS-51

Через Универсальный Асинхронный Приемо-Передачик (УАПП) осуществляется прием и передача информации, представленной последовательным кодом (*младшими битами вперед*), в полном дуплексном режиме обмена (как у COM-порта компьютера). В состав УАПП (или Последовательного Порта) входят :

- принимающий и
- передающий сдвигающие регистры, а также
- специальный буферный регистр (SBUF) приемопередатчика.

Запись байта в буфер SBUF приводит к автоматической перезаписи байта в сдвигающий регистр передатчика и инициирует начало последовательной передачи байта. Наличие буферного регистра приемника SBUF позволяет совмещать операцию чтения ранее принятого байта с последовательным приемом очередного байта. Если к моменту окончания приема байта предыдущий байт не был считан из SBUF, то он будет потерян.

Управление режимами работы УАПП определяется кодом, записанным в **РЕГИСТР УПРАВЛЕНИЯ/СТАТУСА ПОСЛЕДОВАТЕЛЬНОГО ПОРТА (SCON)**:

Прямой байтовый адрес SCON: dir – 98h.
Допускается адресация отдельных бит SCON : bit – 98h_9Fh.

	7	6	5	4	3	2	1	0
Адрес : bit	9Fh	9Eh	9Dh	9Ch	9Bh	9Ah	99h	98h
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

- SM0, SM1 – определяют режимы работы УАПП (см. табл. 16);
- SM2 – разрешение многопроцессорной работы; (в режимах 2 и 3 при SM2 = 1 бит прерывания R1 не устанавливается, если принятый девятый бит данных RB8 = 0);

- REN – разрешение ПРИЕМА последовательных данных:
REN = 1 – разрешение приема,
REN = 0 – запрет приема;
- TB8 – девятый бит передаваемых данных в режимах 2 и 3; устанавливается и сбрасывается программно;
- RB8 – девятый бит принятых данных в режимах 2 и 3;
- TI – флаг прерывания передатчика; устанавливается аппаратно в конце выдачи 8-го бита в режиме 0 или в начале стоп-бита – в других режимах; сбрасывается программой;
- RI – флаг прерывания приемника; устанавливается аппаратно в конце приема 8-го бита в режиме 0 или в середине стоп-бита – в других режимах; сбрасывается программой.

Таблица 16 - Режимы работы последовательного порта УАПП

SM0,SM1	Режим	Наименование	Скорость обмена
0 0	0	Передача и прием 8-ми битовых данных через двунаправленный вывод RxD ; через вывод TxD выдаются синхроим-пульсы сдвига	Ft / 12
0 1	1	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 8-ми битовых данных и стоп-бита (1)	Fov / 16, при SMOD=1 Fov / 32 при SMOD=0
1 0	2	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 9-ти битовых данных и стоп-бита (1)	Ft / 32, при SMOD=1 Ft / 64 при SMOD=0
1 1	3	Передача (через вывод TxD) и прием (через вывод RxD): старт-бита (0), 9-ти битовых данных и стоп-бита (1)	Fov / 16, при SMOD=1 Fov / 32 при SMOD=0

Fov – частота переполнения T/C1, работающего в режиме таймера или в режиме счетчика внешних событий. (Прерывание от T/C1 должно быть запрещено).

SMOD – старший бит регистра управления мощностью PCON.

В режиме 2 и 3 в многопроцессорных системах один из контроллеров (или IBM PC) играет роль ведущего, а остальные – ведомые. Механизм такой работы аппаратно поддерживается битом SM2 регистра SCON.

Ведущий микроконтроллер посылает вначале посылки «байт адреса». Адрес отличается от данных тем, что его девятый бит установлен в «1», а у данных – девятый бит равен «0». При SM2 = 1 байт адреса вызывает преры-

вание, а байт данных – нет. Микроконтроллер, у которого байт адреса совпал с собственным кодом, сбрасывает бит SM2 и имеет возможность принимать следующие за ним байты данных. Остальные ведомые оставляют бит SM2 установленным и не реагируют на последующие байты данных.

Таблица 17 – Программирование УАПП для стандартных скоростей обмена

Скорость приема/передачи, Кбод	Ft, МГц	Регистр SCON Режим УАПП		Регистр PCON SMOD	Регистр TMOD для T/C 1			Таймер T/C 1	
		SM0	SM1		C/~T	M1	M0	TH 1	TL 1
1000	12	0	0						
375	12	1	0	1					
62,5	12	*	1	1	0	1	0	0FFh	
19,2	11,059	*	1	1	0	1	0	0FDh	
9,6	11,059	*	1	0	0	1	0	0FDh	
4,8	11,059	*	1	0	0	1	0	0FAh	
2,4	11,059	*	1	0	0	1	0	0E4h	
1,2	11,059	*	1	0	0	1	0	0E8h	
0,1375	11,059	*	1	0	0	1	0	18h	
0,110	6	*	1	0	0	1	0	72h0	
0,110	12	*	1	0	0	0	1	0FEh	0EBh

* – бит SM0 = 0 для режима 1 УАПП (8-ми битовые данные),
бит SM0 = 1 для режима 3 УАПП (9-ми битовые данные).

РЕГИСТР УПРАВЛЕНИЯ МОЩНОСТЬЮ ПОТРЕБЛЕНИЯ (PCON)

Прямой адрес PCON: dir – 87H.

	7	6	5	4	3	2	1	0
PCON :	SMOD				GF 1	GF0	PD	IDL

- SMOD – бит удвоения скорости приема/передачи УАПП (см. табл. 16 и табл. 17);
- GF1, GF0 – резервные биты, содержат триггеры, доступные по записи и чтению;
- PD – бит включения режима микропотребления;
- IDL – бит включения режима холостого хода.

5 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ МИКРОКОНТРОЛЛЕРОВ ТИПА 8052

Этот микроконтроллер отличается от базовой модели 80C51 наличием внутреннего ОЗУ (РПД) размером 256 байт и третьего таймера-счетчика Т/С 2. Вывод P1.0 получил альтернативную функцию Т2 (счетный вход Таймера 2), а вывод P1.1 – альтернативную функцию Т2ЕХ (вход управления фиксацией/перезагрузкой значения Таймера 2). Последовательный порт этих микроконтроллеров может тактироваться от Таймера 2.

Увеличение размера резидентной памяти данных привело к наложению старших 128 байт ОЗУ данных и пространства регистров специальных функций. Поэтому обращение к регистрам специальных функций (RSF) производится с использованием прямой байтовой адресации (как и в базовой модели 80C51), а к старшим 128 байтам РПД – только с использованием косвенной адресации.

К набору регистров специальных функций добавлены регистры данных (TL 2, TH 2), перезагрузки/захвата (RCAP2L, RCAP2H) и управления (T2CON) Таймера 2.

5.1 ТАЙМЕР 2

Таймер 2 представляет собой 16-ти разрядный таймер-счетчик, который может работать в трех режимах:

- в режиме захвата (фиксации) текущего значения Таймера 2;
- в режиме прямого счета с автоперезагрузкой исходного значения (16-ти разрядный счетчик с программируемым коэффициентом деления – ДПКД);
- в режиме задающего генератора для последовательного порта.

Шестнадцатиразрядный регистр данных Таймера 2 состоит из младшего байта – TL2 (прямой адрес – 0CCh) и старшего байта TH2 (прямой адрес – 0CDh). Данные для автоперезагрузки хранятся в регистрах: младший байт – RCAP2L (прямой адрес – 0CAh), старший байт – RCAP2H (прямой адрес – 0CBh).

Захват (фиксация) текущего значения из регистров TH2_TL2 осуществляется в регистры RCAP2H_RCAP2L с установкой флага переполнения EXF2. В режиме счетчика с программируемым коэффициентом деления в регистры TH2_TL2 из регистров RCAP2H_RCAP2L перезагружается коэффициент деления (в дополнительном коде).

РЕГИСТРОМ УПРАВЛЕНИЯ является T2CON.

Прямой байтовый адрес T2CON:

dir – 0C8h.

	7	6	5	4	3	2	1	0
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#

Таблица 18 – Функции битов регистра T2CON

Имя бита	Функция
TF2 T2CON.7	Флаг переполнения Таймера 2. Устанавливается при переходе счетчиков из состояния все единицы (0FFFFh) в состояние все нули. Должен очищаться программно. Флаг TF2 не устанавливается, если RCLK = 1 или TCLK = 1.
EXF2 T2CON.6	Флаг внешнего события Таймера 2. Устанавливается по спадающему фронту на входе T2EX, если EXEN2 = 1. Является запросом прерывания от Таймера 2.
RCLK T2CON.5	Бит выбора источника синхрос частоты для приемника последовательного порта в его режимах 1 и 3. При RCLK = 1 источником является Таймер 2, при RCLK = 0 источником является Таймер 1.
TCLK T2CON.4	Бит выбора источника синхрос частоты для передатчика последовательного порта в его режимах 1 и 3. При TCLK = 1 источником является Таймер 2, при TCLK = 0 источником является Таймер 1.
EXEN2 T2CON.3	Бит разрешения внешнего события Таймера 2.
TR2 T2CON.2	Бит программного запуска/останова Таймера 2. При TR2 = 1 таймер запускается.
C/T2# T2CON.1	Бит выбора типа событий для Таймера 2. При C/T2# = 1 он работает как счетчик внешних событий, при C/T2# = 0 как таймер.
CP/RL2# T2CON.0	Бит выбора режима работы Таймера 2. При CP/RL2# = 1 по спадающему фронту на входе T2EX (если EXEN2 = 1) он переходит в режим захвата. При CP/RL2# = 0 по спадающему фронту на входе T2EX (если EXEN2 = 1) или по переполнению Таймера 2 он переходит в режим автоперезагрузки. Если RCLK = 1 или TCLK = 1, этот бит игнорируется, а Таймер 2 работает в режиме перезагрузки по переполнению.

Таймер 2 имеет два флага запроса прерывания: TF2 и EXF2. Оба запроса объединяются логической операцией «ИЛИ» (см. рис. 9 и рис. 10) и обслуживаются с использованием одного вектора прерывания 2Bh. Идентификацию конкретного запроса производит процедура обслуживания прерывания. Флаг TF2 устанавливается при переполнении счетного регистра Таймера 2 (при RCLK = TCLK = 0), а флаг EXF2 устанавливается по спадающему фронту на входе T2EX, если EXEN2 = 1.

СИСТЕМА ПРЕРЫВАНИЙ включает шесть источников запросов – к исходной системе микроконтроллеров 80C51 добавлено прерывание от Таймера 2, которое имеет два флага TF2 и EXF2 с единственным вектором 002Bh (с этого адреса должна начинаться подпрограмма обслуживания прерывания).

Система прерываний микроконтроллеров 8052 обслуживается регистрами IE и IP, в которые включены дополнительные биты: ET2 – разрешения прерыванием от Таймера 2 и PT2 – приоритет прерывания Таймера 2.

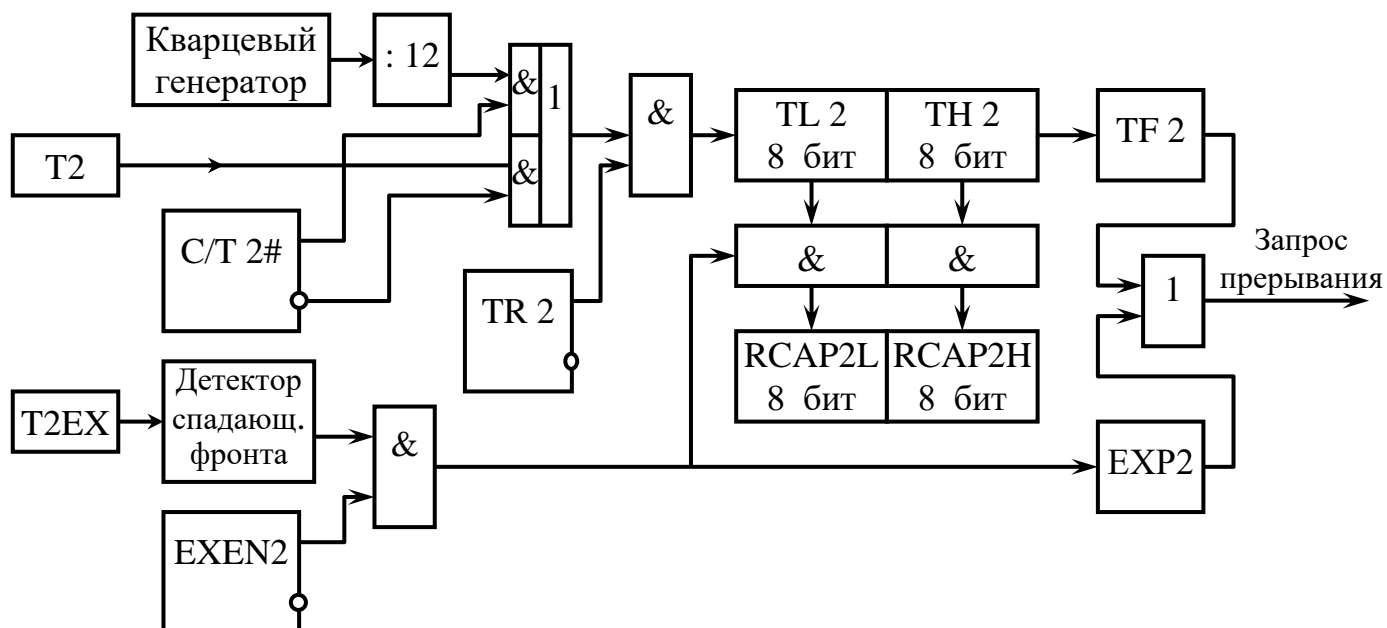


Рис. 9 – Логика работы Таймера 2 в режиме захвата

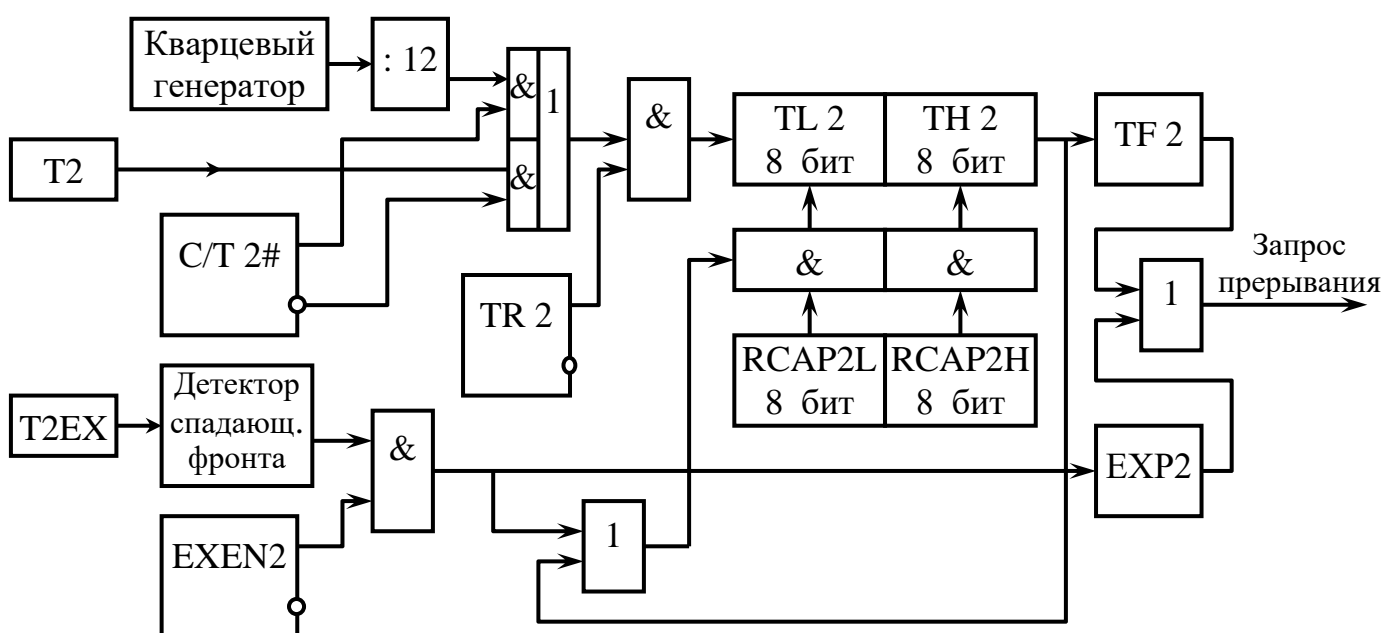


Рис. 10 – Логика работы Таймера 2 в режиме автоперезагрузки

РЕГИСТР МАСКИ (РАЗРЕШЕНИЯ) ПРЕРЫВАНИЙ 8052 – IE

Прямой байтовый адрес IE:

dir – 0A8h.

Допускается адресация отдельных бит IE :

bit – 0A8h_0AFh.

Адрес : bit	7	6	5	4	3	2	1	0
IE	0AFh		0ADh	0ACh	0ABh	0AAh	0A9h	0A8h
	EA		ET2	ES	ET 1	EX 1	ET 0	EX 0

РЕГИСТР ПРИОРИТЕТОВ ПРЕРЫВАНИЙ 8052 – IP

Прямой байтовый адрес IP: dir – 0B8h.

Допускается адресация отдельных бит IP : bit – 0B8h_0BFh.

	7	6	5	4	3	2	1	0
Адрес : bit			0BDh	0BCh	0BBh	0BAh	0B9h	0B8h
IP			PT2	PS	PT 1	PX 1	PT 0	PX 0

6 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ МИКРОКОНТРОЛЛЕРОВ ТИПА 8xC52 / 54 / 58

В этих микроконтроллерах введены два режима пониженного энергопотребления:

Режим холостого хода (Idle) – в этом режиме центральный процессор отключается, а система прерываний, счетчики-таймеры и другие блоки ввода-вывода продолжают функционировать. Счетчик команд, регистры и РПД сохраняют свои значения. Потребляемая мощность в режиме холостого хода составляет 15÷30 % от номинальной.

Одним из двух возможных способов выхода из режима холостого хода является формирование любого разрешенного запроса прерывания. Другим способом выхода из состояния холостого хода является подача активного сигнала на вход RESET.

Режим микропотребления (Power down) – в этом режиме приостанавливается выполнение всех функций микроконтроллера, поскольку прекращает работать синхрогенератор. Состояние РПД сохраняется, а содержимое регистров специальных функций теряется. Выход из состояния микропотребления может быть осуществлен только подачей активного сигнала на вход RESET.

Начиная с микроконтроллеров этой линии **Таймер 2** в режиме автоперезагрузки способен считать как в прямом, так и в обратном направлении. Кроме того, он может использоваться для формирования внешнего сигнала программируемой частоты на выводе порта P1.0.

Для управления Таймером 2 введен дополнительный регистр управления T2MOD.

В структуру микроконтроллеров 8xC54/58 дополнительно введен **второй регистр приоритетов прерываний IPH**, работающий совместно с регистром IP. Это увеличивает число уровней приоритетов для каждого прерывания до четырех.

7 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ МИКРОКОНТРОЛЛЕРОВ ТИПА 8xC51FA / 51FB / 51FC

Основным отличием микроконтроллеров 8xC51Fх от предыдущих чипов семейства MCS-51 является наличие **блока PCA (programmable counter array)**. Это блок ввода-вывода, предназначенный для выполнения различных операций счета и определения временных интервалов, в том числе при широтно-импульсной модуляции.

Блок PCA состоит из 16-ти разрядного таймера-счетчика (регистры CL и CH) и пяти 16-ти разрядных модулей фиксации-сравнения. Таймер-счетчик является источником временной базы и счетчиком событий, значения его текущего счета передаются в пять модулей фиксации-сравнения, реализованных на пяти парах регистров CCAPxL-CCAPxH.

Управление работой и режимами таймера-счетчика осуществляется при помощи регистра режима – CMOD и регистра управления – CCON. Режимы работы модулей сравнения-захвата определяются пятью регистрами CCAPMх.

Линии порта P1 имеют альтернативные функции:

- P1.2 – вход таймера-счетчика внешних событий;
- P1.3 ... P1.7 – входы при фиксации, выходы при сравнении и ШИМ для пяти модулей.

Модуль 4 блока PCA может быть запрограммирован на выполнение функции 16-ти разрядного **сторожевого таймера (watchdog timer – WDT)**. В этом режиме при совпадении числа в таймере-счетчике с величиной, занесенной предварительно в регистры данных модуля, осуществляется сброс и инициализация микроконтроллера.

У микроконтроллеров 8xC51Fх имеется семь источников прерываний (добавлено прерывание от блока PCA). Система приоритетов прерываний осталась четырехуровневой. Вектор прерываний от блока PCA равен 0033h.

8 МИКРОКОНТРОЛЛЕРЫ ТИПА 8xC51GB

Эта группа объединяет наиболее функционально развитые микроконтроллеры семейства MCS-51. По сравнению с младшими моделями увеличилось число параллельных портов ввода-вывода до шести, количество внешних входных сигналов прерываний увеличилось до восьми, введен второй блок PCA и аппаратно реализован сторожевой таймер. Принципиальным нововведением является включение 8-ми разрядного аналого-цифрового преобразователя.

Расширение набора выполняемых функций потребовало применить корпус с увеличенным количеством выводов (68PLCC).

Набор функциональных возможностей микроконтроллеров типа 8xC51GB:

- 8-ми разрядное АЛУ;
- Внутренняя память программ – от 8 до 32 кБайт;
- Внутреннее ОЗУ данных (РПД) – 256 байт;
- 6 универсальных программируемых 8-ми разрядных параллельных портов;
- Три 16-ти разрядных программируемых счетчика-таймера:
- Дуплексный последовательный порт с расширенными возможностями;
- Дополнительный последовательный порт SEP;
- Два блока PCA;
- Блок сторожевого таймера;
- Система прерываний: 8 внешних источников, 15 векторов, 4 приоритета прерываний;
- 2 режима пониженного энергопотребления;
- 3 уровня защиты памяти программ;
- Режим отладки ONCE.

9 ПРАКТИЧЕСКИЕ ЗАДАНИЯ ПО ПРОГРАММИРОВАНИЮ MCS-51

ЗАДАНИЕ 1. Переслать содержимое нулевого Банка РОНов в ВПД, начиная с адреса 5000H :

```

                mov     PSW, #08h      ; Выбор 1-го Банка РОНов
                mov     R2, #8         ; Счетчик циклов → R2
                mov     DPTR, #5000h   ; Начальный адрес → DPTR
                mov     R0, #0         ; Начальный адрес Банка 0
met :           mov     A, @R0         ; Пересылка байта из РОНа
                movx    @DPTR, A       ;                      в ВПД
                inc     DPTR           ; Наращивание адреса ВПД
                inc     R0             ; Наращивание адреса РПД
                djnz    R2, met        ; Уменьшение счетчика и повтор

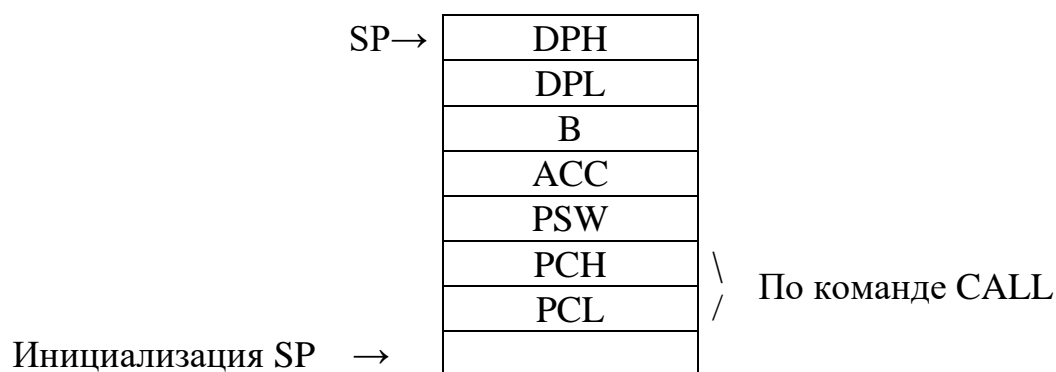
```

ЗАДАНИЕ 2. Программа обработки прерываний должна начинаться сохранением в стеке содержимого регистров : ACC, B, PSW, DPTR и переходом к Банку РОНов 1. Перед окончанием подпрограммы восстанавливаются исходные значения регистров :

```

SUBINT :  push     PSW      ; Сохранение в стеке PSW
          push     ACC      ; Сохранение в стеке Аккумуля.
          ; (Обратить внимание на прямую байтовую адресацию Акк.)
          push     B        ; Сохранение в стеке B
          push     DPL      ; Сохранение в стеке DPTR
          push     DPH      ;
          mov      PSW, #08h ; Выбор Банка РОНов 1
          * * * * *

```



```

          * * * * *
          pop      DPH      ; Восстановление DPTR
          pop      DPL      ;
          pop      B        ; Восстановление B
          pop      ACC      ; Восстановление Аккумуля.
          pop      PSW      ; Восстановление PSW
          reti

```

Возврат в нулевой Банк РОНов происходит при восстановлении из стека PSW.

ЗАДАНИЕ 3. Перевести однобайтовый шестнадцатеричный операнд в двоично-десятичный упакованный формат (BCD-код). Исходный операнд находится в R5. Результат поместить в R6 (число сотен) и в R5 (десятки, единицы) :

```

mov      A, R5      ; Пересылка операнда в Акк.
mov      B, #100    ; Делитель 100 → B
div      AB         ; Акк. содержит число сотен
mov      R6, A      ; Пересылка числа сотен → R6
mov      A, B       ; Пересылка остатка → Акк.
mov      B, #10     ; Делитель 10 → B
div      AB         ; Десятки → Акк., единицы → B
swap A              ; Пересылка десятков в старшую тетраду
add      A, B       ; Пересылка единиц → Акк.
mov      R5, A      ; Пересылка результата → R5

```

ЗАДАНИЕ 4. Перевести однобайтовый двоично-десятичный операнд (BCD-код) в шестнадцатеричный. Исходный операнд находится в R5. Результат поместить в Акк.:

```

mov      A, R5      ; Пересылка операнда → Акк.
anl      A, #0Fh    ; Выделение младшей тетрады
xch      A, R5      ; Младшую тетраду → R5
anl      A, #0F0h   ; Выделение старшей тетрады
swap     A          ; Перестановка тетрад
mov      B, #10     ; Множитель 10 → B
mul      AB         ; Десятки исходного числа → A
add      A, R5      ; Результат → Акк.

```

ЗАДАНИЕ 5. Рассчитать 5 значений функции $Y = 7 * x^2 + 15$. (x – изменяется от 3 с шагом 2). Результат разместить в РПД, начиная с адреса 40h. В память последовательно поместить младший байт результата, затем – старший байт.

```

mov      R0, #40h   ; Начальный адрес массива → R0
mov      R2, #5     ; Счетчик циклов → R2
mov      R3, #3     ; Начальное значение x → R3
met:     mov      A, R3 ; Перемещение x → A
         mov      B, R3 ; Перемещение x → B

```


mul	AB	; $x^2 \rightarrow A$
mov	B,#7	; Записать 7 $\rightarrow B$
mul	AB	; $7 * x^2 \rightarrow AB$
add	A,#15	; Младший байт результата $\rightarrow A$
mov	@R0,A	; Переслать младший байт в память
inc	R0	; Увеличение адреса памяти
mov	A,B	; Пересылка старшего байта $\rightarrow A$
addc	A,#0	; Добавление к старшему байту Carry
mov	@R0,A	; Переслать старший байт в память
inc	R0	; Увеличение адреса памяти
inc	R3	; Наращивание x
inc	R3	
djnz	R2,met	; Конец цикла
nop		

Команда *nop* в конце программы нужна только для отладки, потому что на эту команду устанавливается «точка останова».

ЗАДАНИЕ 6. Найти целое значение аргумента, при котором значение функции $Y = 45 * x + 21$ превысит 2048.

	mov	R2,#0	; Начальное значение x $\rightarrow R2$
met:	inc	R2	; Наращивание значения x
	mov	A,#45	; Записать 45 $\rightarrow A$
	mov	B,R2	; Переслать x $\rightarrow B$
	mul	AB	; $45 * x \rightarrow AB$
	add	A,#21	; Младший байт результата $\rightarrow A$
	mov	A,B	; Пересылка старшего байта $\rightarrow A$
	addc	A,#0	; Добавление к старшему байту Carry
	cjne	A,#8,met	; Сравнение старшего байта с 8
	nop		

Число 2048 представляется в двоичной системе как 1000 00000000B или в шестнадцатеричной системе – 08 00h. Поэтому вычисления заканчиваются, если старший байт функции достигнет значения 8

ЗАДАНИЕ 7. Выдать содержимое Аккумулятора в последовательном коде (младшим битом вперед) через младший разряд Порты P1

	mov	R2, #8	; Счетчик бит $\rightarrow R2$
met :	rrc	A	; Сдвиг Акк. через флаг C
	mov	P1.0, C	; Передача бита в Порт P1
	call	DELAY	; Вызов подпрограммы задержки
	djnz	R2, met	; Уменьшение счетчика и повтор

ЗАДАНИЕ 8. Выдать Содержимое Аккумулятора в коде «Манчестер 2» (младшим битом вперед) через нулевой разряд Порта P2. Каждый бит передается двумя интервалами : первый интервал содержит прямое значение, второй – инверсию бита :

```

man :      mov      R2, #8      ; Счетчик бит → R2
           rrc      A          ; Сдвиг Акк. через флаг C
           mov      P2.0, C     ; Передача прямого значения
           cpl      C          ; Инверсия бита
           nop      ; Выравнивание длительности
           nop      ; интервалов
           mov      P2.0, C     ; Передача инверсии бита
           djnz     R2, man     ; Уменьшение счетчика и повтор

```

Студентам предлагается самостоятельно рассчитать время передачи одного бита информации в ЗАДАНИИ 7 и в ЗАДАНИИ 8.

ЗАДАНИЕ 9. Вычислить Булеву функцию трех переменных :

$$Y = X \& \sim Z \vee W \& (X \vee Z).$$

Переменные X, Z, W поступают на линии Порта P1[0], P1[1], P1[2] соответственно. Результат Y вывести на линию P1[3] :

```

Y   bit   P1.3      ; Спецификация битов Порта P1
X   bit   P1.0
Z   bit   P1.1
W   bit   P1.2

      orl      P1, 07h    ; Настройка младших битов P1 на ввод
      mov      C, X       ; Ввод X
      anl      C, /Z      ; C → X & NOT (Z)
      mov      F0, C      ; Пересылка результата → F0
      mov      C, X       ; Ввод X
      orl      C, Z       ; C ← X ∨ Z
      anl      C, W       ; C ← W & (X ∨ Z)
      orl      C, F0      ; Результат → C
      mov      Y, C       ; Вывод результата → P1[3]

```

ЗАДАНИЕ 10. Подсчитать в регистре R2 количество единичных битов, поступивших в порт P1.

```

      mov      A, P1      ; пересылка операнда в Аккумулятор
      mov      R2, #0     ; очистка счетчика единичных битов

```

```

m1:    mov     R3, #8           ; счетчик количества повторений
        rrc     A              ; сдвиг младшего бита во флаг CARY
        jnc     m2             ; переход, если флаг CARY – пустой
        inc     R2             ; наращивание счетчика единичных битов
m2:    djnz    R3, m1          ; окончание цикла
        nop

```

ЗАДАНИЕ 11. Заменить содержимое младшего байта DPTR произведением его младшей и старшей тетрад.

```

        mov     A, DPL         ; загрузка младшего байта DPTR в Acc
        mov     B, A           ; копирование операнда в регистр B
        anl     B, #0Fh        ; выделение младшей тетрады в B
        anl     A, #0F0h       ; выделение старшей тетрады в A
        swap    A              ; обмен тетрад в Acc
        mul     AB             ; умножение тетрад
        mov     DPL, A         ; пересылка результата в младший байт
                               ; DPTR

```

ЛАБОРАТОРНАЯ РАБОТА ПО ПРОГРАММИРОВАНИЮ MCS-51

1 ЦЕЛЬ РАБОТЫ

- углубить и закрепить знания по архитектуре микроконтроллера I8051 (K1816BE51) и навыки по его программированию;
- научиться работать с программой-имитатором микроконтроллера I8051 (K1816BE51);
- приобрести практические навыки в составлении, отладке и выполнении программ, написанных на языке ассемблера для программирования микроконтроллера I8051 (K1816BE51).

2 САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ

Перед выполнением лабораторной работы студентам необходимо изучить программную модель и систему команд языка ассемблера микроконтроллера I8051 (K1816BE51).

Изучить основные сведения о работе программы-имитатора «**Raiso-nance 8051**» микроконтроллера I8051 (K1816BE51), функциональные возможности и режимы работы программы-имитатора.

В соответствии с вариантами заданий составить программы на языке Ассемблера MCS-51.

3 ОСНОВНЫЕ СВЕДЕНИЯ О РАБОТЕ С ПРОГРАММОЙ «MC Studio»

Для создания проекта необходимо в меню «Файл» выбрать команду «Создать» и в предложенном перечне выбрать «Проект».



В открывшемся окне указать:


- имя проекта (например: lab_1);
- выбрать путь к проекту – указать папку (например: D\AK\LAB);
- выбрать модель контроллера (например: Intel – 80C51)
- и нажать кнопку «ОК».

В открывшемся окне ввести программу на языке Ассемблер.

Для компиляции проекта (перевода программы в машинные коды) необходимо нажать «CTRL + F9» или выбрать на панели инструментов

кнопку .

Для начала отладки программы необходимо нажать F9 или выбрать на панели инструментов кнопку . Остановка отладки – нажать «CTRL + F2» или кнопку на панели инструментов .

Для пошаговой отладки необходимо нажимать клавишу F7 или кнопку на панели инструментов .

4 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1 ИССЛЕДОВАТЬ ВЫЧИСЛЕНИЯ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ в микроконтроллере. Номер варианта выбирается в соответствии с последней цифрой номера зачетной книжки.

Вариант 1. Рассчитать значение функции $Y = 15x + 10$ (x изменяется в интервале от 5 до 20 с шагом 1). Результат разместить в РПД с адреса 40h (в массив последовательно занести сначала младший, а затем старший байт результата).

Вариант 2. Рассчитать значение функции $Y = 3X + 15$ (x изменяется в интервале от 10 до 100 с шагом 10). Результат разместить в РПД с адреса 30h (в массив последовательно занести сначала младший, а затем старший байт результата).

Вариант 3. Рассчитать значение функции $Y = 5X - 50$ (x изменяется в интервале от 0 до 20 с шагом 2). Результат разместить в РПД с адреса 30h.

Вариант 4. Составить программу вычитания четырехбайтовых беззнаковых чисел. Первое число находится в РПД по адресу $30 \div 33h$, второе – по адресу $38 \div 3Bh$. Результат поместить на место первого операнда.

Вариант 5. Массив чисел был архивирован и помещен в новый массив, в котором предыдущий элемент указывает число, а последующий – количество повторений этого числа в исходном массиве. В результирующем массиве описано 8 пар чисел. Найти сумму членов исходного массива. Результат разместить в регистрах R3, R4, R5.

Вариант 6. Рассчитать 16 значений функции $Y = 250/X$ (для X , начинающегося с 10 с шагом 8). Результаты округлить до целого значения и разместить в РПД с адреса 40h.

Вариант 7. Перевести однобайтовый шестнадцатеричный операнд в двоично-десятичный упакованный формат. Исходный операнд находится в регистре R5. Результат разместить в регистрах R4 (число сотен) и R3 (десятки, единицы).

Вариант 8. В РПД, начиная с адреса 30h находится массив из 20 элементов. Подсчитать количество элементов массива, попавших в интервал от 50 до 100. Результат запомнить в регистре R5.

Вариант 9. В РПД, начиная с адреса 30H, находится массив из 16 чисел. Найти максимальный элемент массива и поместить в R2 его значение, а в R3 его адрес.

Вариант 10. В регистре R5 находится двоично-десятичный операнд. Перевести операнд в шестнадцатеричное значение и поместить в R5

4.2 ИССЛЕДОВАТЬ ВЫПОЛНЕНИЕ ЛОГИЧЕСКИХ ФУНКЦИЙ, оперирующих с битовыми данными. Составить в соответствии с вариантом задания программу, реализующую Булеву функцию четырех переменных. Исходными значениями являются:

- A – 3-й бит аккумулятора;
- B – 5-й бит ячейки РПД по адресу 30H;
- C – 7-й бит порта P0;
- D – флаг переноса.

Вариант 1. Вычислить значение логической функции $(A \vee B) \& (C \vee D)$.

Вариант 2. Вычислить значение логической функции $(A \& C) \vee (B \& D)$.

Вариант 3. Вычислить значение логической функции $A \& (B \vee C) \& D$.

Вариант 4. Вычислить значение логической функции $(A \vee C \vee D) \& B$.

Вариант 5. Вычислить значение логической функции $((A \& D) \vee C) \& B$.

Вариант 6. Вычислить значение логической функции $(A \vee B) \vee (C \vee D)$.

Вариант 7. Вычислить значение логической функции $A \& (B \vee C \vee D)$.

Вариант 8. Вычислить значение логической функции $A \& B \vee C \vee D$.

Вариант 9. Вычислить значение логической функции $A \vee B \& C \vee D$.

Вариант 10. Вычислить значение логической функции $A \& B \& C \& D$.

4.3 ИССЛЕДОВАТЬ ПРИМЕНЕНИЕ КОМАНДЫ CJNE.

Вариант 1. В РПД, начиная с адреса 20h, находится массив из 16 элементов. Подсчитать и сохранить в регистрах: R2 – количество элементов массива, меньших значения 128; R3 – количество элементов массива, равных 128; R4 – количество элементов массива, больших 128.

Вариант 2. В РПД с адреса 20h находится массив, состоящий из 16 элементов. Суммировать элементы массива до тех пор, пока значение суммы не превысит 512. Выдать в R3 номер элемента, на котором произошло переполнение. Если сумма элементов не достигла значения 512, то выдать в регистре R3 значение 0.

Вариант 3. Для функции $Y=20X+45$ выдать в R2 первое значение аргумента, при котором значение функции превысит 1024. Начальное значение аргумента $X=10$.

Вариант 4. В РПД с адреса 20h находится массив из 16 чисел. Элементами массива являются числа 32, 64, 96 и 128. Подсчитать и сохранить в регистрах $R4 \div R7$ количество повторений каждого элемента.

Вариант 5. В РПД по адресам $20h \div 2Fh$ находится массив. С адреса $30h$ создать массив, в который входят адреса элементов первого массива, равных 128. В регистре R2 сохранить число элементов, равных 128. Прервать выполнение программы, если будет найдено 5 элементов со значением 128.

Вариант 6. В РПД с адресов $20h$ и $30h$ находятся 2 массива, состоящие из 16 элементов каждый. Подсчитать количество элементов первого массива, которые имеют равные значения во 2 массиве. Результат занести в регистр R2.

Вариант 7. Для функции $Y=40X+10$ получить первое значение, превышающее 512, начиная с $X=1$. Значение аргумента записать в R4, функции - в R5, R6.

Вариант 8. В ВПД, начиная с адреса $100h$, находится массив из 10 элементов. Получить в регистре R3 число элементов, равных $55h$. Счет прервать, если число элементов превысит 3.

Вариант 9. Для функции $15X+85$ найти первое значение аргумента, при котором младший байт функции равен 155.

Вариант 10. В ВПД с адреса $300h$ находится массив из 15 чисел. Элементами массива являются числа 10, 20, 30 и 180. Подсчитать и сохранить в регистрах R4 - R7 количество повторений каждого элемента.

4.4 ИССЛЕДОВАТЬ ПРИМЕНЕНИЕ ТАБЛИЧНЫХ ДАННЫХ при программировании микроконтроллера.

Вариант 1. Создать в ВПД, начиная с адреса $40h$, массив из 10 чисел, элементами которого являются квадраты чисел (от 0 до 15), прочитанных из порта P1. Таблица квадратов чисел расположена в ПЗУ, начиная с адреса $100h$.

Вариант 2. Для кодирования информации используется алгоритм перестановки, суть которого состоит в том, что символы исходного алфавита заменяются на те же символы, но в другом порядке. Соответствие символов исходного алфавита – кодированному приведено в таблице 4.1. Для определения конца сообщения используется символ «*» с кодом $2Ah$.

В РПД с адреса $30h$ находится сообщение неизвестной длины (не более 20 символов). Закодировать сообщение в соответствии с таблицей 4.1 и разместить в РПД с адреса $50h$.

Таблица 4.1 – Соответствие символов исходного алфавита символам закодированного алфавита.

Исходный алфавит	Символ	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
	Код (h)	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA
Закодиров. алфавит	Символ	З	О	Х	Ы	Я	И	А	П	Ц	Б	Й
	Код (h)	C7	CE	D5	DB	DF	C8	C0	CF	D6	DC	C9

Исходный алфавит	Символ	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
	Код (h)	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5
Закодиров. алфавит	Символ	В	Р	Ч	Э	_	К	Г	С	Ш	Ю	Л
	Код (h)	C2	D0	D7	DD	5F	CA	C3	D1	D8	DE	CB

Исходный алфавит	Символ	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	_
	Код (h)	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	5F
Закодиров. алфавит	Символ	Д	Т	Щ	М	Е	Н	У	Ъ	Ж	Б	Ф
	Код (h)	C4	D2	D9	CC	C5	CD	D3	DA	C6	C1	D4

Вариант 3. Для кодирования информации используется алгоритм перестановки, суть которого состоит в том, что символы исходного алфавита заменяются на те же символы, но в другом порядке. Соответствие символов исходного алфавита – закодированному приведено в таблице 4.1. Для определения конца сообщения используется символ «*» с кодом 2Ah.

В РПД с адреса 20h находится закодированное сообщение длиной 29 символов. Текст сообщения:

«ГЭЭОМИЧЦИФКЗГЙЭЯЦКЭХЗЧЭФХИКЧЭ*».

Раскодировать сообщение в соответствии с таблицей 4.1 и разместить текст раскодированного сообщения в РПД с адреса 50h.

Вариант 4. Зависимость $Y(X)$ представлена в таблице 4.2 (значения заданы в десятичном виде).

Таблица 4.2 – Таблица зависимости $Y(X)$.

X	00	01	02	03	04	05	06	07	08	09
Y	0	1	2	4	8	16	32	64	128	240

Составить программу нахождения аргумента по значению функции. Значение функции (Y) находится в регистре R7. Значение аргумента (X) разместить в регистре R0. Если значение (Y) отсутствует в таблице, то выдать в регистре R0 значение 0Fh.

Вариант 5. Зависимость функции от двух аргументов X и Y задана в табличном виде (таблица 4.3). Значения в таблице даны в десятичном виде.

Таблица 4.3 – Зависимость функции от аргументов Y и X.

		Y									
		00	01	02	03	04	05	06	07	08	09
X	0	00	01	02	03	04	05	06	07	08	09
	1	32	33	34	35	36	37	38	39	40	41
	2	64	65	66	67	68	69	70	71	72	73

Составить программу для выдачи в порт P2 значений функции в зависимости от аргументов, которыми являются входные сигналы портов P0 (значение X) и P1 (значение Y).

Вариант 6. В зависимости от осведомительных сигналов, поступающих в порт P0, в порт P1 выдаются управляющие сигналы. При значении порта P0, меньшем 100, в порт P1 выдается 00, а при значении порта P0, большем 113 – 255. Зависимость выходного значения порта P1 от входного значения порта P0 в интервале 100-113 определяется графиком, представленным на рис. 4.1.

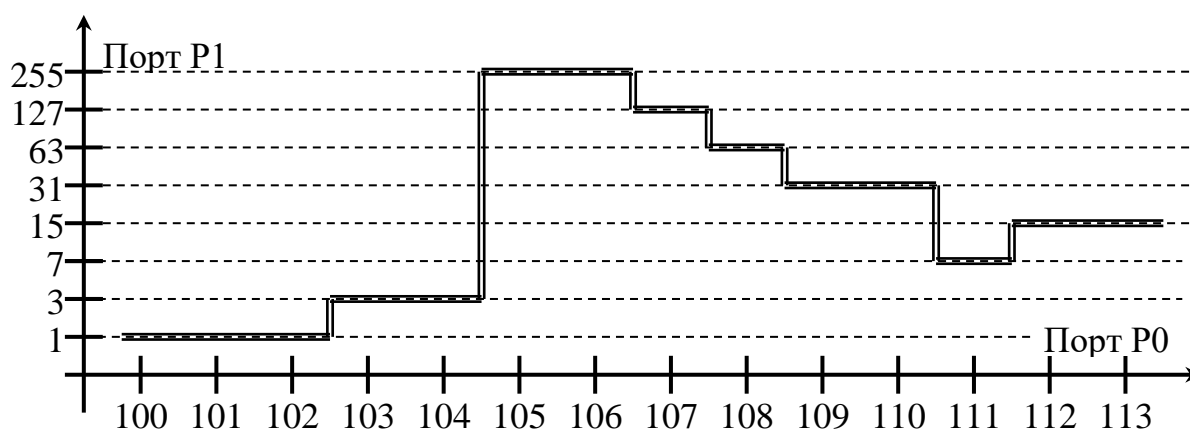


Рис. 4.1 – Зависимость значения порта P1 от порта P0.

Вариант 7. Зависимость значения функции от значения аргумента представлена в таблице 4.4 в десятичном виде. Составить программу, выдающую в порт P2 значение X в зависимости от поступившего в порт P0 значения Y. Если значение Y отсутствует в таблице, то выдать в порт P0 значение FFh.

Таблица 4.4 Таблица зависимости Y(X).

X	00	01	02	03	04	05	06	07	08	09	10
Y	10	25	40	57	93	99	102	115	125	149	157

Вариант 8. Зависимость функции от двух аргументов X и Y представлена в табличном виде (таблица 4.5). Значения даны в десятичной системе счисления.

Составить программу, которая в зависимости от значения функции, поступившего в порт P0, выдает в порты P1 и P2 значения X и Y.

Таблица 4.5 – Зависимость функции от аргументов Y и X.

		Y									
		00	01	02	03	04	05	06	07	08	09
X	10	20	25	30	35	40	45	50	55	60	65
	20	70	72	74	76	78	80	82	84	86	88
	30	90	91	92	93	94	95	96	97	98	99

Вариант 9. Зависимость значения функции от значения аргумента представлены в таблице 4.6 в десятичном виде.

В ВПД, начиная с адреса 10h, располагается массив из 10 элементов. Разместить с адреса 20h массив, элементы которого являются функциями от элементов первого массива в соответствии с таблицей 4.6. Если значение X отсутствует в таблице, записать вместо Y – значение 255.

Таблица 4.6 Таблица зависимости Y(X).

X	00	10	20	30	40	50	60	70	80	90	100
Y	28	35	45	15	69	78	55	39	98	129	168

Вариант 10. Составить программу, выдающую в порт P1 значение в зависимости от входного значения порта P2.

Зависимость выходных значений от входных представлена графиком (рис.4.2) в десятичной системе счисления. При превышении допустимого входного значения в порт выдается значение 00.

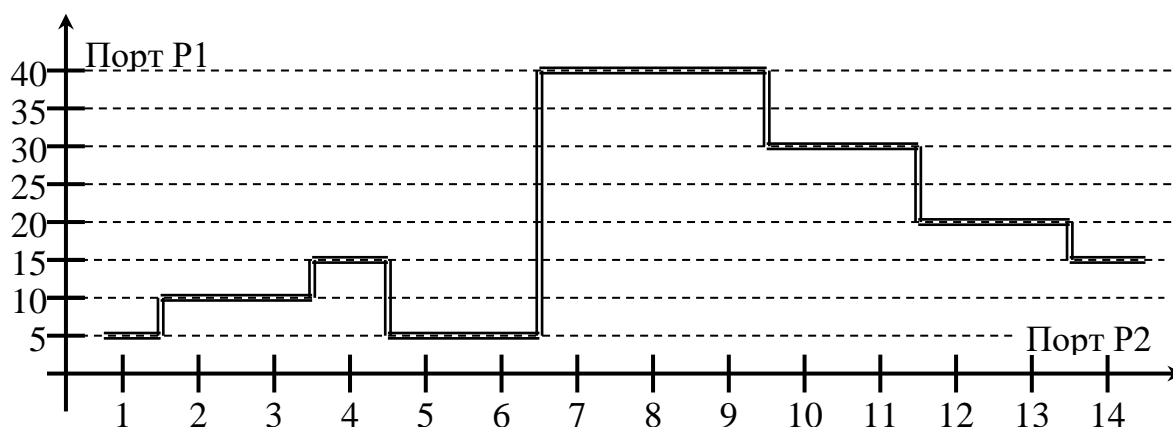


Рис.4.2 – Зависимость значения порта P1 от порта P0

4.5 ИССЛЕДОВАТЬ ПРИМЕНЕНИЕ ПОДПРОГРАММ. В соответствии с вариантом разработать программу с использованием подпрограммы.

Вариант 1. В порты микроконтроллера P0-P3 поступают двоично-десятичные данные. Перевести данные в шестнадцатеричный формат и разместить в РПД последовательно с адреса 30h.

Вариант 2. Выдать последовательно в порты P1 и P2 микроконтроллера содержимое младших байт счетчиков в двоично-десятичном формате (в P1 – сотни, в P2 – десятки и единицы).

Вариант 3. В порты P0÷P3 поступают шестнадцатеричные данные. Занести в РПД, начиная с адреса 40h количество единиц, поступивших в каждый порт.

Вариант 4. Для каждого из регистров R0, R3 и регистра-расширителя В последовательно выдать в порты информацию о содержимом регистров:

- P0 - прямое значение байта;
- P1 - инверсное значение байта;
- P2 - количество нулей в байте;
- P3.0 - флаг контроля на четность.

Вариант 5. В каждый из портов P0÷P2 поступают данные от двух четырехразрядных датчиков. Выдать в порт P3 сумму шести датчиков, подключенных к портам P0÷P2.

Вариант 6. Записать в регистры R3, R7 и регистр-расширитель В произведение их старшей и младшей тетрады соответственно.

Вариант 7. Выдать в порты P0÷P2 количество единиц, содержащихся в регистрах R0,R7 и регистре-расширителе В соответственно.

Вариант 8. Считать с интервалом в 10 миллисекунд 5 значений из порта P0, используя подпрограмму задержки на 10 миллисекунд. Входные коды записать в РПД, начиная с адреса 40h

Вариант 9. На вход внешнего прерывания \sim INT0 микроконтроллера через схему «8ИЛИ-НЕ» подключено 8 источников прерываний ИСТ0-ИСТ7. Эти же источники подключены к выводам порта P2. Составить подпрограмму обслуживания прерывания, которая определяет номер источника прерывания и переходит по соответствующему адресу. В случае одновременного поступления нескольких запросов вступает в действие система приоритетов, которая имеет вид в порядке убывания: ИСТ3, ИСТ5, ИСТ7, ИСТ4,

ИСТ0, ИСТ1, ИСТ2, ИСТ4. Адреса начала подпрограмм установить самостоятельно.

Вариант 10. Составить подпрограмму, которая по сигналу внешнего прерывания $\sim INT1$ выдает в порты P0÷P2 текущую сумму таймеров-счетчиков.

5 КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Объяснить порядок выполнения работы и полученные результаты.
2. Отличительные особенности контроллеров семейства MCS-51.
3. Структурная организация микроконтроллеров семейства MCS-51.
4. Распределение памяти данных MCS-51.
5. Распределение памяти программ MCS-51.
6. Флаги MCS-51. Слово состояния процессора MCS-51.
7. Команды, модифицирующие флаги.
8. Система команд MCS-51. Типы команд.
9. Методы адресации MCS-51.
10. Команды пересылки обмена и загрузки MCS-51.
11. Арифметические команды MCS-51.
12. Логические команды MCS-51.
13. Команды, оперирующие с битами, в системе команд MCS-51.
14. Команды условных переходов MCS-51.
15. Команды безусловных переходов MCS-51.
16. Команды инкремента и декремента.
17. Таймеры/Счетчики MCS-51. Программирование таймеров/счетчиков.
18. Режимы работы 0 и 1 таймеров/счетчиков MCS-51.
19. Режимы работы 2 и 3 таймеров/счетчиков MCS-51.
20. Режимы прерываний MCS-51. Программирование режимов прерываний.
21. Приоритеты прерываний MCS-51.
22. Последовательный Порт MCS-51. Программирование последовательного порта.
23. Режимы работы последовательного порта MCS-51.
24. Программирование таблиц.
25. Расширение памяти программ и памяти данных микроконтроллера.

ОСНОВНАЯ ЛИТЕРАТУРА ПО КУРСУ

1. Сташин В.В. и др. Проектирование цифровых устройств на однокристальных микроконтроллерах М.: Энергоатомиздат, 1990
2. Однокристальные микроЭВМ. Справочник. - М.: МИКАП, 1994
3. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс.- М.: Издательство ЭКОМ, 1999.- 400с.