Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики им. А.Н. Тихонова

Департамент компьютерной инженерии

«Разработка и отладка устройств на базе микроконтроллеров семейства MCS-51».

Методическое руководство к лабораторному практикуму.

МОСКВА, 2019

УДК 681.3.01(031)

Ю.И. Гудков, С.Н. Сафонов, А.Л. Тув, «Разработка и отладка устройств на базе микроконтроллеров семейства MCS-51»: Методическое руководство к лабораторному практикуму. М.: МИЭМ НИУ ВШЭ, 2019.

Методическое руководство предназначено для студентов направления 09.03.01. «Информатика и вычислительная техника». Лабораторный практикум ориентирован на практическое освоение программных и аппаратных ресурсов «классического» микроконтроллера семейства MCS-51, способов реализации типовых функций ввода/вывода и обработки информации и может быть полезен при изучении курсов «Микроконтроллерные системы», «Интерфейсы робототехнических систем», «Встроенные и распределенные системы контроля и управления», выполнения прикладных проектов и выпускных квалификационных работ.

Введение.

Однокристальный микроконтроллер является составной частью практически любого современного цифрового устройства. Интеграция на кристалле высокопроизводительного процессора, памяти и периферийных модулей ввода/вывода позволяет с минимальными затратами решить широкий круг задач проектирования функциональных модулей и приборов для бытовой техники, промышленной автоматики и других специализированных систем обработки информации.

В настоящее время микроконтроллеры выпускаются многими производителями интегральных схем и являются массовым, относительно недорогим и доступным изделием. Наибольшее распространение получила продукция таких фирм, как Intel, Microchip, Atmel, Dallas Semiconductor, Philips Semiconductor, Motorola, которые поставляют на рынок почти 90% микроконтроллеров с различным набором программно-аппаратных ресурсов, числом выводов, диапазоном питающих напряжений и токов потребления.

Современной тенденцией развития микроконтроллеров является переход к RISC-архитектуре ядра, которая позволяет при той же тактовой частоте, а следовательно и потребляемой мощности, добиться более высокой производительности по сравнению с CISC-архитектурой.

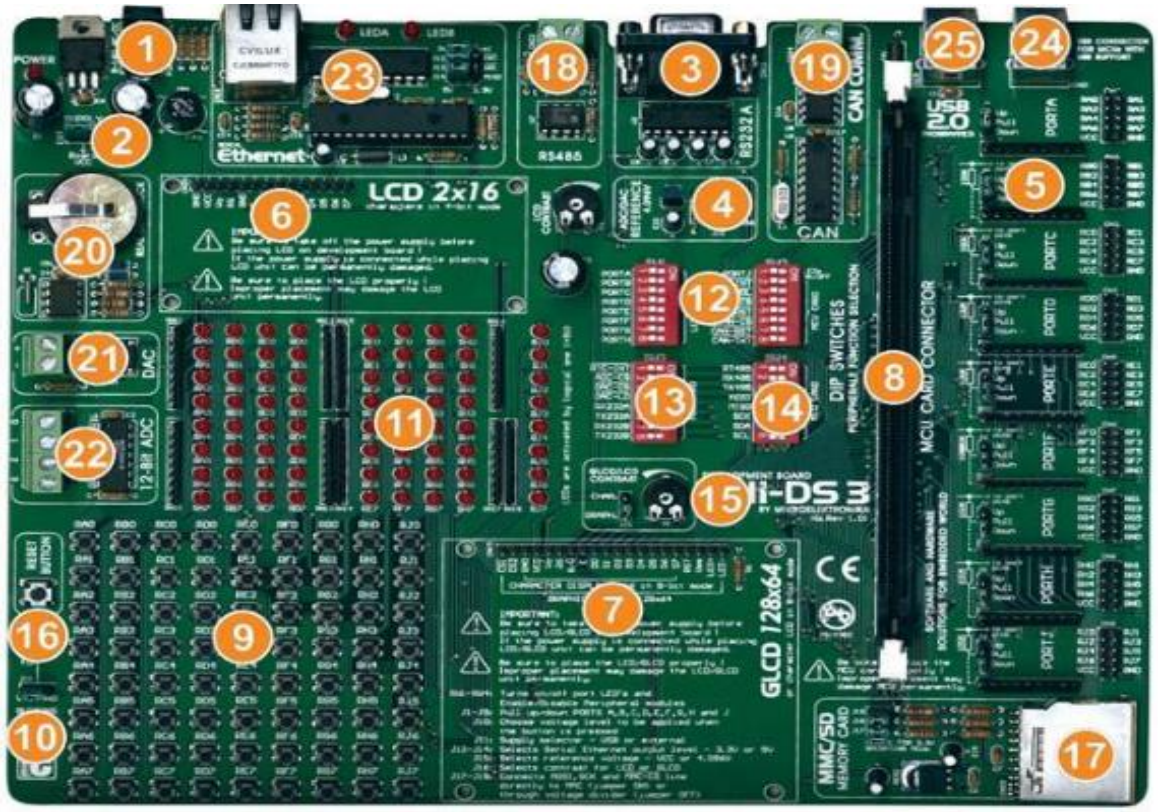
Однако, на рынке до сих пор востребованы микроконтроллеры семейства MCS-51, архитектура которых была запатентована фирмой Intel более 30 лет назад. На сегодняшний день данная архитектура является классической для микроконтроллеров с CISC-ядром, а предложенные подходы к организации адресного пространства памяти, составу аппаратных ресурсов, делению системы команд на функциональные группы, стали классическими и при разработке большинства современных микроконтроллеров с RISC и AVR архитектурой.

Микроконтроллеры семейства MCS-51 выпускались и выпускаются не только фирмой Intel, но и Atmel, Dallas Semiconductor, Philips Semiconductor и другими. Каждая фирма добавляет свою «изюминку» в технологию и резидентные ресурсы микроконтроллера, делая его привлекательным для применения в новых областях, при этом отрабатывая новые технические решения для реализации в своих «фирменных» разработках микроконтроллеров.

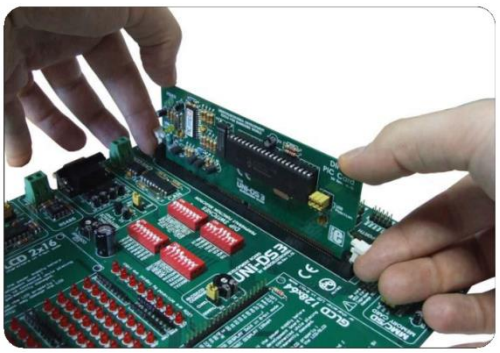
Решение любых вычислительных задач, а в особенности задач управления процессами требует эффективного использования программно-аппаратных средств микроконтроллера. Значительная часть задач, связанных с реализацией стандартных интерфейсных функций и функций управления внешними объектами в микроконтроллере может быть решена с помощью резидентных аппаратных ресурсов. Доступ к таким ресурсам наиболее рационально организовать на машинном уровне, что обеспечит максимальное быстродействие и минимальный объем памяти соответствующих программных модулей. В связи с этим лабораторный практикум предусматривает использование ассемблера в качестве основного языка программирования. Разработанные в ходе выполнения технического задания программные модули в дальнейшем могут использоваться как пользовательские библиотечные функции при решении более сложных задач проектирования специальных систем обработки информации.

Конфигурация универсального учебного стенда.

- Перед началом работы изучите [1] структуру и основные компоненты универсальной платы среды разработки UNI-DS3.

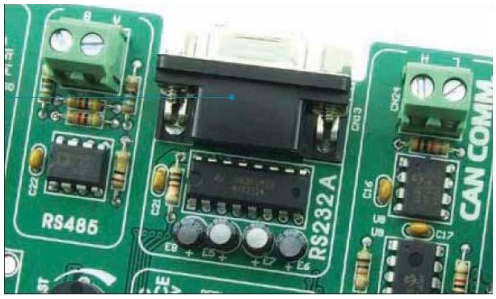


- Установите в **разъем № 8** платы UNI-DS3 плату целевого микроконтроллера семейства MCS-51 с установленным микроконтроллером AT89S8253. Все порты микроконтроллера соединены с контактами разъема, что позволяет их коммутировать с любыми модулям универсальной платы стенда и платами расширения.





- Соедините **разъем №3** (рис.1.4) платы UNI-DS3 стандартным удлинителем с COM-портом ПК. Установите секции 5,6 переключателя SW3 в положение ON, а секции 7,8 переключателя SW3 в положение OFF.

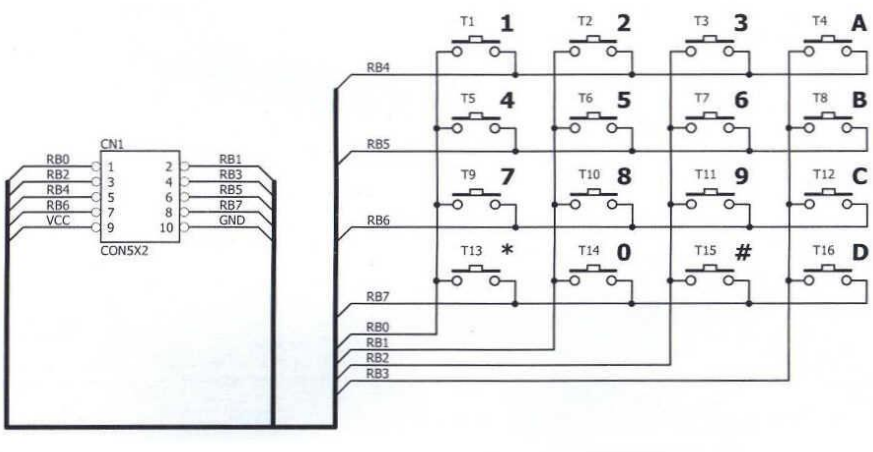


- Соедините **разъем №25** (рис.1.5) платы UNI-DS3 стандартным кабелем с USB-портом персонального компьютера и установите коммутационную перемычку J11 в положение «USB Power Supply».



- Подключите к **разъему №5** (CN2 - PORTB) плату матричной клавиатуры. Принципиальная схема матричной клавиатуры приведена ниже. Достоинством подобной коммутации кнопок является существенное уменьшение числа выводов микроконтроллера, необходимых для опроса клавиатуры данного типа. Максимальная экономия линий ввода/вывода достигается при подключении кнопок в узлы **«квадратной»** матрицы. Однако, в отличие от варианта подключения каждой кнопки непосредственно ко входу микроконтроллера, для опроса матричной клавиатуры требуется реализация специального алгоритма, который должен быть разработан в ходе выполнения лабораторного практикума.





- Задайте аппаратную конфигурацию микроконтроллерной системы согласно требованиям технического задания на лабораторный практикум с помощью коммутационных перемычек и переключателей платы UNI-DS3 используя таблицу соответствия обозначений портов микроконтроллера семейства MCS-51 и компонентов универсальной платы.

Компоненты платы UNI-DS3	**Порты (выводы) микроконтроллера**	**Элементы коммутации**
Светодиоды RA0…RA7	P0.0…P0.7	Подключение SW1:1
Светодиоды RB0…RB7	P1.0…P1.7	Подключение SW1:2
Светодиоды RC0…RC7	P2.0…P2.7	Подключение SW1:3
Светодиоды RD0…RD7	P3.0…P3.7	Подключение SW1:4
Кнопки RA0…RA7	P0.0…P0.7	J10 задает сигнал (GND или VCC), подаваемый на выводы портов при нажатии кнопки.
Кнопки RB0…RB7	P1.0…P1.7	
Кнопки RC0…RC7	P2.0…P2.7	
Кнопки RD0…RD7	P3.0…P3.7	
Модуль RS232	P3.0, P3.1, P3.2	SW3:5, SW3:6, SW3:1
Модуль Ethernet	P1.5…P1.7 P3.3…P3.6	SW4:4…SW4:6 SW2:2… SW2:5
Модуль CAN	P1.5…P1.7 P3.3, P3.5, P3.6	SW4:4…SW4:6 SW2:6, SW2:7, SW2:8
Модуль RTC	P3.2 (RTC-INT) P1.0 (SDA) P1.1 (SCL)	SW3:1 SW4:7 SW4:8
Модуль MMC/SD	P1.5…P1.7	SW4:4… SW4:6
Модуль RS485	P3.0, P3.1, P3.7	SW4:1, SW4:2, SW4:3
Модуль DAC	P1.2 (DAC-CS) P1.3 (DAC-LD) P1.5 (MOSI) P1.7 (SCK)	SW3:3 SW3:4 SW4:4 SW4:6
Модуль ADC	P1.5 (MOSI) P1.6 (MISO) P1.7 (SCK) P2.0 (ADC-CS)	SW4:4 SW4:5 SW4:6 SW3:2
Разъемы подключения плат расширения: CN1 (PORTA), CN2 (PORTB), CN3 (PORTC), CN4 (PORTD)	P0, P1, P2, P3	J1, J2, J3, J4 подключают/отключают «подтягивающие» резисторы к выводам портов P0,P1,P2,P3

Система команд микроконтроллеров семейства MCS-51.

Система команд содержит 111 базовых операций, которые по функциональному признаку могут быть разделены на 5 групп:

- Операции пересылки данных.
- Арифметические операции.
- Логические операции.
- Операции с отдельными битами.
- Операции передачи управления.

Команды имеют длину от одного до трех байт и выполняются за 1, 2 или 4 машинных цикла. Длительность машинного цикла составляет 12 / Fosc, где Fosc – частота кварцевого резонатора микроконтроллера. Первый байт любой команды содержит поле кода операции (КОП) и для некоторых команд поле параметра. Второй и третий байты команды – параметр (адрес операнда, адрес перехода или непосредственные данные).

Название команды	Мнемоника	Б	Ц	Операция
Операции пересылки данных				
Пересылка в аккумулятор из регистра	MOV A, Rn	1	1	(A) ← (Rn)
Пересылка в аккумулятор прямо-адресуемого байта	MOV A, ad	2	1	(A) ← (ad8)
Пересылка в аккумулятор косвенно-адресуемого байта	MOV A, @Ri	1	1	(A) ← ((Ri))
Загрузка в аккумулятор константы	MOV A, #d8	2	2	(A) ← #d8
Пересылка в регистр из аккумулятора	MOV Rn, A	1	1	(Rn) ← (A)
Пересылка в регистр прямо-адресуемого байта	MOV Rn, ad	2	2	(Rn) ← (ad8)
Загрузка в регистр константы.	MOV Rn, #d	2	1	(Rn) ← #d8
Пересылка аккумулятора по прямому адресу	MOV ad8, A	2	1	(ad8) ← (A)
Пересылка регистра по прямому адресу.	MOV ad8, Rn	2	2	(ad8) ← (Rn)
Пересылка прямо-адресуемого байта по прямому адресу	MOV ad8, ad8	3	2	(ad8) ← (ad8)
Пересылка косвенно-адресуемого байта по прямому адресу.	MOV ad8, @Ri	2	2	(ad8) ← ((Ri))
Пересылка по прямому адресу константы.	MOV ad8, #d8	3	2	(ad8) ← #d8
Пересылка аккумулятора в косвенно-адресуемый байт.	MOV @Ri, A	1	1	((Ri)) ← (A)
Пересылка прямо-адресуемого байта в косвенно-адресуемый байт.	MOV @Ri, ad8	2	2	((Ri)) ← (ad8)
Пересылка константы в косвенно-адресуемый байт.	MOV @Ri, #d8	2	1	((Ri)) ← #d8
Загрузка указателя данных.	MOV DPTR, #d16	3	2	(DPTR)←#d16
Пересылка в аккумулятор байта из программной памяти.	MOVC A, @A+DPTR	1	2	(A)←((A)+(DPTR))
Пересылка в аккумулятор байта из программной памяти.	MOVC A, @A+PC	1	2	(PC)←(PC)+1, (A)←((A)+(PC))
Пересылка в аккумулятор байта из внешней памяти данных.	MOVX A, @Ri	1	2	(A)←((Ri))
Пересылка в аккумулятор байта из внешней памяти данных.	MOVX A, @DPTR	1	2	(A)←((DPTR))
Пересылка во внешнюю память данных из аккумулятора.	MOVX @Ri, A	1	2	((Ri))←(A)
Пересылка во внешнюю память данных из аккумулятора.	MOVX @DPTR, A	1	2	((DPTR))←(A)
Загрузка в стек.	PUSH ad8	2	2	(SP)←(SP)+1, ((SP))←(ad)
Извлечение из стека.	POP ad8	2	2	(ad)←(SP), (SP)←(SP) - 1
Обмен аккумулятора с регистром.	XCH A, Rn	1	1	(A) ↔ (Rn)
Обмен аккумулятора и прямо-адресуемого байта.	XCH A, ad8	2	1	(A) ↔ (ad8)
Обмен аккумулятора и косвенно-адресуемого байта.	XCH A, @Ri	1	1	(A) ↔ ((Ri))
Обмен младших тетрад аккумулятора и косвенно-адресуемого байта	XCHD A, @Ri	1	1	$(A_{0\ldots3}) \leftrightarrow ((Ri)_{0\ldots3})$
Арифметические операции				

Сложение аккумулятора и регистра	ADD A, Rn	1	1	(A) ← (A) + (Rn)
Сложение аккумулятора и прямо-адресуемого байт	ADD A, ad8	2	1	(A) ← (A) + (ad8)
Сложение аккумулятора и косвенно-адресуемого байта	ADD A, @Ri	1	1	(A) ← (A) + ((Ri))
Сложение аккумулятора и константы	ADD A, #d8	2	1	(A) ← (A) + #d8
Сложение аккумулятора, регистра и переноса	ADDC A, Rn	1	1	(A)←(A)+(Rn)+(C)
Сложение аккумулятора, прямо-адресуемого байта и переноса	ADDC A, ad8	2	1	(A)←(A)+(ad8)+(C)
Сложение аккумулятора, косвенно-адресуемого байта и переноса	ADDC A, @Ri	1	1	(A)←(A)+((Ri))+(C)
Сложение аккумулятора, константы и переноса	ADDC A, #d8	2	1	(A)←(A)+ #d8+(C)
Десятичная коррекция аккумулятора	DA A	1	1	
Вычитание из аккумулятора регистра и переноса.	SUBB A, Rn	1	1	(A)←(A)-(C)-(Rn)
Вычитание из аккумулятора прямо-адресуемого байта и переноса	SUBB A, ad8	2	1	(A)←(A)-(C)-((ad))
Вычитание из аккумулятора косвенно-адресуемого и переноса	SUBB A, @Ri	1	1	(A) ← (A) - (C) - ((Ri))
Вычитание из аккумулятора константы и переноса.	SUBB A, #d8	2	1	(A) ← (A) - (C) - #d8
Инкремент аккумулятора	INC A	1	1	(A) ← (A) + 1
Инкремент регистра	INC Rn	1	1	(Rn) ← (Rn) + 1
Инкремент прямо-адресуемого байта	INC ad8	2	1	(ad8) ← (ad8) + 1
Инкремент косвенно-адресуемого байта	INC @Ri	1	1	((Ri)) ← ((Ri)) + 1
Инкремент указателя данных.	INC DPTR	1	2	(DPTR)←(DPTR)+ 1
Декремент аккумулятора.	DEC A	1	1	(A) ← (A) – 1
Декремент регистра.	DEC Rn	1	1	(Rn) ← (Rn) – 1
Декремент прямо-адресуемого байта	DEC ad8	2	1	(ad8) ← (ad8) – 1
Декремент косвенно-адресуемого байта	DEC @Ri	1	1	((Ri)) ← ((Ri)) – 1
Умножение аккумулятора на регистр-расширитель B.	MUL AB	1	4	(B)(A) ← (A)*(B)
Деление аккумулятора на регистр-расширитель B.	DIV AB	1	4	(B).(A) ← (A)/(B)
Логические операции				
Логическое «И» аккумулятора и регистра.	ANL A, Rn	1	1	(A) ← (A) AND (Rn)
Логическое «И» аккумулятора и прямо-адресуемого байта.	ANL A, ad8	2	1	(A) ← (A) AND (ad8)
Логическое «И» аккумулятора и косвенно-адресуемого байта.	ANL A, @Ri	1	1	(A) ← (A) AND ((Ri))
Логическое «И» аккумулятора и константы.	ANL A, #d8	2	1	(A) ← (A) AND #d
Логическое «И» прямо-адресуемого байта и аккумулятора.	ANL ad8, A	2	1	(ad8) ← (ad8) AND (A)
Логическое «И» прямо-адресуемого байта и константы.	ANL ad8, #d8	3	2	(ad8) ← (ad8) AND #d8
Логическое «ИЛИ» аккумулятора и регистра.	ORL A, Rn	1	1	(A) ← (A) OR (Rn)
Логическое «ИЛИ» аккумулятора и прямо-адресуемого байта.	ORL A, ad8	2	1	(A) ← (A) OR (ad8)
Логическое «ИЛИ» аккумулятора и косвенно-адресуемого байта.	ORL A, @Ri	1	1	(A) ← (A) OR ((Ri))

Логическое «ИЛИ» аккумулятора и константы.	ORL A, #d8	2	1	(A) ← (A) OR #d
Логическое «ИЛИ» прямо-адресуемого байта и аккумулятора.	ORL ad8, A	2	1	(ad8) ← (ad8) OR (A)
Логическое «ИЛИ» прямо-адресуемого байта и константы.	ORL ad8, #d8	3	2	(ad8) ← (ad8) OR #d8
Исключающее «ИЛИ» аккумулятора и регистра.	XRL A, Rn	1	1	(A) ← (A) XOR (Rn)
Исключающее «ИЛИ» аккумулятора и прямо-адресуемого байта.	XRL A, ad8	2	1	(A) ← (A) XOR (ad8)
Исключающее «ИЛИ» аккумулятора и косвенно-адресуемого байта.	XRL A, @Ri	1	1	(A) ← (A)xor((Ri))
Исключающее «ИЛИ» аккумулятора и константы.	XRL A, #d8	2	1	(A)←(A)xor #d8
Исключающее «ИЛИ» прямо-адресуемого байта и аккумулятора.	XRL ad8, A	2	1	(ad8) ← (ad8)xor(A)
Исключающее «ИЛИ» прямо-адресуемого байта и константы.	XRL ad8, #d8	3	2	(ad8)←(ad8)xor #d8
Сброс аккумулятора.	CLR A	1	1	(A) ← 0
Инверсия аккумулятора.	CPL A	1	1	(A) ← not(A)
Сдвиг аккумулятора влево циклический	RL A	1	1	(A_{n+1})←(A_n),n=0÷6, (A_0) ← (A_7)
Сдвиг аккумулятора влево через перенос	RLC A	1	1	(A_{n+1})←(A_n), n=0÷6 (A_0)←(C),(C)←(A_7)
Сдвиг аккумулятора вправо циклический	RR A	1	1	(A_n)←(A_{n+1}),n=0÷6, (A_7)←(A_0)
Сдвиг аккумулятора вправо через перенос	RRC A	1	1	(A_n)←(A_{n+1}), n=0÷6 (A_7)←(C),(C)←(A_0)
Обмен тетрадами в аккумуляторе.	SWAP A	1	1	$(A_{0\ldots3})$ ↔ $(A_{4\ldots7})$
Операции с отдельными битами				
Сброс переноса.	CLR C	1	1	(C) ← 0
Сброс бита.	CLR bit8	2	1	(bit8)) ← 0
Установка переноса.	SETB C	1	1	(C) ← 1
Установка бита.	SETB bit8	2	1	(bit8) ← 1
Инверсия переноса.	CPL C	1	1	(C) ← not (C)
Инверсия бита.	CPL bit8	2	1	(bit8) ← not(bit8)
Логическое «И» бита и переноса.	ANL C,bit8	2	2	(C)←(C) and (bit8)
Логическое «ИЛИ» бита и переноса.	ORL C,bit8	2	2	(C) ← (C) or (bit8)
Логическое «И» инверсии бита и переноса.	ANL C,/bit8	2	2	(C) ← (C) and (not(bit8))
Логическое «ИЛИ» инверсии бита и переноса.	ORL C,/bit8	2	2	(C) ← (C) or (not(bit8))
Пересылка бита в перенос.	MOV C,bit8	2	1	(C) ← (bit8)
Пересылка переноса в бит.	MOV bit8,C	2	2	(bit8) ← (C)
Операции передачи управления				
Безусловный длинный переход.	LJMP ad16	3	2	(PC) ← ad16
Безусловный переход в пределах страницы памяти 2 Кб.	LJMP ad11	2	2	$(PC_{0\text{-}10})$ ← ad11
Безусловный переход в пределах страницы памяти в 256 байт.	SJMP rel	2	2	(PC) ← (PC)+ rel
Безусловный косвенный переход.	JMP @A+DPTR	1	2	(PC) ← (A) + (DPTR)
Переход, если аккумулятор равен нулю.	JZ rel	2	2	(A)=0→

				(PC)←(PC)+rel (A)≠0→ (PC)←(PC)+2
Переход, если аккумулятор не равен нулю.	JNZ rel	2	2	(A)≠0→ (PC)←(PC)+rel (A)=0→ (PC)←(PC)+2
Переход, если перенос равен единице.	JC rel	2	2	(C)=1→ (PC)←(PC)+rel (C)=0→ (PC)←(PC)+2
Переход, если перенос равен нулю.	JNC rel	2	2	(C)=0→ (PC)←(PC)+rel (C)=1→ (PC)←(PC)+2
Переход, если бит установлен.	JB bit8,rel	3	2	(bit8)=1→ (PC)←(PC)+rel (bit8)=0→ (PC)←(PC)+3
Переход, если бит сброшен.	JNB bit8,rel	3	2	(bit8)=0→ (PC)←(PC)+3 (bit8)=1→ (PC)←(PC)+rel
Переход, если бит установлен, с последующим сбросом бита.	JBC bit8,rel	3	2	(bit8)=1→ (PC)←(PC)+rel, (bit8)=0 (bit8)=0→ (PC)←(PC)+3
Декремент регистра и переход, если результат не нулевой.	DJNZ Rn,rel	2	2	(Rn) ← (Rn) – 1; (Rn)=0→ (PC)←(PC) + 2 (Rn) ≠0 → (PC)←(PC) + rel
Декремент прямо-адресуемого байта и переход, если результат не нулевой.	DJNZ ad8,rel	3	2	(ad8) ← (ad8) – 1; (ad8) =0 → (PC)←(PC)+3 (ad8) ≠0 → (PC)←(PC)+rel
Сравнение аккумулятора с прямо-адресуемым байтом и переход, если не равно.	CJNE A, ad8, rel	3	2	(ad8) =(A)→ (PC)←(PC)+3 (ad8) ≠(A)→ (PC)←(PC)+rel
Сравнение аккумулятора с константой и переход, если не равно.	CJNE A, #d8, rel	3	2	(#d8) =(A)→ (PC)←(PC)+3 +(#d8) ≠(A)→ (PC)←(PC)+rel
Сравнение регистра с константой и переход, если не равно.	CJNE Rn, #d8, rel	3	2	(#d8) =(Rn)→ (PC)←(PC)+3 (#d8) ≠(Rn)→ (PC)←(PC)+rel
Сравнение косвенно-адресуемого байта с константой и переход, если не равно.	CJNE @Ri, #d8, rel	3	2	(#d8) =((Ri))→ (PC)←(PC)+3 (#d8) ≠((Ri))→

				(PC)←(PC)+rel
Безусловный длинный вызов подпрограммы.	LCALL ad16	3	2	(SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC) ← ad16
Безусловный вызов подпрограммы в пределах страницы памяти 2 Кб.	ACALL ad11	2	2	(SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC_{0-10}) ← ad11
Возврат из подпрограммы.	RET	1	2	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Возврат из подпрограммы обработки прерывания.	RETI	1	2	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Пустая операция.	NOP	1	1	(PC) ← (PC) + 1

Условные обозначения.

- Rn, где (n = 0…7) - регистр общего назначения в выбранном банке.
- @Ri, где (i = 0,1) - регистр общего назначения в выбранном банке, используемый в качестве регистра косвенного адреса.
- ad8 - адрес прямо-адресуемого байта в резидентной памяти данных и регистров специальных функций.
- ad11 - 11-разрядный адрес перехода в пределах страницы программной памяти 2KB.
- ad16 - 16-разрядный адрес перехода в пределах всего объема программной памяти.
- rel - относительный адрес перехода (8-разрядное число со знаком).
- #d8 - непосредственный операнд (8-разрядная константа).
- #d16 - непосредственный операнд (16-разрядная константа).
- bit8 - адрес прямо-адресуемого бита.
- /bit8 - инверсия прямо-адресуемого бита.
- A – аккумулятор.
- PC – программный счетчик.
- DPTR - регистр указатель данных.
- SP – регистр указатель стека.
- ( ) - содержимое ячейки памяти или регистра.

Литература.

1. Ю.И. Гудков, С.Н. Сафонов, А.Л. Тув «Универсальный лабораторный стенд для изучения однокристальных микроконтроллеров»: Методическое руководство к лабораторному практикуму. М: МИЭМ НИУ ВШЭ, 2019.