МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ”

для студентів спеціальності
141 – Електроенергетика, електротехніка та електромеханіка
освітня програма «Електротехнічні системи електроспоживання»
усіх форм навчання

Частина 1

2021

Конспект лекцій з дисципліни «Основи мікропроцесорної техніки» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітня програма «Електротехнічні системи електроспоживання» усіх форм навчання. Частина 1. /Укл: О.С. Назарова, В.В. Осадчий – Запоріжжя: НУ «Запорізька політехніка», 2021. – 42 с.

Укладач: О. С. Назарова, к.т.н., доцент
В. В. Осадчий, к.т.н., доцент

Рецензент: А.В. Пирожок, к.т.н., доцент

Відповідальний за випуск: О.С. Назарова, к.т.н., доцент

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 07 від 02.12.2020 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 05 від 18.02.2021 р.

ЗМІСТ

ПЕРЕДМОВА

Конспект лекцій містить матеріали з вивчення дисципліни «Основи мікропроцесорної техніки (ОМПТ)» у відповідності до навчальних планів ОКР бакалаврів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електротехнічні системи електроспоживання».

Мета навчити студентів застосовувати основи мікропроцесорної техніки, функціональні можливості мікропроцесорних систем і призначення пов'язаних з ними об'єктів при виконанні інженерних завдань за фахом.

Завдання сформувати у студентів знання, вміння та навички, необхідні для розуміння питань щодо програмування МК-51, ADuC 841, здійснення перевірки розроблених програм за допомогою емулятора Franklin Software, програми «WDS Analog Devices»; ознайомитися з можливостями мікропроцесорних засобів автоматизації фірми SIEMENS.

Для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електротехнічні системи електроспоживання»усіх форм навчання.

1 ФОРМИ ПРЕДСТАВЛЕННЯ ДАНИХ У МІКРОПРОЦЕСОРНИХ СИСТЕМАХ

Всі мікропроцесорні комплекти використовують двійкову систему числення, коли для представлення чисел використовується цифрова база: 0 і 1 [1,2].

Єдиної інформацією в мікропроцесорних системах є біт, який може мати стан 0 або 1.

Логічний «0»: 0 ... 0,2 В

Логічна «1»: 2,4 ... 5В

Кратними інформаційними одиницями є:

1 байт = 8 бітів, (за допомогою байта можна описати числа від 0 до 255)

Наприклад, інформація, що описується числом 155 (наприклад, рівень температури, рівень звуку в MP3 плеєрі і т.д.). Для передачі у вигляді байта використовується група з 8 провідників. Кожен з них має свою вагу, який визначається ступенем при 2 (табл. 1.1):

Таблиця 1.1 – Приклад опису інформації у двійковому коді.

$2^7 = 128$	1	старший біт
$2^6 = 64$	0	
$2^5 = 32$	0	
$2^4 = 16$	1	
$2^3 = 8$	1	
$2^2 = 4$	0	
$2^1 = 2$	1	
$2^0 = 1$	1	молодший біт

155-128=27
27-16=11
11-8=3
3-2=1
1-1=0
155= 10011011 B (B - binary або двійкова система числення)

1 слово = 2 байта = 16 бітів (з допомогою слова можна описати числа від 0 до 65535)

1 Кбайт = 210 байта = 1024 байта,

1 Мбайт = 210 Кбайта = 1024 Кбайта,

1 Гбайт = 210 Мбайта = 1024 Мбайта.

Недоліком двійкової форми подання інформації в мікропроцесорних системах є громіздкість одержуваних чисел, тому в мікропроцесорних комплектах для опису двійкових чисел використовують 16-кову форму їх подання (HEX-код). З умови, що один 16-ковий знак описує 4 біта, то з'являється можливість істотно скоротити число знаків для опису двійкових чисел [3]. Це особливо важливо, коли необхідно реалізувати введення інформації та її індикацію в мікропроцесорних системах.

У 16-ковій системі числення використовується цифрова база: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Для опису двійкового числа в 16-ковій системі числення користуються такими алгоритмами.

У 16-ковій системі числення один 16-ковий знак описує 4 біта, тому все двійкове число розбивається справа наліво на групи з 4 бітів, кожен з яких має наступні ваги:

1-ий біт (молодший) – $2^0 = 1$;

2-ий біт – $2^1 = 2$;

3-ий біт – $2^2 = 4$;

4-ий біт (старший) – $2^3 = 8$;

У таблиці 1.2 наведено перетворення тетради з двійкової форми представлення у шістнадцяткову та десяткову.

Таблиця 1.2 – Таблиця перетворення тетради з двійкової форми представлення у шістнадцяткову та десяткову

BIN (двійкова)				HEX (шістнад-цяткова)	DEC (десят-кова)
$2^3 = 8$ ст.біт	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$ мол.біт		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	A	10
1	0	1	1	B	11
1	1	0	0	C	12
1	1	0	1	D	13
1	1	1	0	E	14
1	1	1	1	F	15

Таблиця 1.3 - Приклад перетворення чисел з двійкової системи у шістнадцяткову

Двійкова система								Шістнадцяткова система
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	
8	4	2	1	8	4	2	1	
1	1	0	1	0	1	1	0	= 0D6H
0	1	1	1	1	0	0	1	= 79H

Для опису роботи дискретних і цифрових пристроїв [4] використовується математичний апарат алгебри логіки або булевої алгебри. Основними поняттями булевої алгебри є логічна змінна й логічна функція [1].

Логічна змінна – це величина, яка може приймати тільки два можливі значення, одне з яких за твердженням вважається «справжнім» а друге – «помилковим». У булевій алгебрі справжнє значення змінної позначається символом «1» (логічна одиниця), неправильне – символом «0» (логічний нуль).

Арифметичні операції з двійковими числами.

Додавання

0+0=0

0+1=1

1+0=1

1+1=0 и перенос 1 в наступний біт

Приклад використання операції «додавання» з числами у двійковій системі

+	0	1	1	0	0	0	1	1
	0	1	0	0	1	0	1	0
=	1	0	1	0	1	1	0	1

Віднімання

0-0=0

1-0=1

0-1=1 і потім з біта, в якому є одиниця

1-1=0

Приклад використання операції «віднімання» з числами у двійковій системі

-	0	1	1	0	0	0	1	1
	0	1	0	0	1	0	1	0
=	0	0	0	1	1	0	0	1

-	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0
=	1	1	1	1	1	1	1	0

2 ЗАГАЛЬНА ХАРАКТЕРИСТИКА МК-51

Мікропроцесор (МП) - пристрій, який обробляє інформацію відповідно до програми, яка подається по командам на його входи, і реалізований у вигляді однієї або декількох великих інтегральних схем (ВІС) [5].

Сукупність інтегральних схем (ІС), сумісних з конструктивно-технологічного виконання і призначених для спільного застосування при побудові мікро-ЕОМ і мікропроцесорних контролерів, називаються «мікропроцесорним комплектом» (chipset) [6].

Мікроконтролер (МК) - пристрій у вигляді БІС, що містить в собі мікропроцесор, пам'ять, порти вводу-виводу і інші периферійні пристрої, які дозволяють організувати функціонально закінчену мікропроцесорну систему (іноді з можливістю розширення функціональних можливостей) [7].

Рисунок 2.1 – Завантаження жорсткого диску об’ємом 5 Мб (1956 р.)

За функціональним призначенням мікропроцесорні ВІС можна розділити на два класи:

- універсальні ВІС (КР580ИК80А, К1810ВМ86, К586ІК1, INTEL-4004, 8008,8080, 8086, 8088, що 80186, 80286, 80386, і т.д.);
- спеціалізовані БІС (К1800ВС1, К1802ВС1, К1804ВС1Б, К1804ВС2К531Ік1, КР5821ХК1 і т.д.).

Мікропроцесорні комплекти діляться за такими ознаками:

- тип мікроелектронної технології, використовуваної при виготовленні ВІС комплекту, який визначає швидкодію ВІС, номінальні рівні використовуваних сигналів («логічний 0», «логічна 1»), потужність БІС (здатність навантаження входів і виходів ВІС);
- структури кристалів ВІС, кількість елементів (транзисторів) на кристалі, кількість висновків корпуса ВІС;
- довжина слова (кількість розрядів), оброблюваного МП і пересилаються між БІС і мікропроцесорної системою (МПС);
- розрядність шини адреси, що визначає обсяг пам'яті;
- тип керуючого пристрою (схемне або вбудоване);
- система команд, кількість команд, що виконуються операції, можливі способи адресації;
- число рівнів переривання і можливість прямого доступу до пам'яті;
- пропускна здатність інтерфейсів вводу / виводу;
- кількість і рівні напруги живлення.
- число входять до мікропроцесорний комплект ВІС.

Перші мікропроцесорні пристрої з'явилися на ринку в 1973 р. Фірма Intel запропонувала власне мікропроцесорний комплект – сам мікропроцесор Intel 8080 у вигляді одного кристала й комплект периферійних пристроїв у вигляді окремих ВІС, що містять паралельний інтерфейс, контролер переривань, таймер триканальний, контролер прямого доступу до пам'яті й модуль послідовного інтерфейсу. Побудова систем керування, що працюють у реальному часі, коли реакція системи на мінливі параметри об'єкта повинна бути миттєвою (швидкою), виходила громіздкою і недешевою. Удосконалення мікропроцесорів (збільшення розрядності й частоти тактування відбувається відповідно до прогресу технологій, а для систем керування, працюючих у реальному часі, фірмою Intel була запропонована структура ВІС, у якої на один кристал, крім мікропроцесора, інтегровано всі периферійні пристрої: паралельний

інтерфейс, таймери, контролер переривань, послідовний інтерфейс, пам'ять програм і пам'ять даних. Це дозволило скоротити апаратні витрати й здешевити системи керування. Такі ВІС одержали назву мікроконтролери (МК). Перший МК фірмою Intel мав марку 8051 АН (у СРСР було випущено аналог ДО1816ВЕ51), а сім'я мікроконтролерів одержала позначення MCS-51 [8].

Архітектура сімейства MCS-51 виявилася настільки вдалою, що серед 8-розрядних мікроконтролерів на світовому ринку вже багато років вона займає лідируючі позиції. Крім фірми Intel, мікроконтролери з архітектурою MCS-51 випускають фірми Siemens, Philips, Atmel та ін [1].

Удосконалення технології й підвищення ступеня інтеграції розширюють функції периферійних пристроїв, збільшують продуктивність, обсяги пам'яті програм і даних, але саме ядро команд MCS-51 залишається повністю сумісним з молодшими моделями, що дозволяє використовувати величезний накопичений досвід розробки й налагодження програмного забезпечення.

В даному курсі лекцій вивчення основ мікропроцесорної техніки буде проводитися на прикладі 8-розрядного мікроконтролера INTEL 8051.

Восьми розрядні однокристальних мікро ЕОМ сімейства MCS 51 спочатку були виконані по n-МОП технології фірмою INTEL. Це пристрій поклало початок цілому сімейству мікроконтролерів, які можна об'єднати назвою MCS 51 (Micro Processor Command Set 51).

Загальним для мікропроцесорів цієї системи є система команд, розроблена спочатку для процесора INTEL 8051. Цей мікроконтролер включає до свого складу всі компоненти для побудови закінченої мікропроцесорної системи, але в той же час дозволяє розширити її функціональні можливості додатковими модулями і мікросхемами.

Структурна схема приведена на рисунку 2.1 [3].

Мікроконтролери INTEL 8051 містять такі вузли, необхідні для автономної роботи [3,9]:

- центральний 8-ми розрядний процесор на основі арифметико-логічного пристрою (АЛУ);
- пам'ять програм (ППЗУ) обсягом 4 кБ;
- пам'ять даних обсягом 128 байт;
- чотири 8-ми розрядних програмованих портів введення / виведення (P0, P1, P2, P3);

- два 16-ти розрядних таймерів-лічильників;
- система переривань з п'ятьма векторами переривань і двома рівнями пріоритетів;
- послідовний інтерфейс;
- тактовий генератор.

Ця мікросхема (рис. 2.2) має 32 регістри загального призначення (РОН) (в один момент доступні 8 регістрів R0 ... R7), 128 визначених користувачем програмно керованих прапорів (бітів), набір регістрів спеціальних функцій (SFR). РОН і визначені користувачем програми керовані прапори розташовані в адресному просторі внутрішнього ОЗУ даних.

Регістрами називаються цифрові пристрої, використовувані для запису й зберігання n -розрядного двійкового слова [1]. Вони будуються на базі тригерів, число яких у схемі регістру звичайно відповідає числу розрядів двійкового слова. Особливістю роботи схеми регістру є те, що запис, зчитування або перетворення інформації відбувається одночасно у всіх його тригерах. У багаторозрядних оперативних запам'ятовувальних пристроях (ОЗП) регістри є базовими елементами. Їх називають комірками пам'яті. У мікроконтролерах розрядність таких гнізд звичайно становить 8 або 16 біт. Крім зберігання інформації, в деяких типах регістрів забезпечується можливість виконання додаткових функцій для її перетворення. До них відносяться: перетворення даних з паралельної форми подання в послідовну, і навпаки; перетворення прямого коду у зворотний; виконання зрушення на один або кілька розрядів убік молодшого або старшого розрядів; інкрементування або декрементирування вмісту регістру. У структурі мікропроцесора це основний елемент.

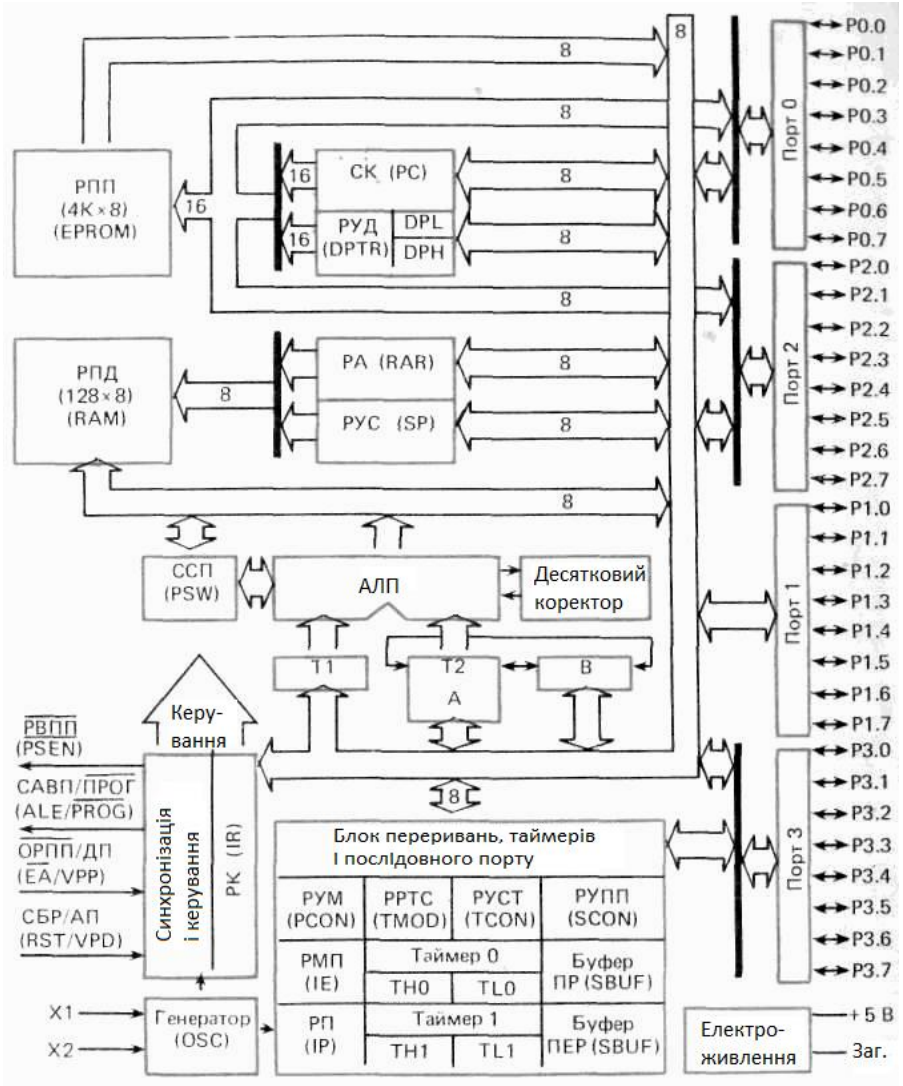


Рисунок 2.1 - Структурна схема мікропроцесора INTEL 8051

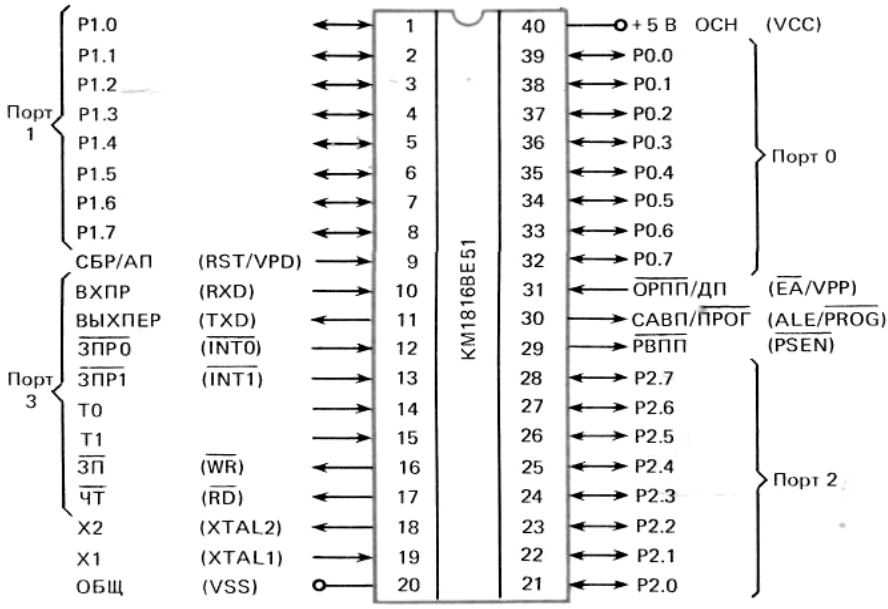


Рисунок 2.2 - Мікросхема MCS 51

Регістри спеціальних функцій (SFR) із зазначенням адрес наведені в таблиці 2.1

Таблиця 2.1 - Блок регістрів спеціальних функцій

Символ		Назва	Адреса
*	ACC	Акумулятор	OEOH
*	B	Регістр- розширювач акумулятора	0F0H
*	PSW	Слово стану програми	0D0H
	SP	Регістр- покажчик стека	81H
	DPTR	Регістр- покажчик даних(DPH) (DPL)	83H82H
*	PO	Порт 0	80H
*	P1	Порт 1	90H
*	P2	Порт 2	OAOH

Продовження таблиці 2.1

*	P3	Порт 3	OBOH
*	IP	Регістр пріорітетів	0B8H
*	IE	Регістр маски переривань	0A8H
	TMOD	Регістр режима таймера	89H
*	TCON	Регістр управління/статуса таймера	88H
	THO	Таймер 0 (старший байт)	8CH
	TLO	Таймер 0 (молодший байт)	8AH
	TH1	Таймер 1 (старший байт)	8DH
	TL1	Таймер 1 (молодший байт)	8BH
*	SCON	Регістр управління прийомопередатчиком	98H
	SBUF	Буфер прийомопередатчика	99H
	PCON	Регістр управління потужністю	87H

Примітка. Регістри, імена яких відзначені знаком (*), допускають адресацію окремих біт.

2.1 Арифметико логічний пристрій (АЛП)

8-ми бітове пристрій може виконувати арифметичні дії: додавання, віднімання, множення і ділення; логічні операції: «І», «АБО», що виключають «АБО», а також операції циклічного зсуву, арифметичного зсуву, скидання, інвертування і ін.

Найпростіша операція додавання використовується в АЛП для інкрементування вмісту регістрів, просування регістра покажчика даних (DPTR) і автоматичного обчислення наступного адреси резидентної пам'яті програми [3].

Найпростіша операція віднімання використовується для декрементирування регістрів і порівняння змінних.

Найпростіші команди автоматично утворюють пару для виконання в АЛП таких операцій як інкрементування 16-ти розрядних бітних пар.

В АЛП реалізується механізм каскадного виконання найпростіших операцій для реалізації складних команд.

Для виконання однієї з команд за результатами порівняння в АЛП здійснюють такі дії:

- тричі декрементується лічильник команд РС;
- двічі проводиться читання;
- виконується порівняння двох змінних;
- формується 16-ти бітний адреса переходу;
- приймається рішення робити або не робити перехід.

Дуже важливою особливістю АЛП є його здатність оперувати не тільки з байтами, але і з бітами. В окремих програмах доступні біти можуть бути встановлені (в одиницю), скинуті (в нуль), інвертировані, передані, перевірені та використані в логічних операціях.

Ця здатність АЛП оперувати з бітами настільки важлива, що в багатьох описах процесора МК-51 йдеться про наявність в ньому булевского процесора [3].

Для управління об'єктами часто застосовують алгоритми з операціями над вхідними та вихідними логічними змінними, реалізація яких засобами звичайних процесорів сполучення з певними труднощами.

АЛП може оперувати з 4-ма типами операндів: Булевського (1 біт), цифровими (4 біта), байтними (8 біт) і адресними (16 біт).

АЛП може виконувати 51 різну операцію з пересилання або перетворення даних. Також використовується 11 режимів адресації (7 для даних і 4 для адрес). Тому шляхом комбінування операція / режим адресації базове число команд 111 розширюється до 255 з 256 можливих при однобайтному ході операції.

2.2 Резидентна (внутрішня) пам’ять

Пам’ять програм і пам’ять даних, які розміщені на кристалі процесора фізично і логічно розділені і мають різні способи адресації. Також вони працюють під управлінням різних сигналів і виконують різні функції [3]. Така організація пам’яті процесора називається гарвардською структурою процесора (процесори мають загальний адресний простір ОЗУ і ПЗУ називаються фон Неманська структура).

ПЗП (ROM – Read Only Memory) – пристрій для зберігання й зчитування незмінних даних. Цей вид пам'яті призначено для зберігання програми, а також незмінних констант. Така пам'ять є енергонезалежною – при відключенні живлення записана інформація зберігається. Після поновлення живлення вона може бути зчитана.

ОЗП (RAM – Random-Access Memory) – пристрій для запису, зберігання й зчитування змінюваних даних. У даному виді пам'яті зберігається інформація, що модифікується й використовується в процесі роботи. Це можуть бути різні змінні або результати проміжних і остаточних обчислень. Інформація при такому виді пам'яті губиться після вимикання живлення [1].

Пам’ять програм (ПЗУ або РПП) має ємність 4кБ і призначена для зберігання команд, констант, керуючих слів ініціалізації таблиць різних ввідних і вихідних змінних (перекодування) і т.д.

РПП має 16-ти бітну шину адреси, через яку забезпечується доступ з регістра команд (DPTR). DPTR виконує функції базового регістра при непрямих переходах підпрограми або використовується в командах оперують з таблицями.

Пам’ять даних ОЗУ або РПД призначена для зберігання змінних, які змінюються в процесі виконання програми. Адресуються 1-м байтом і мають ємність 128 байт. Крім того, до адресного простору РПД додають такі адреси регістрів спеціальних функцій SFR.

Пам’ять програм також як і пам'ять даних може бути розширено до 64 Кб кожна шляхом підключення зовнішніх БІС пам'яті.

2.3 Акумулятор і слова стану програми PSW

Акумулятор є джерелом операнда і місцем фіксації операнда-результату при виконанні арифметичних, логічних і ряду операцій передачі даних [3]. Крім того тільки з використанням акумулятора можуть бути виконані операції зсуву, перевірка на нуль, формування прапора паритету і т.д.

При виконанні багатьох команд формуються також ряд ознак, які фіксуються в слові стану програми (PSW (programm state word)).

У таблиці 1.2 наводиться перелік прапорів PSW, даються їх символічні імена і описуються умови формування.

Найбільш активним бітом в регістрі стану програми є прапор переносу С, який змінюється в процесі виконання багатьох операцій:

арифметичних, логічних, зсуву. Крім того прапор З виконує функцію булевого акумулятора в командах які маніпулюють з бітами.

Прапор переповнення OV фіксує арифметичне переповнення в операціях з фіксованими числами зі знаком і робить можливим використання арифметики з додатковим кодом.

Таблиця 2.2 - Формат слова стану програми (ССП).

Символ	Позиція	Ім’я та призначення
C	PSW.7	Прапор переносу. Встановлюється і скидається апаратурними засобами або програмою при виконанні арифметичних і логічних операцій
AC	PSW.6	Прапор допоміжного переносу. Встановлюється і скидається тільки апаратурними засобами при виполне¬ніі команд додавання і віднімання і сигналізує про перенесення або позику в бите 3
F0	PSW.5	Прапор 0. Може бути встановлений, скинутий або перевірений програмою як прапор, специфіціруемий користувачем
RS1	PSW.4	Вибір банку регістрів. Встановлюється і скидається програмою для вибору робочого банку регістрів (див. примітку)
RS0	PSW.3	
OV	PSW.2	Прапор переповнення. Встановлюється і скидається аппаратурно при виконанні арифметичних операцій
-	PSW.1	Не використовується
P	PSW.0	Прапор паритету. Встановлюється і скидається апаратно в кожному циклі команди і фіксує непарне / парне число одиничних біт в акумуляторі, тобто виконує контроль по парності

АЛП не керує бітом селекції вибору банків (RS0, RS1). Їх значення повністю визначається програмою і використовується для вибору одного з 4-х банків регістра.

Слово стану програми (PSW) включає в себе чотири прапори [3]:

C - перенесення;

AC - допоміжний перенесення;

OV - переповнення;

P - паритет.

Прапор паритету безпосередньо залежить від поточного значення акумулятора. Якщо число одиничних бітів акумулятора непарне, то прапор P встановлюється, а якщо парне - скидається.

Прапор AC встановлюється в разі, якщо при виконанні операції додавання / віднімання між тетрадами байта виник перенос / позику.

Прапор C встановлюється, якщо в старшому біті результату виникає перенесення або позику. При виконанні операцій множення і ділення прапор З скидається.

Прапор OV встановлюється, якщо результат операції додавання / віднімання не вкладається в семи бітах і старший (восьмий) біт результату не може інтерпретуватися як знаковий. При виконанні операції ділення прапор OV скидається, а в разі поділу на нуль встановлюється. При множенні прапор OV встановлюється, якщо результат більше 255.

Таблиця 2.3 - Вибір робочого банку регістрів

RS1	RS0	Банк	Адреси
0	0	0	00H – 07H
0	1	1	08H – 1FH
1	0	2	10H – 1FH
1	1	3	18H – 1FH

Широке поширення набуло уявлення про те, що мікропроцесор спираються на акумулятор, більшість команд працює з ним. У процесорі INTEL 8051 інша справа, хоча цей мікропроцесор має один акумулятор, проте він може виконати безліч команд без його участі. Наприклад будь-який регістр може бути переданий з будь-якої комірки ОЗУ в регістр. Будь-який регістр може бути завантажений

безпосереднім операндом. Крім того змінні можуть бути перевірені, інкрементирувати і декрементирувати без використання акумулятора.

2.4 Регістри покажчики

8-ми бітний покажчик стека (SP) може адресувати будь-яку область внутрішньої ОЗУ. Його вміст інкрементирується, перш ніж дані будуть заповнені в стеці в ході виконання команди PUSH. Вміст SP декрементирується після виконання команд POP і RET. Подібний спосіб адресації стека преінкрементний (поступальний) [1,3].

Після сигналу RESET в SP завантажується код 07, це означає, що перший елемент в сітці буде розташовуватися 08H і вище.

2-х байтний регістр покажчика даних DPTR зазвичай використовують для фіксації 16-ти розрядної адреси або як два нерівних 8-ми розрядних регістра DPH (high), DPL (low).

SBUF представляє собою два незалежних регістра (приймача і передавача). Завантаження байтів SBUF викликає початок передачі через останній порт, коли з SBUF зчитується байт, його джерелом є приймач останнього порту.

IE, TCON, SCON, PCON використовується для фіксації керуючих біт, стану схеми переривання, таймерів-лічильників, приймачів після порту і схеми живлення.

2.5 Пристрій управління і синхронізації. Часові параметри роботи мікропроцесорів

Кварцової резонатор підключають до XTAL1 (X1), XTAL2 (X2) і управляє роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації [1,3].

Пристрій на основі сигналів синхронізації формує машинний цикл фіксованої тривалості, який дорівнює 12-ти періодів кварцового резонансу. Більшість команд виконується за один машинний цикл.

Деякі команди оперують з 2-х байтними словами або використовують звернення до зовнішньої пам'яті виконуються за два машинних циклу і тільки команди множення і ділення вимагають чотири машинних циклу. На основі цих особливостей пристрою управління проводиться розрахунок тривалості виконання програм.

3 СТРУКТУРА КОМАНДНОГО РЯДКА

Мітка: команда операнд 1, операнд 2; комментар до команди.

Мітка - символьне позначення рядка програми. Може складатися з латинських букв і цифр. В імені мітки не використовуються розділові знаки та інші символи. Ім'я мітки не повинно збігатися з ім'ям будь-якої з існуючих команд. В одній програмі імена міток не можуть повторюватися.

Команда - умовно прийняте буквене позначення якої-небудь дії. Команда при компіляції програми буде перетворена в код, який формується з двох цифр в шістнадцятковій системі (1-ша з вертикалі, 2-га з горизонталі - див. Таблицю команд - дати відтворити і сказати про кольорові олівці - 5 шт.).

Операнд - число, над якими виконуються дії, або вказівку для знаходження цього числа.

Таблиця 3.1 - Види операндів

№	Назва	Умовне позначення	Приклад
1.	Регістри	Rn	R0 – R7
		Ri	R0, R1 – регістри покажчики
		A	акумулятор
		B	допоміжний регістр
2.	Числа константи	#data, #d8	#1AH; #00001111B
3.	Комірки пам'яті (адреса)	dir, ad	(20H)
4.	Порти	Pn	P0 – P3
5.	Біти	bit	P0.1; C – біт позички / перенесення; P - біт паритету (парності) і ін. Біти слова стану програми (PSW)

Коментар - буквено-символьний розшифровка або візуальне опис дії команди. Використовується на розсуд розробника програми. Обов'язково відділяється від тексту команди знаком «;».

4 МЕТОДИ АДРЕСАЦІЇ

Методи адресації - це способи звернення до елементу даних на його адресу [1-3].

Регістровий метод адресації використовується для вказівки на операнди, розташовані в регістрі.

Якщо в команді міститься ім'я регістра Rn (R0, R1, R2, ..., R7) а також A, B, то така команда належить регістровому методу адресації.

наприклад:

```
MOV A, Rn          ; (Rn) → (A)
ADD A, Rn          ; (A) + (Rn) → (A)
```

Пряма байтова адресація використовується для звертання до комірок пам'яті.

Якщо в команді міститься адреса комірки dir (20H, 21H і т.д.), то така команда належить до прямої адресації.

Наприклад:

```
MOV A, dir         ; (dir) → (A)
ADD A, dir         ; (A) + (dir) → (A)
```

Безпосередня адресація використовує константи #data, які згідно чітко зазначених у в команді. Ознака константи в команді - знак «#».

Наприклад:

```
MOV A, #data       ; #data → (A)
ADD A, #data       ; (A) + #data → (A)
```

Непряма реєстрова адресація використовується для звертання до комірок пам'яті ОЗУ, адреса яких зазначений в регістрі-покажчику Ri.

Як правило, цими регістрами-покажчиками є R0 і R1.

Ознака побічно-реєстрової адресації - знак «@».

Наприклад:

```
MOV A, @Ri         ; ((Ri)) → (A),
```

@Ri - це вміст комірки пам'яті, адреса якої знаходиться в регістрі Ri.

```
ADD A, @Ri         ; A + ((Ri)) → A
```

Пряма побітова адресація використовується для звернення до окремо адресуються 128 бітам, розташованим в комірках пам'яті з адресами від 20H до 2FH і до окремо адресуються бітам регістрів спеціальних функцій.

наприклад:

MOV C, bit ; (bit) → (C)

bit - однобайтна адреса біта.

CLR bit ; (bit) = 0

SETB bit ; (bit) = 1

Система команд MCS 51

Система команд містить 111 базових команд.

Види команд по функціональності:

- передачі даних;
- арифметичні операції;
- логічні операції;
- операції з бітами;
- передачі управління.

Формат команд: більшість має 1 або 2 байта і виконується за 1 або 2 машинних циклу.

При тактовій частоті 12 МГц тривалість машинного циклу складає 1 мкс.

5 КОМАНДИ ПЕРЕДАЧІ ДАНИХ

Всі основні шляхи пересилання данних в контролері зобразимо у вигляді графа (рис. 5.1).

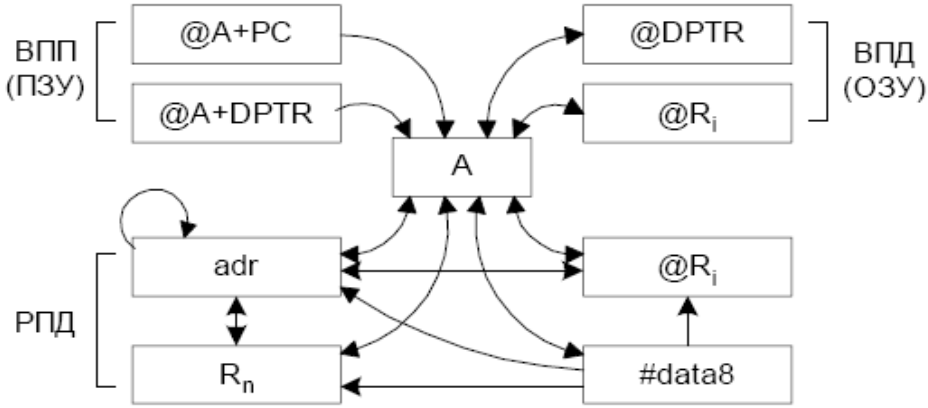


Рисунок 5.1 - Структурна схема основних шляхів передачі даних

@ A + PC - комірка пам'яті програм, адреса якої обчислюється як сума, вмісту лічильника команд PC і акумулятора (побічно реєстрова адресація);

@ A + DPTR - комірка пам'яті програм, адреса якої обчислюється як сума 16-розрядного регістра покажчика DPTR і акумулятора;

@DPTR - комірка зовнішньої пам'яті даних, адреса якого знаходиться в DPTR (адреса 16-розрядний);

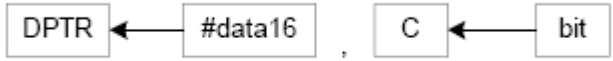
@Ri - комірка ВПД, 8-розрядний адреса якої знаходиться в Ri (i = 0,1);

adr (ad) - комірка, РПД адресу якої прямо вказано в команді;

Rn - операнд знаходиться в регістрі контролера, поточного реєстрового банку (n = 0,7);

Data8 (# d8) - 8-розрядне значення (байт).

Крім зазначених шляхів є ще два:



Тобто в DPTR можна завантажувати 16-розрядне значення, а в біт перенесення C - прямо адресований біт.

Можна виділити три типи команд пересилання:
- пересилання даних в РПД;
- пересилання даних в ВПД;
- читання даних з ВПП.

Більшу частину команд даної групи - команди передачі і обміну байтів. Всі ці команди не змінюють прапори результатів за винятком команди завантаження PSW і акумулятора.

MOV «приймач», «джерело»

1. MOV A, Rn ; (Rn) → A
де Rn- РОН (R0, R1, .., R7)
2. MOV A, ad ; (ad) → A
де ad- однобайтна адреса комірки пам'яті
3. MOV A, #d8 ;d8 → A
де d8- однобайтне число(константа)
4. MOV Rn, A ;A → Rn
5. MOV Rn, ad, ; (ad) → Rn
6. MOV Rn,#d8 ; d8→ Rn
7. MOV ad, A ;A → (ad)
8. MOV ad, Rn ;Rn → (ad)
9. MOV ad, #d8 ;d8 → (ad)
10. MOV add, ads ;(ads) → (add),
де ads - адреса комірки пам'яті джерела
add - адреса комірки пам'яті приймача
11. MOV DPTR, # d16 ; d16 → (DPTR),
де DPTR - 16-ти розрядний регістр - покажчик даних
D16 - 16-ти розрядна постійна
12. MOV A, @Ri ; ((Ri)) → A,
де @ Ri- це вміст комірки пам'яті, адреса якої знаходиться в регістрі Ri (R0 або R1)
13. MOV ad, @Ri ;((Ri)) → (ad)
14. MOV @Ri, A ;A → ((Ri))
15. MOV @Ri, ad ;(ad) → ((Ri))

16. MOV @Ri, #d8 ;d8 → ((Ri))
17. MOVC A, @A + DPTR ;((A)+ (DPTR)) → A
18. MOVC A, @A + PC ;((A)+ (PC)) → A

Для зовнішньої пам'яті данних:

19. MOVX A, @Ri ;((Ri))→ A
20. MOVX A ,@DPTR ; ((DPTR)) → A
21. MOVX @Ri, A ;A → ((Ri))
22. MOVX @DPTR, A ;A→ ((DPTR))

При зверненні до зовнішньої пам'яті з використанням MOVX @Ri - є 256 байт, при звернення до зовнішньої пам'яті з використанням MOVX @DPTR - 64к байт.

Команди обміну.

XCH A, Rn ;A ↔ Rn
XCH A, ad ;A ↔ (ad)
XCH A, @Ri ;A ↔ Ri
XCHD A, @Ri ;(A_{0-3}) ↔ (Ri_{0-3})

Обмін молодшої тетради акумулятора з молодшою тетрадою байта непрямо адресованої комірки.

Команди роботи зі стеком (Stack):

Стек організується в області пам'яті ОЗП. Працює за принципом: «перший прийшов - останній вийшов» (LIFO - Last Input - First Output). Покажчик верхівки стека SP (Stack Pointer) служить для зберігання адреси комірки ОЗП, що вказує на вершину стека. Вказівка адреси верхівки стека здійснюється командою:

MOV SP, # d8 ; d8 → SP

Завантаження даних в стек здійснюється я командою:

PUSH ad ; SP + 1 → SP
(Ad) → ((SP))

Повернення даних зі стека здійснюється командою:

POP ad ; ((SP)) → (ad)
SP-1 → SP

Акумулятор. У 51-му процесорі звернення до акумулятора може бути виконано реєстровою та прямою адресацією. Залежно від способу адресації акумулятора застосовується одне із символічних імен. Для реєстрової адресації A; для прямої - ACC (адреса 0E0H).

```
MOV A, 34H          ; (34H) → A
MOV ФСС, 34H        ; (34H) → (0E0H)
PUSH ACC            ; SP + 1 → SP
                    (0E0H) → ((SP))
```

При прямій адресації звернення до акумулятора здійснюється як до одного з P0H і його адреса вказується у 2му або 3му байті команди. Використання реєстрової адресації краще, однак, не завжди можливе (при зверненні до окремих бітів акумулятора).

6 АРИФМЕТИЧНІ КОМАНДИ

Дану групу складають 24 команди, які виконують операції додавання, віднімання, інкремента, декремента, множення і ділення [3].

Додавання

ADD «А», «операнд 2»; (А) + «операнд 2» →(А)

1.	ADD A, Rn	;A + Rn → A
2.	ADD A, dir	;A + (dir) → A
3.	ADD A, #d	;A + #d → A
4.	ADDC A, Rn	;A + Rn + (C) → A
5.	ADDC A, dir	;A + (dir) + (C) → A
6.	ADDC A, #d	;A + #d + (C) → A
7.	ADD A, @Ri	;A + ((Ri)) → A
8.	ADDC A, @Ri	;A + ((Ri)) + (C) → A

Віднімання

SUBB «А», «операнд 2»; (А) - «операнд 2» -(С)→(А)

9.	SUBB A, Rn	;A – Rn – (C) → A
10.	SUBB A, dir	;A – (dir) – (C) → A
11.	SUBB A, #d8	;A - #d – (C) → A
12.	SUBB A, @Ri	;A – ((Ri)) – (C) → A

Множення

13. MUL AB; (A) * (B) → (B) (A)

умовно приймаємо, що результат дії - в акумуляторі (ціле число)

Ділення

14. DIV AB; (A) / (B) → (A). (B)

умовно приймаємо, що результат дії - в акумуляторі (ціле число)

Інкремент

15.	INC A	;A + 1 → A
16.	INC Rn	;Rn + 1 → Rn
17.	INC dir	;(dir) + 1 → (dir)
18.	INC DPTR	;DPTR + 1 → DPTR
19.	INC @Ri	;((Ri)) + 1 → ((Ri))

Декремент

20.	DEC A	;A – 1 → A
21.	DEC Rn	;Rn – 1 → Rn
22.	DEC dir	;(dir)– 1 → (dir)
23.	DEC @Ri	;((Ri)) – 1 → ((Ri))

Двійково-десяткова корекція

24. DA A;

Приклад 6.1.

Арифметичні команди і команди передачі даних.

$(20H) + 12H - (21H) / 02H \rightarrow (22H)$

10 H 2AH

```
Org 0h
Mov a, 21h      ;(21H)→(a)
Mov b, #02H     ;02H→(b)
Div ab          ;(a)/(b) →(a).(b)
Mov r1,a        ;(a) →(r1)
Mov a, 20H      ;(20H))→(a)
Add a, #12h     ;(a)+12H →(a)
Clr c           ;(C)=0
Subb a, r1      ;(a)-(r1)-(C) →(a)
Mov 22h, a      ;(a) →(22H)
L1: jmp l1
end
```

2AH/02H=15H
10H+12H=22H
22H-15H=0DH

ПЕРЕЛІК ПОСИЛАНЬ

1. Мікропроцесорна техніка: навч. посібник / В.В. Ткачов, Г.Грулер, Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. – 188 с.

2. Белов, А. В. Самоучитель по микропроцессорной технике. – СПб.: Наука и техника, 2003. – 224 с.

3. Сташин В.В. Проектирование цифровых устройств на одно кристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Молногонцева – М.: Энергоатомиздат, 1990. – 224 с.

4. Карташов, Б. А. Системы автоматического регулирования с микроЭВМ / Б.А. Карташов, Е.А. Шабаев – Зерноград: АЧГАА, 2008. – 42 с.

5. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы / Под ред. В.Б. Смолова – М.: Радио и свіязь, 1981. – 328 с.

6. Липовецкий Г.П. и др. Однокристальные микроЭВМ семейств МК48, МК51. – М., 1992. – 344 с.

7. Ишматов, З. Ш. Микропроцессорное управление електроприводами и технологическими объектами. Полиномиальные методы. – Екатеринбург, УГТУ-УПИ, 2007. – 278 с.

8. Магда, Ю. С. Микроконтроллеры серии 8051: практический поход. – М. ДМК Пресс, 2008. – 228 с.

9. Горюнов А. Г., Ливенцов С.Н. Архитектура микроконтроллера INTEL 8051: Учебное пособие. – Томск, Изд-во ТПУ, 2005. – 86 с.

10. Веприк, В.Н. и др. Микроконтроллеры семейства MCS-51: Учебное пособие / В.Н. Веприк, В.А. Афанасьев, А.И. Дружинин, А.А. Земсков, А.Р. Исаев, О.В. – Новосибирск, 1997. – 62 с.

11. Назарова О. С. Ідентифікація кутової швидкості при завадах в оптичній системі енкодера / О.С. Назарова, В. В. Осадчий, І. А. Мелешко, М. О. Олєйніков // Вісник НТУ «ХПІ» - Харків, 2019. – С.65-69. http://doi.org/10.20998/2079-8024.2019.16.12

12. Sadovoi O., Nazarova O., Bondarenko V., Pirozhok A., Hutsol T., Nurek T., Glowacki Sz. Modeling and research of electromechanical systems of cold rolling mills. Monograph. – Krakow: Traicon, 2020. – 138 p.

13. Nazarova Olena. Computer Modeling of Multi-Mass Electromechanical Systems. The Third International Workshop on

Computer Modeling and Intelligent Systems (CMIS-2020), Vol. 2608, pp. 489-498. CEUR-WS.org/Vol-2608/paper36.pdf

14. Osadchyy V. and Nazarova O., "Laboratory Stand for Investigation of Liquid Level Microprocessor Control Systems," 2020 IEEE Problems of Automated Electrodrive. Theory and Practice (PAEP), Kremenchuk, Ukraine, 2020, pp. 1-4, doi: 10.1109/PAEP49887.2020.9240868

15. Методичні вказівки з виконання лабораторних робіт №1, 2, 3 "Ознайомлення зі стендом засобів автоматизації та приводи фірми SIEMENS" з дисципліни «Дискретна автоматика» для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА денної форми навчання. / Укладачі: Кулинич Е.М., Осадчий В.В. - Запоріжжя: ЗНТУ, 2017. - 70с.

16. Методичні вказівки з виконання лабораторних робіт №4, 5,6 «Автоматизація на основі LOGO! фірми SIEMENS» з дисципліни «Дискретна автоматика» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка денної форми навчання. / Укладачі: Кулинич Е.М., Осадчий В.В. – Запоріжжя: ЗНТУ, 2017. – 46 с.

17. Електроніка і мікропроцесорна техніка / Сенько В.І., Лисенко В.П., Юрченко О.М., Лукін В.Є., Руденський А.А. — К. : «Агроосвіта», 2015. — 676 с.

18. Коваленко, М. А. Автономний експериментальний стенд для випробування уніполярного крокового двигуна на базі мікроконтролера / М. А. Коваленко, Д.С. Мацюк // Електротехніка та електроенергетика. – 2015. – № 2. – С. 15-20.

19. Analog Devices – Режим доступу: https://www.analog.com/en/index.html.

20. Системи автоматизації для задоволення будь-яких вимог – Режим доступу: https://new.siemens.com/ua/uk/produkty/avtomatyzatsiya-promyslovosti/systemy-avtomatyzatsiyi.html

21. Дрешпак Н.С. Системи контролю енергоефективності виробничих процесів та шляхи їх удосконалення / Н.С. Дрешпак // Електротехніка та електроенергетика, № 1, 2020. - С. 40-48. DOI 10.15588/1607-6761-2020-1-5

Додаток А

Перелік команд мікроконтролера Intel 8051

Таблиця А.1 - Група команд пересилання даних

Назва команди	Мнемокод	Т	Б	Ц	Операція
Пересилання в акумулятор з регістра (n=0..7)	MOV A,Rn	1	1	1	(A)←(Rn)
Пересилання в акумулятор вмісту комірки ВПД	MOV A,dir	3	2	1	(A)←(dir)
Пересилання в акумулятор байта з РПД (i=0,1)	MOV A,@Ri	1	1	1	(A)←((Ri))
Завантаження в акумулятор константи	MOV A,#d	2	2	1	(A)←#d
Пересилання в регістр з акумулятора	MOV Rn,A	1	1	1	(Rn)←(A)
Пересилання в регістр вмісту комірки ВПД	MOV Rn,dir	3	2	2	(Rn)←(dir)
Завантаження в регістр константи	MOV Rn,#d	2	2	1	(Rn)←#d
Пересилання у комірку ВПД вмісту акумулятора	MOV dir,A	3	2	1	(dir)←(A)
Пересилання у комірку ВПД вмісту регістра	MOV dir,Rn	3	2	2	(dir)←(Rn)
Пересилання у комірку ВПД з комірки ВПД	MOV dir,dir	9	3	2	(dir)←(dir)
Пересилка байта з РПД у комірку ВПД	MOV dir,@Ri	3	2	2	(dir)←((Ri))
Пересилання константи у комірку ВПД	MOV dir,#d	7	3	2	(dir)←#d
Пересилання в РПД вмісту акумулятора	MOV @Ri,A	1	1	1	((Ri))←(A)
Пересилання в РПД вмісту комірки ВПД	MOV @Ri, dir	3	2	2	((Ri))←(dir)

Продовження таблиці А.1

Пересилання в РПД константи	MOV @Ri,#d	2	2	1	((Ri))←#d
Завантаження вказівника даних	MOV DPTR,#d16	13	3	2	(DPTR)←#d16
Пересилання байта коду, пов'язаного з DPTR в акумулятор	MOVC A,@A+DPTR	1	1	2	(A)←((A)+(DPTR))
Пересилання байта коду пов'язаного з PC в акумулятор	MOVC A,@A+PC	1	1	2	(PC)←(PC)+1 (A)← ((A)+(PC))
Пересилання байта із ЗПД в акумулятор	MOVX A,@Ri	1	1	2	(A)←((Ri))
Пересилання байта із ЗПД в акумулятор	MOVX A,@DPTR	1	1	2	(A)←((DPTR))
Пересилання з акумулятора в комірку ВПД	MOVX@Ri,A	1	1	2	((Ri))←(A)
Пересилання з акумулятора комірку ЗПД	MOVX @DPTR,A	1	1	2	((DPTR))←(A)
Завантаження комірки ВПД у стек	PUSH dir	3	2	2	(SP)←(SP)+1 ((SP))←(dir)
Вивантаження зі стека в комірку ВПД	POP dir	3	2	2	(dir)←(SP) (SP)←(SP)-1
Обмін акумулятора з регістром	XCH A,Rn	1	1	1	(A)↔(Rn)
Обмін акумулятора з коміркою ВПД	XCH A, dir	3	2	1	(A)↔(dir)
Обмін акумулятора з непрямоадресованою коміркою ВПД	XCH A,@Ri	1	1	1	(A)↔((Ri))
Обмін молодшими тетрадами між непрямоадресованою коміркою ВПД і акумулятором	XCHD A,@Ri	1	1	1	$(A_{0..3})\leftrightarrow((Ri)_{0..3})$

Таблиця А.2 - Команди арифметичних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Додавання акумулятора і регістра (n=0..7)	ADD A,Rn	1	1	1	(A) ← (A)+(Rn)
Додавання акумулятора і комірки ВПД	ADD A, dir	3	2	1	(A) ← (A)+(dir)
Додавання акумулятора і непрямо адресованої комірки ВПД	ADD A,@Ri	1	1	1	(A) ← (A)+((Ri))
Додавання акумулятора і константи	ADD A,#d	2	2	1	(A) ← (A)+#d
Додавання акумулятора і регістра з урахуванням переносу	ADDC A,Rn	1	1	1	(A) ← (A)+(Rn)+(C)
Додавання акумулятора і коміркою ВПД з урахуванням переносу	ADDC A, dir	3	2	1	(A) ← (A)+(dir)+(C)
Додавання акумулятора і непрямо адресованої комірки ВПД з урахуванням переносу	ADDC A,@Ri	1	1	1	(A) ← (A)+((Ri))+(C)
Додавання акумулятора і константи з урахуванням переносу	ADDC A,#d	2	2	1	(A) ← (A)+#d+(C)
Десяткова корекція акумулятора	DA A	1	1	1	Якщо ($A_{0..3}$)>9 або ((AC)=1), то ($A_{0..3}$) ← ($A_{0..3}$)+6, і якщо ($A_{4..7}$)>9 або ((C)=1), то ($A_{4..7}$) ← ($A_{4..7}$)+6

Продовження таблиці А.2

Віднімання від акумулятора регістра і позики	SUBB A,Rn	1	1	1	(A) ← (A)-(C)-(Rn)
Віднімання від акумулятора комірки ВПД і позики	SUBB A, dir	3	2	1	(A) ← (A)-(C)-(dir)
Віднімання від акумулятора непрямо адресованої комірки ВПД і позики	SUBB A,@Ri	1	1	1	(A) ← (A)-(C)-((Ri))
Віднімання від акумулятора константи і позики	SUBB A,#d	2	2	1	(A) ← (A)-(C)-#d
Інкремент акумулятора	INC A	1	1	1	(A) ← (A)+1
Інкремент регістра	INC Rn	1	1	1	(Rn) ← (Rn)+1
Інкремент комірки ВПД	INC dir	3	2	1	(dir) ← (dir)+1
Інкремент непрямо-адресованої комірки ВПД	INC @Ri	1	1	1	(Ri) ← (Ri)+1
Інкремент покажчика даних	INC DPTR	1	1	2	(DPTR) ← (DPTR)+1
Декремент акумулятора	DEC A	1	1	1	(A) ← (A)-1
Декремент регістра	DEC Rn	1	1	1	(Rn) ← (Rn)-1
Декремент комірки ВПД	DEC dir	3	2	1	(dir) ← (dir)-1
Декремент непрямо-адресованої комірки ВПД	DEC @Ri	1	1	1	(Ri) ← (Ri)-1

Продовження таблиці А.2

Множення акумулятора на регістр В	MUL AB	1	1	4	(B)(A) ← (A)*(B)
Ділення акумулятора на регістр В	DIV AB	1	1	4	(A).(B) ← (A)/(B)

Таблиця А.3 - Команди логічних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Логічне І акумулятора і регістра	ANL A,Rn	1	1	1	(A)←(A) ∧ (Rn)
Логічне І акумулятора і комірки ВПД	ANL A, dir	3	2	1	(A)←(A) ∧ (dir)
Логічне І акумулятора і непрямо адресованої комірки ВПД	ANL A,@Ri	1	1	1	(A)←(A) ∧ ((Ri))
Логічне І акумулятора і константи	ANL A,#d	2	2	1	(A)←(A) ∧ #d
Логічне І комірки ВПД і акумулятора	ANL dir,A	3	2	1	(dir)←(dir) ∧ (A)
Логічне І комірки ВПД і константи	ANL dir,#d	7	3	2	(dir)←(dir) ∧ #d
Логічне АБО акумулятора і регістра	ORL A,Rn	1	1	1	(A)←(A) ∨ (Rn)
Логічне АБО акумулятора и комірки ВПД	ORL A, dir	3	2	1	(A)←(A) ∨ (dir)
Логічне АБО акумулятора і непрямоадресованої комірки ВПД	ORL A,@Ri	1	1	1	(A)←(A) ∨ ((Ri))

Продовження таблиці А.3

Логічне АБО акумулятора і константи	ORL A,#d	2	2	1	(A)←(A) ∨ #d
Логічне АБО комірки ВПД і акумулятора	ORL dir,A	3	2	1	(dir)←(dir) ∨ (A)
Логічне АБО комірки ВПД і константи	ORL dir,#d	7	3	2	(dir)←(dir) ∨ #d
Що виключає АБО акумулятора і регістра	XRL A,Rn	1	1	1	(A)←(A) ∀ (Rn)
Що виключає АБО акумулятора і комірки ВПД	XRL A, dir	3	2	1	(A)←(A) ∀ (dir)
Що виключає АБО акумулятора і непрямоадресованої комірки ВПД	XRL A,@Ri	1	1	1	(A)←(A) ∀ ((Ri))
Що виключає АБО акумулятора і константи	XRL A,#d	2	2	1	(A)←(A) ∀ #d
Що виключає АБО комірки ВПД і акумулятора	XRL dir,A	3	2	1	(dir)←(dir) ∀ (A)
Що виключає АБО комірки ВПД і константи	XRL dir,#d	7	3	2	(dir)←(dir) ∀ #d
Очищення акумулятора	CLR A	1	1	1	(A)←0
Інверсія акумулятора	CPL A	1	1	1	$(A) \leftarrow (\overline{A})$
Зрушення акумулятора вліво	RL A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_0)\leftarrow (A_7)$

Продовження таблиці А.3

Зрушення акумулятора вліво через прапор переносу	RLC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$
Зрушення акумулятора вправо	RR A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(A_0)$
Зрушення акумулятора вправо через прапор переносу	RRC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$
Обмін місцями тетрад в Акумуляторі	SWAP A	1	1	1	$(A_{0-3})\leftarrow(A_{4-7})$

Таблиця А.4 - Команди роботи з бітами

Назва команди	Мнемокод	Т	Б	Ц	Операція
Очищення переносу	CLR C	1	1	1	$(C)\leftarrow 0$
Очищення біта	CLR bit	4	2	1	$(bit)\leftarrow 0$
Установка переносу	SETB C	1	1	1	$(C)\leftarrow 1$
Установка біта	SETB bit	4	2	1	$(bit)\leftarrow 1$
Інверсія переносу	CPL C	1	1	1	$(C) \leftarrow (\overline{C})$
Інверсія біта	CPL bit	4	2	1	$(bit) \leftarrow (\overline{bit})$
Логічне І біта і прапору переносу	ANL C,bit	4	2	2	$(C)\leftarrow(C) \wedge (bit)$
Логічне І інверсії біта і переносу	ANL C,/bit	4	2	2	$(C) \leftarrow (C) \wedge (\overline{bit})$

Продовження таблиці А.4

Логічне АБО біта і прапора переносу	ORL C,bit	4	2	2	(C)←(C) ∨ (bit)
Логічне АБО інверсії біта і прапора переносу	ORL C,/bit	4	2	2	$(C) \leftarrow (C) \vee (\overline{bit})$
Пересилання біта в прапор переносу	MOV C,bit	4	2	1	(C)←(bit)
Пересилання прапору переносу в біт	MOV bit,C	4	2	2	(bit)←(C)

Таблиця А.5 - Команди передачі керування

Назва команди	Мнемокод	Т	Б	Ц	Операція
Довгий перехід	LJMP	12	3	2	(PC)← dir 16
Абсолютний перехід всередині сторінки у 2 Кбайта	AJMP dir11	6	2	2	(PC)←(PC)+2 $(PC_{0..10})$← dir 11
Короткий відносний перехід всередині сторінки у 256 байт	SJMP rel	5	2	2	(PC)←(PC)+2 (PC)← (PC)+rel
Непрямий відносний перехід	JMP @A+DPTR	1	1	2	(PC)←(A)+(DPTR)
Перехід, якщо акумулятор дорівнює нулю	JZ rel	5	2	2	(PC)←(PC)+2, якщо (A)=0, то (PC)← (PC)+rel

Продовження таблиці А.5

Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	5	2	2	(PC)←(PC)+2, якщо (A)≠0, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює одиниці	JC rel	5	2	2	(PC)←(PC)+2, якщо (C)=1, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює нулю	JNC rel	5	2	2	(PC)←(PC)+2, якщо (C)=0, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює одиниці	JB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює нулю	JNB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=0, то (PC)← (PC)+rel
Перехід, якщо біт установлено, з наступним скиданням біта	JBC bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (bit)←0 і (PC)← (PC)+rel
Декремент регістра і перехід, якщо він не дорівнює нулю	DJNZ Rn,rel	5	2	2	(PC)←(PC)+2, (Rn)←(Rn)-1, якщо (Rn)≠0, то (PC)← (PC)+rel
Декремент комірки ВПД і перехід, якщо її вміст не дорівнює нулю	DJNZ dir,rel	8	3	2	(PC)←(PC)+2, (dir)←(dir)-1, якщо (dir)≠0, то (PC)← (PC)+rel

Продовження таблиці А.5

Порівняння акумулятора з коміркою ВПД і перехід, якщо вони не дорівнюють одне одному	CJNE A, dir,rel	8	3	2	(PC)←(PC)+3, якщо (A)≠(dir),то (PC)← (PC)+rel, якщо (A)<(dir), то (C)←1, інакше (C)←0
Порівняння акумулятора з константою і перехід, якщо вони не дорівнюють одне одному	CJNE A,#d,rel	10	3	2	(PC)←(PC)+3, якщо (A)≠#d, то (PC)← (PC)+rel, якщо (A)<#d, то (C)←1, інакше (C)←0
Порівняння регістра з константою і перехід, якщо вони не дорівнюють одне одному	CJNE Rn,#d,rel	10	3	2	(PC)←(PC)+3, якщо (Rn)≠#d, то (PC)← (PC)+rel, якщо (Rn)<#d, то (C)←1, інакше (C)←0
Порівняння непрямоадресованої комірки ВПД з константою і перехід, якщо вони не дорівнюють одне одному	CJNE @Ri,#d,rel	10	3	2	(PC)←(PC)+3, якщо ((Ri))≠#d, то (PC)← (PC)+rel, якщо ((Ri))<#d, то (C)←1, інакше (C)←0
Довгий виклик підпрограми	LCALL dir16	12	3	2	(PC)← (PC)+3, (SP)←(SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC)← dir 16

Продовження таблиці А.5

Абсолютний виклик підпрограми в межах сторінки в 2 Кбайта	ACALL dir11	6	2	2	(PC)←(PC)+2, (SP)← (SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC0..10)← dir 11
Повернення з підпрограми	RET	1	1	2	(PC8..15)← ((SP)), (SP)←(SP)-1, (PC0..7)← ((SP)), (SP)←(SP)-1
Повернення з підпрограми оброблення переривання	RETI	1	1	2	($PC_{8..15}$)←((SP)), (SP)←(SP)-1, ($PC_{0..7}$)← ((SP)), (SP)←(SP)-1
Порожня команда	NOP	1	1	1	(PC)← (PC)+1