МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять та самостійної роботи

з дисципліни

"МІКРОПРОЦЕСОРНА ТЕХНІКА В АВТОМАТИЗОВАНИХ СИСТЕМАХ"

для студентів спеціальностей
175 „Інформаційно-вимірювальні технології“,
освітня програма: „Інформаційні системи моніторингу і контролю“;
176 „Мікро- та наносистемна техніка“,
освітня програма: „Мікро- та наноелектронні прилади і пристрої“
першого (бакалаврського) рівня вищої освіти
денної й заочної форм навчання

2025

Методичні вказівки до практичних занять та самостійної роботи з дисципліни "Мікропроцесорна техніка в автоматизованих системах" для студентів спеціальностей 175 „Інформаційно-вимірювальні технології“,освітня програма: „Інформаційні системи моніторингу і контролю“;176 „Мікро- та наносистемна техніка“, освітня програма: „Мікро- та наноелектронні прилади і пристрої“першого (бакалаврського) рівня вищої освіти денної й заочної форм навчання / Укл.: Віталій РЕВА. – Запоріжжя: НУ «Запорізька політехніка», 2025. - 112 с.

Укладач: Віталій РЕВА, доц., канд.фіз.-мат.наук

Рецензент: Ольга ВАСИЛЕНКО, доц., канд.техн.наук

Відповідальний
за випуск: Андрій КОРОТУН, проф., канд.фіз.-мат.наук

Затверджено
на засіданні кафедри ІБ та Н
Протокол №5
від “ 22 “ січня 2025 р.

Рекомендовано до видання
на засіданні МНК ФІБЕК
Протокол №6
від “ 29 “ січня 2025 р.

ЗМІСТ

ВСТУП

Мікропроцесори (МП) і мікропроцесорні системи (МПС) є в даний час найбільш масовими засобами обчислювальної техніки. Після появи перших МП у 1971 р., число нових розробок мікропроцесорних БІС продовжує безупинно збільшуватися.

Досвід розробки різних систем показав, що очікувана від МП універсальність є в значній мірі обмеженою. Різноманітність мікропроцесорних БІС, що виражається в їх різній базовій технології, архітектурі, технічних характеристиках поставило розробників цифрових систем перед складними проблемами, а саме: які особливості цього класу пристроїв і тенденції його розвитку, чим відрізняється використання могутніх мікро-ЕОМ від застосування міні-ЕОМ, як проєктувати системи на основі МП, у чому полягають особливості підходів до розробки програмного забезпечення мікро-ЕОМ і багато інших.

Рішення цих проблем вимагає від розробника систем глибоких знань цифрової обчислювальної техніки (ОТ), включаючи технічні й алгоритмічні питання, представлення особливостей МП БІС, а головне досвіду розробки систем на основі МП.

Курс "Мікропроцесорна техніка" вивчається протягом одного семестру.

В результаті вивчення дисципліни студент повинний знати:

-характеристики і параметри, умовні графічні зображення і принцип дії основних різновидів мікропроцесорів і мікроконтролерів;

-принципи побудови мікро-ЕОМ на основі різних мікропроцесорних комплектів;

-методику аналізу і вибору оптимального технічного рішення;

-системи команд мікропроцесорів і правила складання програмного забезпечення РЕА на основі мікропроцесорних систем;

-сучасний стан мікропроцесорної і комп'ютерної техніки і перспективи їхнього розвитку.

ПІДГОТОВКА ДО ПРОВЕДЕННЯ ПРАКТИЧНОЇ РОБОТИ ТА ОФОРМЛЕННЯ ЗВІТУ

Практична робота включає самостійне опрацювання теоретичного матеріалу, вивчення документації з апаратних засобів, методик проведення розрахунків, програмування і моделювання, обробку й інтерпретацію результатів. При виконанні практичних робіт необхідно:

- підготуватися до експрес-опитування з теоретичного матеріалу, необхідного для виконання роботи (щоб отримати допуск до лабораторної роботи);
- підготувати і оформити план виконання практичної роботи;
- після виконання практичної роботи провести необхідні розрахунки і оформити звіт, який повинен містити:

а) титульний аркуш (див. Додаток А);
б) мету роботи;
в) опис словами;
г) алгоритм роботи у вигляді блок схеми;
д) необхідні математичні розрахунки роботи пристрою;
е) текст розробленої програми;
ж) основні результати моделювання пристрою;
з) стислі висновки по роботі, в яких:

- вказати, що досліджувалось;
- вказати методику (чи метод) дослідження;
- вказати особливості алгоритму роботи пристрою;
- визначити характер роботи пристрою при моделюванні;
- відзначити подібність і відмінність результатів моделювання від теоретичних даних;
- здати звіт з виконаної практичної роботи та відповісти на запитання викладача (отримати залік за виконану роботу).

1. ПРАКТИЧНА РОБОТА №1 «ПРЕДСТАВЛЕННЯ ДАННИХ У МПС»

Мета – Мікропроцесорна система, формат представлення базових даних у МПС.

1.1 Основні визначення

Мікропроцесор (МП) − це мікросхема або сукупність невеликого числа мікросхем (відповідно один або декілька кристалів ВІС), що виконує над данними арифметичні та логічні операції і здійснює програмне керування обчислювальним процесом.

Мікропроцесорні засоби випускаються промисловістю у вигляді наборів мікросхем (chip-set), сумісних за рівнями напруги живлення, сигналів і представлення інформації, що включають МП, мікросхеми оперативної та постійної пам'яті, управління вводом/виводом, генератором тактових сигналів та ін.

Мікропроцессори (мікропроцесорні засоби) служать основою для створення різних універсальних та спеціалізованих мікро-ЕОМ, мікропроцесорних інформаційно-керуючих систем, програмованих мікроконтролерів, різноманітних мікропроцесорних приладів та пристроїв контролю, керування та обробки даних.

Мікро-ЕОМ або мікрокомп'ютером називають пристрій обробки даних, що містить один або кілька мікропроцесорів, ВІС постійної та оперативної пам'яті, ВІС керування введенням та виходом інформації та деякими іншими схемами. Мікрокомп'ютер такого складу іноді називають "голим" через відсутність у нього периферійних пристроїв (зовнішніх ЗП та пристроїв введення та виведення інформації). Мікрокомп'ютери в такій конфігурації часто застосовуються як вбудовані в різні верстати, машини, технологічні процеси керуючих пристроїв (контролерів).

Мікрокомп'ютери широкого призначення, що використовуються для виконання обчислювальних робіт, управління складними технологічними процесами, оснащені необхідними периферійними пристроями (дисплеями, друкованими пристроями, ЗП на гнучких дисках, аналого-цифровими та цифроаналоговими перетворювачами тощо).

Мікропроцесорною системою (МПС) зазвичай називають спеціалізовану інформаційну або управляючу систему, побудовану на основі мікропроцесорних засобів.

Мікрокомп'ютер з невеликими обчислювальними ресурсами і спрощеною системою команд, орієнтований не на виробництво обчислень, а на виконання процедур логічного управління індивідуальним обладнанням, називають програмованим мікроконтроллером або просто **мікроконтроллером**.

Логічна організація (архітектура) мікропроцесорів (мікропроцесорних засобів) орієнтована на досягнення універсальності застосування, високої продуктивності і технологічності.

Універсальність МП (мікропроцесорних засобів) визначається можливістю їх різноманітного застосування і забезпечуються програмним управлінням МП, що дозволяє виконувати програмне налаштування МП на виконання певної функції, магістрально-модульний принцип побудови, а також спеціальними апаратурно-логічними засобами: надоперативна реєстрова пам'ять, багаторівневою системою переривань, прямим доступом до пам'яті, програмно-налаштовуваними засобами керування вводом/виводом і т.д.

Відносно висока продуктивність МП досягається використанням для їх побудови швидкодіючих великих і надвеликих інтегральних електронних схем і спеціальних архітектурних рішень, таких як стекова пам'ять, різноманітними методами адресації, гнучкою системою команд (або мікрокоманд) і ін.

Технологічність мікропроцесорних засобів забезпечується модульним принципом конструювання, який передбачає реалізацію цих засобів у вигляді набору функціонально закінчених ВІС, що просто об'єднується у відповідних обчислювальних пристроях, машинах і комплексах.

Висока універсальність і гнучкість МП, яка досягається завдяки програмному управлінню, низька вартість, невеликі розміри, підвищена надійність, можливість вбудовування мікропроцесорних засобів в приладах, машинах і технологічних процесах, забезпечує мікропроцесорам виключно широке застосування в різноманітних пристроях і системах керування та обробки данних.

Використання мікропроцесорів призводить до зміни характеру проєктної роботи розробника пристроїв і систем автоматики: у багатьох випадках проєктування схем замінюється розробкою програм налаштування мікропроцесорної апаратури на виконання визначених функцій.

При розробці засобів мікропроцессорної техніки знайдено подальший, більш глибокий розвиток наступних принципів: модульність; магістральність; мікропрограмованість; регулярність структури.

Модульна організація передбачає побудову систем на основі нових наборів модулів конструктивно, функціонально та електрично закінчених пристроїв, що дозволяють самостійно або в сукупностіності з іншими модулями вирішити обчислювальні чи управлінські задачі певної класу.

Модульний підхід сприяє стандартизації елементів все більш високих рівнів і скорочення витрат на проєктування систем, а також спрощує нарощування потужності і реконфігурацію систем, віддаляє час морального старіння технічних засобів.

Багатофункціональність (універсальність) і спеціалізація модулю − це два протилежні якісні показники, що надаються системі у процесі компромісних рішень, для різноманітних класів систем, виходячи із забезпечення структури системи характеру виконуваної задачі.

Доцільно створювати системи у вигляді сукупності багато-функціональних і спеціалізованих модулів, проблемно і функціонально орієнтованих у рамках певних класів задач, алгорітмов, функцій.

Магістральний − спосіб обміну інформацією всередині модулів і між модулями за допомогою упорядкованих зв'язків (на відміну від довільних зв'язків, який реалізує принцип «кожен з кожним»), які мінімізують число зв'язків. Обмін здійснюється за допомогою спільних магістралей (шин), об'єднуючих вхідні та вихідні лінії окремих елементів і модулів. Магістральність − один з способів забезпечення регулярності структури системи та стандартизації інтерфейсів. З технічної точки зору − це спосіб обміну в вигляді створення спеціальних двонаправлених буферних каскадів з трьома стійкими станами і використанням тимчасового мультиплексування каналів обміну [5, 6].

Мікропрограмованість − це спосіб організації управління, що дозволяє здійснити проблемно орієнтовану систему. Мікропрограмованість підвищує гнучкість пристрою (за рахунок можливості зміни мікропрограм), збільшує регулярність їх структури (за рахунок широкого використання матричних структур типу пам'яті), підвищує надійність пристрою (за рахунок застосування серійно освоєних ВІС пам'ять), спрощує контроль функціонування пристрою (за рахунок того, що контроль блоку мікропрограмного керування зводиться, по суті, до контролю вмісту ЗП).

Регулярність структури − передбачає закономірну повторюваність ряду елементів структури та зв'язків між ними; регулярність структури системи слід розглядати на різних рівнях її організації.

В цілому можна відзначити, що МП як програмований цифровий пристрій обробки інформації характеризується наступними показниками і зв'язками їх з внутрішньою та зовнішньою структурою МП: розрядність; ємність адресної пам'яті; універсальність (спеціалізація); число внутрішніх реєстрів; магістральность; мікропрограмне керування; можливість і кількість рівнів переривань; наявність стекової організації пам'яті та кількість стекових реєстрів: наявність і склад резидентного та крос-програмного забезпечення.

Мікропроцесорна автоматична система (МПАС) − це автоматизована система з вбудованими в неї засобами мікропроцесорної техніки (МТ).

1.2 Представлення данних у МПС

Будь-яке дискретне повідомлення в цифровій техніці представляється у вигляді цифри і символу. Спосіб запису чисел знаками називається системою числення. Є два види: позиційні і непозиційні. Прикладом непозиційної системи числення є римська система (напр. 30 = XXX). У цифровій техніці застосовуються позиційні системи числення, в якій значення кожної цифри числа залежить від її позиції. Кількість цифр в числі визначає назву для системи числення.

Двійкова система числення (Binary).

Основою двійкової системи числення є **q = 2**. Число в такій системі числення утворено із множини цифр **{0,1}**. Базова одиниця

комп'ютерних даних називається біт. Слово **біт** є скороченням англійського виразу **binart digit**. т.т. двійкова цифра. Таким чином двійкове число – це послідовність біт.

Переведення цілих десяткових чисел в двійкову систему числення.

Для переведення цілого десятинного числа в систему числення з основою **q = 2** необхідно таке число послідовно ділити на число 2 до отримання цілого залишку, меншого ніж 2. Результат в двійковій системі числення буде мати вид упорядкованої послідовності залишків від в порядку, зворотному їх отримання (старшу цифру числа дає останній залишок, а молодшу - перший).

Приклад 1.

Перевести з десяткової в двійкову систему числення число 137:

137 → ? bin

137	2							Результат
136	68	2						137 decimal → 10001001
1	68	34	2					
	0	34	17	2				Запис на асемблері робиться за допомогою дескрипторів:
		0	16	8	2			
			1	8	4	2		D – decimal
				0	4	2	2	B – binary
					0	2	1	137d → 10001001b
						0		

Переведення правильної десяткової дробі в двійкову систему числення.

Для переведення дійсного десяткового числа (правильна дріб) в двійкову систему числення необхідно дане число (тільки мантису) послідовно множити на число 2 до тих пір, доки в мантисі не отримаємо чистий нуль або необхідну кількість розрядів. При множенні отримане ціле значення не враховується . Проте вони послідовно зараховуються в результат. Отримаємо упорядковану послідовність цілих дійкових чисел.

Приклад 2: 0.5d → ? bin

	.5
x	2
1	.0

05d → 0.1b

Приклад 3: 0.703125d → ? bin

	0.703125
x	2
1	.406250
x	2
0	.812500
x	2
1	.625000
x	2
1	.250000
x	2
0	.500000
x	2
1	.000000

0.703125d → 0.101101bin

Переведення змішаних десяткових чисел в двійкову систему числення.

Дійсне змішане число в двійковій системі числення утворюється, як і в десятковій, шляхом поєднання (конкатенації) цілої і дробової частини.

117.25d → 1110101.01b

Переведення двійкових чисел в десяткову систему числення.

Двійкова система числення являється позиційною системою числення з основою 2., тому для неї справедлива формула з якою ми ознайомилися на початку. Т.т. ми можемо розкласти двійкове число по ступеням двійки.

Приклад 4: 1110101.01b → ? dec

Число	1	1	1	0	1	0	1	.	0	1
Позиція	6	5	4	3	2	1	0		-1	-2

$1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2} = 117.25$

1110101.01b → 117.25d

Шіснадцяткова система числення (Hexadecimal).

Дана система числення **(hex)** визначена множиною **{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,}** і дозволяє суттєво зжати двійкову інформацію, полегшуючи тим самим її видачу в різних лістінгах. Шіснадцяткова система числення - це базова система при спілкуванні людини і комп'ютера і в операційній системі MS DOS і Windows зокрема.

Переведення десяткових чисел в шіснадцяткову систему числення.

Щоб перевести число із десяткової системи числення в шіснадцяткову, можна скористатися загальним правилом, т.т. послідовно ділити число на 16, поки не отримаємо залишок менше, ніж 16 (для цілих чисел), або помножити на 16 (для правильної дробі). Це не зовсім зручно, при діленні або множенні легко помилитися. Тому прийнято спочатку перевести число в двійкову систему числення, так як ми робили раніше, а тоді кожну двійкову тетраду представити шіснадцятковою цифрою. Тетради можна отримати, починаючи від десяткової точки (вліво – для цілої частини числа, вправо – для дробової), додаючи, якщо потрібно незначущі нулі.

Приклад 5: перевести число 137 із десяткової в шіснадцяткову систему числення, використовуючи двійкову систему числення як проміжну.

137 → ? hex

137d → 1000 1001b
8 9 h
137d → 89h

Приклад 6:
0.5d → ? hex
0.5d → 0.1b = 0.1000b
0. 8h

В даному прикладі ми, для того щоб отримати тетраду, двійкове число розширили незначущими нулями.

0.5d → 0.8h

Приклад 7:
0.703125d → ? hex
0.703125d → 0.1011 01b = 0.1011 0100b
0. B 4h

В даному випадку потрібно зробити розширення до двійкової тетради:

0.703125d → 0.B4h

Приклад 8:

0.05d → ? hex
0.05d → 0.000011(0011)b = 0.0000 1100 1100 (1100)b
0. 0 C C (C)h

В даному прикладі, ми розписали періодичний дріб так, щоб отримати необхідні двійкові тетради. Маємо періодичний шіснацяткову дріб:

0.05d → 0.0CC(C)h

Приклад 9:

117.25d → ? hex
117.25d → 111 0101.01b = 0111 0101. 0100b
7 5. 4h

В даному прикладі прийшлось розширити до тетради і дробову і цілу частину двійкового числа. В результаті отримаємо:

117.25d → 75.4h

Переведення цілих шіснацяткових чисел в десяткову систему числення.

Оскільки між шіснацятковою і двійковою системами числення існує дуже тісний зв'язок, то зазвичай будь-яке шіснацяткове число в десяткове переводять в два етапи, використовуючи наступну схему:

hex → bin → dec

Приклад 10:

35DDh	→	? dec
3000h	→	12 288
500h	→	280
D0h	→	208
Dh	→	13

Знову все складаємо, отримуємо:

35DDh → 13 789d

Вісімкова система числення (Octal).

За своїми ідеями і принципам вона повністю ідентична шіснадцятковій системі числення, але зжимає тріаду (трійку) бінарних розрядів. Ця система числення визначена на множені цифр **{0,1,2,3,4,5,6,7}**.

Приклад 11:
117.25d → ? oct
117.25d → 111 0101.01b = 001 110 101. 010b
1 6 5. 2o
В даному прикладі нам довелось розширити до тріади десяткову і цілу частини двійкового числа. В результаті отримаємо:
117.25d → 165.2o

Переведення з вісімкової в двійкову системи числення

Для переведення з вісімкової в двійкову системи числення необхідно кожну цифру числа представити трьома двійковими цифрами (тріадою) так, щоб числове значення кожної тріади дорівнювало відповіднії 8-ковій цифрі

Переведення з двійкової в вісімкову системи числення

Для переведення зліва і праворуч від коми виділяються групи по 3 цифри. При необхідності крайні доповнюються нулями до повних тріад.

Арифметичні операції

Додавання

Виконується так само як і в десятковій системі числення

Віднімання

Віднімання чисел в двійковій і вісімковій системах числення виконується за тими ж правилами що і в десятковій. Якщо від'ємник більше зменшуваного, різниця визначається між більшим і меншим числом, і перед нею ставиться знак мінус

Множення

Операція множення виконується точно також як і в десятковій системі числення.

Прямий код

Використовується при виконанні множення і ділення чисел, а інші коди для заміни віднімання додаванням.

0,011 число позитивне

1,011 число негативне

При виконанні операції множення або ділення двох двійкових дробів знакові розряди складаються незалежно від десяткових частин

Зворотній код

Застосовується для заміни операції віднімання складанням

Для позитивних чисел: зображення правильного двійкового дробу однаково в зворотному і прямому коді. Для запису негативного правильного двійкового дробу в зворотному коді необхідно замінити нулі одиницями і навпаки, а зліва від коми замість -0 поставити1. Тобто -0,0101 = 1,1010

Необхідно враховувати:

− Всі цифри доданків, включаючи і цифри знакових розрядів, розташовані зліва від коми беруть участь у складанні як розряди єдиного числа

− При переповненні, коли зліва від коми в результаті складання з'являються дві цифри, ліва крайня цифра переноситься і складається з молодшим розрядом дробової частини, а решта цифр зліва від коми визначає знак результату

− Якщо число розрядів десяткової частини негативного правильного двійкового дробу менше числа розрядів дробової частини іншого доданка, то перед тим як перевести негативний дріб в зворотний код необхідно доповнити його справа нулями до рівності розрядів другого доданка

Якщо в знаковому розряді числа в зворотному коді стоїть 1, то для переходу до звичайного запису треба в дробовій частині одиниці замінити нулями, а нулі - одиницями, а зліва від коми записати -0

Додатковий код

Так само як і зворотний застосовується для заміни віднімання складанням

При цьому: зображення позитивного правильного двійкового дробу однаково в прямому, зворотному і додатковому кодах.

Для переведення негативного дробу: Необхідно нулі замінити одиницями, а 1 нулями. До молодшого розряду додати одиницю, потім зліва від коми поставити 1.

Необхідно пам'ятати:

− Всі цифри доданків, включаючи і цифри знакових розрядів, розташованих зліва від коми беруть участь у складанні як розряди єдиного числа

− При переповненні, коли зліва від коми в результаті складання з'являються дві цифри, ліва крайня цифра відкидається, а решта цифр зліва від коми визначає знак результату

− Перед тим як перевести негативний дріб в зворотний код необхідно доповнити його справа нулями до рівності розрядів другого доданка

− Якщо в результаті складання зліва від коми вийшов 1, то число негативне, якщо 0, то позитивне (відповідно переводити нічого не потрібно)

Формат представлення базових даних

Цілі числа.

В Асемблері програміст сам вирішує, які дані (цілі або дійсні) він помістить в ту чи іншу комірку пам'яті. Наприклад, просте арифметичне дане, під який відведено 32 або 64 біта, може бути як цілим, так і дійсним.

Цілі числа без знака.

Для 16-розрядної архітектури цілі числа без знака - це позитивні числа, які можуть займати 8, 16, або 32 біта пам'яті. Вважається, що біт 0 - молодший біт (для зручності будемо вважати, що він розташований крайнім справа). Старший біт - 7 (15 або 31) - для зручності розташуємо крайнім зліва. Для без знакових даних всі біти вважаються інформаційними.

Таблиця 1.1 – Формат представлення для 32-бітного без знакового числа

Інформаційне поле {0,1}																		
31	…	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Старший	Біти																	Молодший

Приклад12:

Нехай є десяткове число 40000. В якому форматі його можна представити? Т. т.

Скільки йому потрібно біт і яке буде його внутрішнє (машинне) представлення?

Для початку потрібно перевести це число в двійкову систему числення. Для цього скористаємося стандартним калькулятором Windows. отримуємо:

40000d → 1001 1100 0100 0000 → 9C40h

За кількістю отриманих біт видно, що для зберігання такого числа нам знадобиться мінімум 16 двійкових розрядів.

Можна це число (з запасом) розмістити і в 32-х бітах:

40000d → 0000 0000 0000 0000 1001 1100 0100 0000 → 0000 9C40h

Цілі зі знаком.

Цілі знакові дані можуть бути як позитивними, так і негативними, включаючи нуль. Також вони в два рази менше за діапазоном допустимих значень. Це відбувається тому, що не вся область біт, відводиться під інформацію, як це було з беззнаковими даними. Старший біт завжди відводитися під знак. Він називається біт S - Signum (знак - лат.):

S = 0 - для позитивних чисел;

S = 1 - для негативних чисел;

Таблиця 1.2 – Формат представлення для 32-бітного знакового числа

Інформаційне поле {0,1}																	
31	…	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Старший	Біти																	

Тому під інформацію відводиться на один біт менше.

Розглянемо задачу: що вийде, якщо ми вихідне десяткове число 12345 позбавимо останньої цифри 5? Отримаємо число 1234. В програмуванні це називається зміщенням вправо на один десятковий розряд. І наше число вийде в 10 разів менше! Цілком аналогічно відбувається з двійковим числом, якщо ми його змістимо на один, але двійковий розряд, вправо. Початкове число зменшиться в 2 рази! Наприклад, для позитивного знакового даного, якому відведено один байт інформації, виходить так:

0111 1111b → 7Fh → 127d

Для негативних чисел беруть за основу представлення негативних чисел в додатковому коді, суть якого полягає в наступному:

1)Беремо модуль негативного числа (т.т. Позитивне) і переводимо dec → hex;

2)Робимо інверсію hex-коду (т.т. Нулі стають одиницями, а одиниці - нулями);

3)До отриманого hex-коду додаємо 1.

Приклад13:

Нехай є десяткове число -1607. В якому форматі його можна представити? Т.т. Скільки йому потрібно біт і яке буде його внутрішнє (машинне) представлення?

Рішення.

Наше число негативне, значить, його потрібно представити в додатковому коді:

а) |-1607|=1607 → 647h → 0110 0100 0111b

Тобто, нашому числу досить 12 біт. Але такого формату немає, тому беремо, наприклад, 16 біт, додаючи нулі:

0000 0110 0100 0111b.

б) Робимо інверсію нашого коду: 0000 0110 0100 0111b.

Отримуємо: 1111 1001 1011 1000b.

в) До отриманого додаємо 1

Отримуємо: 1111 1001 1011 1001

Це і є машинне представлення нашого числа -1607 або в шістнадцятковій системі числення: F9B9h. У пам'яті комп'ютера це число буде зберігається задом-наперед, т.т. B9F9h.

Приклад.

Який вийде машинний результат, якщо ми захочемо розмістити наше число -1607 в 32 бітах? Відповідь: FFFF F9B9h

Число обов'язково повинно бути в знаковому форматі. А якщо воно буде інтерпретуватися як без знакове, отримаємо десяткове значення:

FFFF F9B9hc → 4294965689d

Тут ми вийшли на ще одну дуже важливу обставину: двійковий код може бути один і той же для різних чисел! Все залежить від програміста, як він цей код інтерпретує. Тому для Ассемблера немає директив опису знакових і беззнакових даних.

3. Дійсні числа.

Дійсні числа можуть бути типу float (single), double або long double (extended). Від платформи вони не залежать. Ці дійсні числа обробляє співпроцесор. Внутрішнє (машинне) представлення цих чисел досить складне:

Таблиця 1.3 – Формат дійсного числа

S	Характеристика	Нормалізована мантиса

Біт S − знак числа.

Характеристика = Зсув ± Порядок

Зсув − число, що дорівнює половині максимально можливого, яке може поміститися в полі Характеристика. Таким чином, економлять місце і час, так як не потрібно виділяти розряд для знака порядку і робити додатковий негативний код для негативних порядків. Розмір простору, яке ПК використовує для зберігання Характеристики і мантиса, встановлений стандартом IEEE 724-1985 standard for floating point numbers і підтримується всіма сучасними комп'ютерними архітектурами.

Для отримання внутрішнього представлення дійсного числа потрібно в будь-якому випадку перевести його в двійковий формат.

Машинні формати дійсних чисел.

Як уже зазначалося, дійсні базові типи можуть бути типу float (single), double або long double (extended). Всі вони мають для зберігання комірки різної довжини. Звідси і різний діапазон представлення для різних типів дійсних чисел, хоча ідея зберігання майже одна і та ж.

Формат float (single).

Число це зберігається в комірках довжиною 32 біта, які розподіляються наступним чином:

Представлення дійсних чисел в двійковому нормалізованому вигляді

$\pm 1.m_2 * 2^p$,

де m_2 – двійкова мантиса числа, p – порядок двійкового числа.

Розглянемо декілька прикладів:

1) $\pm 1.0d \rightarrow \pm\ 1.0b * 2^0$

2) $\pm 0.5d \rightarrow \pm 0.1b$, в нормалізованому виді $\pm 1.0b * 2^{-1}$

3) $\pm 0.703125d \rightarrow \pm 0.101101b$, в нормалізованому виді $\pm 1.01101b * 2^{-1}$

4) $\pm 0.05d \rightarrow \pm 0.000011b$, в нормалізованому виді $\pm 1.1(0011)b * 2^{-5}$

5) $\pm 117.25d \rightarrow \pm 1110101.01b$, в нормалізованому виді $\pm 1.11010101b * 2^6$

Проаналізувавши приклади, можна помітити, що десяткова точка «плаває», щоб забезпечити нормалізацію. То зміщується вліво і порядок отримує знак плюс, то - вправо, тоді у порядку знак мінус. Звідси і термін - плаваюча точка (floating point).

Таблиця 1.4 – Формат дійсного 32-бітного числа.

				← скритий (невидимий людині) розряд				
S	Характеристика			Нормалізована мантиса				
31	30	…	23	22	21	…	1	0
Біти								

На характеристику виділяється 8 біт (розряди 23-30). Максимально можливе шістнадцяткове число, яке можна розмістити в цих бітах, так само FFh, а його половина за визначенням - це Зсув = 7Fh. Оскільки на мантису залишається всього 23 біта (розряди 0- 22), а ми знаємо, що мантиса завжди нормалізована, виникає цілком резонне питання: навіщо зберігати найпершу одиницю мантиси? Нехай комп'ютер сам її відновлює ... Таким чином, ця найперша одиниця в інформаційні розряди мантиси не потрапляє (прихований розряд), а на мантису в результаті відводитися не 23 і 24 біта. Таке представлення дозволяє вірно відображати 7-8 десяткових цифр.

Формат double

Число зберігається в комірці розміром 64 біта:

Таблиця 1.5 – Формат дійсного 64-бітного числа.

				← скритий (невидимий людині) розряд				
S	Характеристика			Нормалізована мантиса				
63	62	…	52	51	50	…	1	0
Біти								

Формат double

Число зберігається в комірці розміром 80 біт:

Таблиця 1.6 – Формат дійсного 80-бітного числа.

				← скритий (невидимий людині) розряд				
S	Характеристика			Нормалізована мантиса				
79	78	…	64	63	62	…	1	0
Біти								

Контрольні питання.

1. В якому вигляді зберігаються символи в комп'ютері?
2. Якими видами цілих чисел оперує комп'ютер?
3. Які типи дійсних чисел ви знаєте?

Таблиця 1.7 – Відповідність систем числення

Десяткова	Двійкова	Вісімкова	Шіснадцяткова
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Арифметичні операції в позиційних системах числення

Правила виконання арифметичних операцій в десятковій системі добре відомі – це додавання, віднімання, множення і ділення. Ці правила застосовні і до всіх інших позиційних систем числення. Тільки таблицями додавання і множення треба користуватися особливими для кожної системи.

Таблиця 1.8 – Додавання в двійковій системі

+	0	1
0	0	1
1	1	10

Таблиця 1.9 – Додавання в вісімковій системі числення

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Таблиці додавання в будь-який позиційній системі числення легко скласти, використовуючи правило врахування: Якщо сума цифр, які складаються більше або дорівнює основі системи числення, то одиниця переноситься в наступний зліва розряд.

Таблиця 1.10 – Додавання в шіснадцятковій системі числення

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Приклади:

Додавання чисел 10101b + 1011b → ?

$$\begin{array}{cccccc} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & \\ & 1 & 0 & 1 & 0 & 1 \\ + & & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Додавання чисел 10b + 1011b → ?

$$\begin{array}{cccccc} & & & {\scriptstyle 1} & & \\ & & 1 & 0 & 1 & 1 \\ & + & & & 1 & 0 \\ \hline & & 1 & 1 & 0 & 0 \end{array}$$

Додавання чисел 17o + 6o → ?

$$\begin{array}{ccc} & {\scriptstyle 1} & \\ & 1 & 7 \\ + & 1 & 6 \\ \hline \end{array}$$

$2_{(2)}\ \ 5_{(1)}$

1) 7 + 6 = 13 = 8+5

2) 1 + 1 = 2

Додавання чисел 43o + 126o → ?

```
  1
      4   3
+ 1   2   6
-------------
  1   7   1(1)
```

1)3 + 6 = 9 = 8 + 1

Віднімання здійснюється за тими самими правилами, що і в десятковій системі числення.

При відніманні від меншого числа більшого проводиться позика зі старшого розряду.

Приклад: Віднімаємо одиницю з чисел 10b, 10o і 10h

Двійкова 10b – 1b

```
  1
  1   0
_     1
-------
      1(1)
```

1) 2 – 1 = 1

Вісімкова 10o – 1b

```
  1
  1   0
_     1
-------
      7(1)
```

1) 8 – 1 = 7

Шіснадцяткова 10b – 1b

```
  1
  1   0
_     1
-------
      F(1)
```

1) 16 – 1 = 15 = F

Віднімаємо одиницю з чисел 100b, 100o і 100h

Двійкова 10b – 1b

```
  1
  1   0   0
_         1
-----------
    1(2)  1(1)
```

1) 2 – 1 = 1
2) 1 – 0 = 1

Вісімкова 10o – 1o

$$\begin{array}{r} {\scriptstyle 1} \\ 1\quad 0\quad 0 \\ -\quad\quad 1 \\ \hline 7_{(2)}\quad 7_{(1)} \end{array}$$

1) 8 – 1 = 7
2) 7 – 0 = 7

Шіснадцяткова 10h – 1h

$$\begin{array}{r} {\scriptstyle 1} \\ 1\quad 0\quad 0 \\ -\quad\quad 1 \\ \hline F_{(2)}\quad F_{(1)} \end{array}$$

1) 16 – 1 = 15 = F
2) 15 – 0 = 15 = F

Множення

Виконуючи множення багатозначних чисел в різних позиційних системах числення, можна використовувати звичайний алгоритм перемноження чисел в стовпчик, але при цьому результати перемноження і складання однозначних чисел необхідно запозичувати з відповідних даній системі таблиць множення і додавання.

Таблиця 1.11 – Множення в двійковій системі.

×	0	1
0	0	0
1	0	1

Таблиця 1.12 – Множення в вісімковій системі числення

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Таблиця 1.13 – Множення в шіснадцятковій системі числення

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Приклади: Помножити числа 5 і 6
Двійкова система числення

$$
\begin{array}{cccccc}
 & & & 1 & 0 & 1 \\
\times & & & & & \\
 & & & 1 & 1 & 0 \\
\hline
 & & & 0 & 0 & 0 \\
 & & 1 & 0 & 1 & \\
 & 1 & 0 & 1 & & \\
\hline
 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

Перемножимо числа 115 и 51 (вісімкова системи числення)

$$
\begin{array}{cccccc}
 & & & 1 & 6 & 3 \\
\times & & & & & \\
 & & & & 6 & 3 \\
\hline
 & & & 5 & 3 & 1 \\
 & 1 & 2 & 6 & 2 & \\
\hline
 & 1 & 3 & 3 & 5 & 1
\end{array}
$$

1.3 Завдання для практичної роботи
Відповісти на наступні питання:
1. Дати визначення наступним термінам:
   а. Мікропроцесор
   б. Мікропроцесорні засоби
   в. Мікропроцесорна система

г. Мікроконтролери

д. Мікропроцесорна автоматична система

2. Поясніть відмінності між мікропроцесором і мікроконтролером.

Таблиця 1.13 – Завдання для практичної роботи

Варіант	Завдання до роботи
	1. Перевести дане число з десяткової системи числення в двійкову, вісімкову і шістнадцятирічну системи числення; $X_2 \rightarrow X_8$, $X_2 \rightarrow X_{16}$ без мат. обчислень. 2. Перевести дане число в десяткову систему числення. 3. Скласти числа. 4. Виконати віднімання. 5. Виконати множення. Примітка. У завданні 1В отримати п'ять знаків після коми в двійковому уявленні.
1	1. а) $305_{(10)}$ Б) $153{,}25_{(10)}$; В) $248{,}46_{(10)}$ 2. а) $1100111_{(2)}$; б) $100110{,}101_{(2)}$; В) $671{,}24_{(8)}$; г) $41A{,}6_{(16)}$. 3. а) $10000011_{(2)}+1000011_{(2)}$; б) $110010{,}101_{(2)}+1011010011{,}01_{(2)}$; 4. а) $100111001_{(2)}-110110_{(2)}$; б) $1111001110_{(2)}-111011010_{(2)}$. 5. $1100110_{(2)} \times 1011010_{(2)}$.
2	1. а) $164_{(10)}$; б) $712{,}25_{(10)}$; в) $11{,}89_{(10)}$ 2. а) $1001111_{(2)}$; б) $1100111{,}01_{(2)}$;в) $413{,}41_{(8)}$; г) $118{,}8C_{(16)}$. 3. а) $1100001100_{(2)}+1100011001_{(2)}$; б) $111111111{,}001_{(2)}+1111111110{,}0101_{(2)}$. 4. а) $1001101100_{(2)}-1000010111_{(2)}$; б)$1101100110_{(2)}$-111000010(2). 5. $10001_{(2)} \times 1010_{(2)}$.
3	1. а) $273_{(10)}$; б) $97{,}5_{(10)}$; в) $53{,}74_{(10)}$ 2. а) $110000_{(2)}$; б) $1001101{,}00011_{(2)}$; в) $1017{,}2_{(8)}$; г) $111{,}B_{(16)}$. 3. а) $1110001000_{(2)}+110100100_{(2)}$; б) $1001001{,}101_{(2)}+1111000{,}11_{(2)}$. 4. а) $1010111001_{(2)}-1010001011_{(2)}$; б) $1110101011_{(2)}$-$100111000_{(2)}$. 5. $10110_{(2)} \times 10010_{(2)}$.

Варіант	Завдання до роботи
4	1. а) $105_{(10)}$; б) $377,5_{(10)}$; в) $87,27_{(10)}$ 2. а) $11001001_{(2)}$; б) $11110110,01_{(2)}$; в) $112,04_{(8)}$; г) $334,A_{(16)}$. 3. а) $101000011_{(2)}+110101010_{(2)}$; б) $111010,010_{(2)}+1011011,11_{(2)}$. 4. а) $1111111000_{(2)}-100010011_{(2)}$; б) $1111101110_{(2)}-11100110_{(2)}$. 5. $10111_{(2)} \times 10100_{(2)}$.
5	1. а) $500_{(10)}$; б) $1017,25_{(10)}$; в) $123,72_{(10)}$ 2. а) $11010101_{(2)}$; б) $11110001,011011_{(2)}$;в) $1347,17_{(8)}$; г) $155,6C_{(16)}$. 3. а) $1000101101_{(2)}+1100000010_{(2)}$; б) $1111011,01_{(2)}+11100,11_{(2)}$. 4. а) $1101111100_{(2)}-100100010_{(2)}$; б) $1011010110_{(2)}-1011001110_{(2)}$. 5. $11011_{(2)} \times 10011_{(2)}$
6	1. а) $218_{(10)}$; б) $176,25_{(10)}$;в) $253,04_{(10)}$ 2. а) $1110100_{(2)}$; б) $100011,01_{(2)}$; в) $1665,3_{(8)}$; г) $FA,7_{(16)}$. 3. а) $11100000_{(2)}+1100000000_{(2)}$; б) $1101011,01_{(2)}+111111,1101_{(2)}$. 4. а) $10110010_{(2)}-1010001_{(2)}$; б) $1101000000_{(2)}-10000000_{(2)}$ 5. $10000_{(2)} \times 1110_{(2)}$
7	1. а) $306_{(10)}$; б) $218,5_{(10)}$; в) $318,87_{(10)}$ 2. а) $1100111_{(2)}$; б) $101111,01_{(2)}$; в) $465,3_{(8)}$; г) $252,38_{(16)}$. 3. а) $1000001101_{(2)}+1100101000_{(2)}$; б) $1010011,11_{(2)}+1000,1_{(2)}$. 4. а) $1101000101_{(2)}-111111000_{(2)}$; б) $11110101_{(2)}-110100_{(2)}$ 5. $11101_{(2)} \times 1010_{(2)}$.
8	1. а) $113_{(10)}$; б) $128,25_{(10)}$; в) $314,71_{(10)}$ 2. а) $110001_{(2)}$; б) $10011100,111_{(2)}$; г) $704,6_{(8)}$; д) $367,38_{(16)}$. 3. а) $10101100_{(2)}+111110010_{(2)}$; б) $1110111010,10011_{(2)}+1011010011,001_{(2)}$ 4. а) $1010110010_{(2)}-1000000000_{(2)}$; б) $1111100110_{(2)}-10101111_{(2)}$ 5. $10101_{(2)} \times 11010_{(2)}$.

Варіант	Завдання до роботи
9	1. а) $342_{(10)}$; б) $30{,}375_{(10)}$; в) $97{,}14_{(10)}$. 2. а) $10001101_{(2)}$; б) $1110010{,}1011001_{(2)}$; г) $666{,}16_{(8)}$; д) $1C7{,}68_{(16)}$. 3. а) $1101010000_{(2)}+1011101001_{(2)}$; б) $1100101{,}01001_{(2)}+1111111{,}011_{(2)}$ 4. а) $1111110_{(2)}-1111011_{(2)}$; б) $1111100000_{(2)}-111110011_{(2)}$. 5. $1001010_{(2)} \times 1101111_{(2)}$.
10	1. а) $222_{(10)}$; б) $57{,}5_{(10)}$; в) $53{,}35_{(10)}$. 2. а) $101111111_{(2)}$; б) $10011000{,}1101011_{(2)}$; г) $140{,}22_{(8)}$; д) $1DE{,}54_{(16)}$. 3. а) $1101010000_{(2)}+11100100_{(2)}$; б) $1111100100{,}11_{(2)}+1111101000{,}01_{(2)}$ 4. а) $1010010100_{(2)}-11101110_{(2)}$; б) $10000001110_{(2)}-10011100_{(2)}$. 5. $111000_{(2)} \times 100111_{(2)}$.
11	1. а) $500_{(10)}$; б) $1017{,}25_{(10)}$; в) $123{,}72_{(10)}$ 2. а) $11010101_{(2)}$; б) $11110001{,}011011_{(2)}$;в) $1347{,}17_{(8)}$; г) $155{,}6C_{(16)}$. 3. а) $1000101101_{(2)}+1100000010_{(2)}$; б) $1111011{,}01_{(2)}+11100{,}11_{(2)}$. 4. а) $1101111100_{(2)}-100100010_{(2)}$; б) $1011010110_{(2)}-1011001110_{(2)}$. 5. $11011_{(2)} \times 10011_{(2)}$
12	1. а) $218_{(10)}$; б) $176{,}25_{(10)}$;в) $253{,}04_{(10)}$ 2. а) $1110100_{(2)}$; б) $100011{,}01_{(2)}$; в) $1665{,}3_{(8)}$; г) $FA{,}7_{(16)}$. 3. а) $11100000_{(2)}+1100000000_{(2)}$; б) $1101011{,}01_{(2)}+111111{,}1101_{(2)}$. 4. а) $10110010_{(2)}-1010001_{(2)}$; б) $1101000000_{(2)}-10000000_{(2)}$ 5. $10000_{(2)} \times 1110_{(2)}$
13	1. а) $305_{(10)}$ Б) $153{,}25_{(10)}$; В) $248{,}46_{(10)}$ 2. а) $1100111_{(2)}$; б) $100110{,}101_{(2)}$; В) $671{,}24_{(8)}$; г) $41A{,}6_{(16)}$. 3. а) $10000011_{(2)}+1000011_{(2)}$; б) $110010{,}101_{(2)}+1011010011{,}01_{(2)}$; 4. а) $100111001_{(2)}-110110_{(2)}$; б) $1111001110_{(2)}-111011010_{(2)}$. 5. $1100110_{(2)} \times 1011010_{(2)}$.

Варіант	Завдання до роботи
14	1. а) $164_{(10)}$; б) $712,25_{(10)}$; в) $11,89_{(10)}$ 2. а) $1001111_{(2)}$; б) $1100111,01_{(2)}$;в) $413,41_{(8)}$; г) $118,8C_{(16)}$. 3. а) $1100001100_{(2)}+1100011001_{(2)}$; б) $111111111,001_{(2)}+1111111110,0101_{(2)}$. 4. а) $1001101100_{(2)}-1000010111_{(2)}$; б)$1101100110_{(2)}-111000010_{(2)}$. 5. $10001_{(2)} \times 1010_{(2)}$.
15	1. а) $273_{(10)}$; б) $97,5_{(10)}$; в) $53,74_{(10)}$ 2. а) $110000_{(2)}$; б) $1001101,00011_{(2)}$; в) $1017,2_{(8)}$; г) $111,B_{(16)}$. 3. а) $1110001000_{(2)}+110100100_{(2)}$; б) $1001001,101_{(2)}+1111000,11_{(2)}$. 4. а) $1010111001_{(2)}-1010001011_{(2)}$; б) $1110101011_{(2)}-100111000_{(2)}$. 5. $10110_{(2)} \times 10010_{(2)}$.
16	1. а) $105_{(10)}$; б) $377,5_{(10)}$; в) $87,27_{(10)}$ 2. а) $11001001_{(2)}$; б) $11110110,01_{(2)}$; в) $112,04_{(8)}$; г) $334,A_{(16)}$. 3. а) $101000011_{(2)}+110101010_{(2)}$; б) $111010,010_{(2)}+1011011,11_{(2)}$. 4. а) $1111111000_{(2)}-100010011_{(2)}$; б) $1111101110_{(2)}-11100110_{(2)}$. 5. $10111_{(2)} \times 10100_{(2)}$.

2. ПРАКТИЧНА РОБОТА №2
«АРХІТЕКТУРА МІКРОКОНТРОЛЕРА i8051»

2.1 Загальні характеристики мікроконтролера i8051

Класичний мікроконтролер i8051 (MCS51) і вітчизняний аналог КМ1816ВЕ51 виконані на основі високорівневої n-МОН технології і випускалися в корпусі БІС, що має 40 зовнішніх виводів. Цокольовка корпусу MCS51 і найменування виводів показані на рис. 2.1. Для роботи MCS51 потрібно одне джерело електроживлення +5. Через чотири програмованих порти введення / виведення MCS51 взаємодіє із середовищем в стандарті TTL-схем з трьома станами виходу.

Корпус MCS51 має два виводи для підключення кварцевого резонатора, чотири виводи для сигналів, керуючих режимом роботи МК, і вісім ліній порту 3, які можуть бути запрограмовані користувачем на виконання спеціалізованих (альтернативних) функцій обміну інформацією з середовищем. Призначення виводів мікроконтролера 8051.

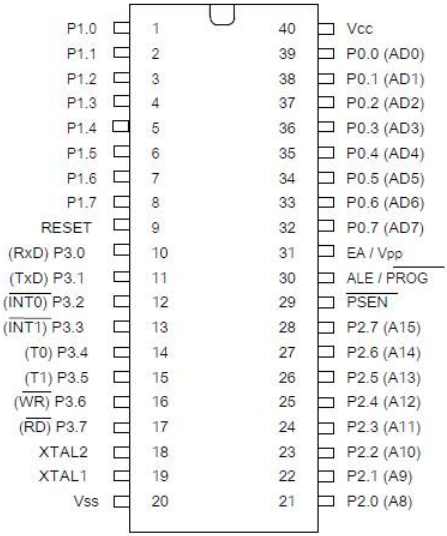


Рисунок 2.1 – Призначення виводів 8051

Позначення на Рис. 2.1:

- Vss – потенціал загального проводу ("землі");
- Vcc – основна напруга літанія +5 В;
- XTAL1, XTAL2 − виводи для підключення кварцевого резонатора;
- RESET (RST) – вхід загального скидання мікроконтролера;
- PSEN – дозвіл зовнішньої пам'яті програм; видається тільки при зверненні до зовнішнього ПЗУ;
- ALE – строб адреси зовнішньої пам'яті;
- EA – відключення внутрішньої програмної пам'яті; рівень 0 на цьому вході змушує мікроконтролер виконувати програму тільки з зовнішнього ПЗУ; ігноруючи внутрішнє (якщо останнє є);
- P0 – восьми бітний двонаправлений порт введення-виведення інформації: при роботі з зовнішніми ОЗУ і ПЗУ по лініях порту в режимі тимчасового мультиплексування видається адреса зовнішньої пам'яті, після чого здійснюється передача або прийом даних;
- P1 – восьмибітний квазі двонаправлений порт введення / виводу: кожен розряд порту може бути запрограмований як на введення, так і на виведення інформації, незалежно від стану інших розрядів;
- P2 – восьмибітний квазі двонаправлений порт, аналогічний P1; крім того, виводи цього порту використовуються для видачі адресної інформації при зверненні до зовнішньої пам'яті програм або даних (якщо використовується 16-бітова адресація останньої).
- P3 – восьмибітний квазі двонаправлений порт, аналогічний. P1; крім того, виводи цього порту можуть виконувати ряд альтернативних функцій, які використовуються при роботі таймерів, порту послідовного введення-виведення, контролера переривань, і зовнішньої пам'яті програм і даних.

2.2 Структура мікроконтролера

Основу структурної схеми MCS51 (рис. 2.2) утворює внутрішня двонаправлена 8-бітна шина, яка пов'язує між собою всі основні вузли і пристрої: резидентну пам'ять програм (RPM), резидентну пам'ять даних (RDM), арифметико-логічний пристрій (ALU), блок регістрів спеціальних функцій, пристрій управління (CU) і порти введення / виводу (P0-P3).

Розглянемо основні елементи структури і особливості організації обчислювального процесу в MCS51.

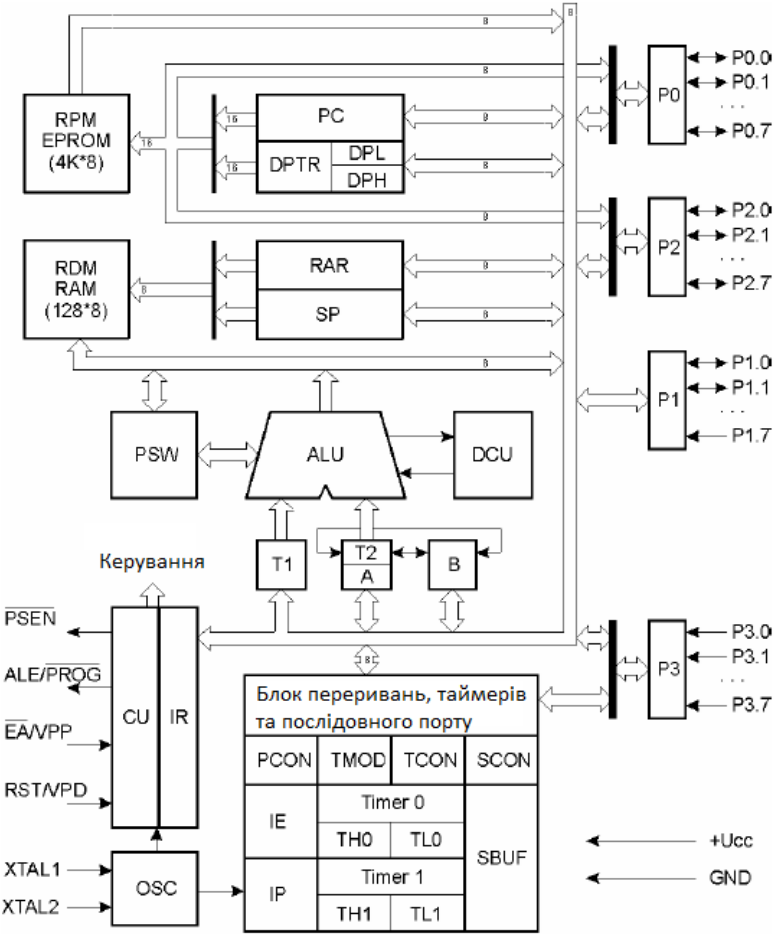


Рисунок 2.2 – Структурна схема MCS51

2.2.1 Арифметико-логічний пристрій

8-бітове арифметико-логічний пристрій (ALU) може виконувати арифметичні операції складання, віднімання, множення і ділення; логічні операції І, АБО, що виключає АБО, а також операції циклічного зсуву, скидання, інвертування і т.п. До входів підключені

програмно-недоступні регістри T1 і T2, призначені для тимчасового зберігання операндів, схема десяткового корекції (DCU) і схема формування ознак результату операції (PSW).

Найпростіша операція додавання використовується в ALU для інкрементування вмісту регістрів, просування регістра- покажчика даних (RAR) і автоматичного обчислення наступної адреси резидентної пам'яті програм. Найпростіша операція віднімання використовується в ALU для декрементування регістрів і порівняння змінних.

Найпростіші операції автоматично утворюють "тандеми" для виконання таких операцій, як, наприклад, інкрементування 16- бітних реєстрових пар. У ALU реалізується механізм каскадного виконання найпростіших операцій для реалізації складних команд. Так, наприклад, при виконанні однієї з команд умовної передачі управління по результату порівняння в ALU тричі інкрементується єлічильник команд (PC), двічі проводиться читання з RDM, виконується арифметичне порівняння двох змінних, формується 16-бітова адреса переходу і приймається рішення про те, робити чи не робити перехід за програмою. Всі перераховані операції виконуються лише за 2 мкс.

Важливою особливістю ALU є його здатність оперувати не тільки байтами, але і бітами. Окремі програмно-доступні біти можуть бути встановлені, скинуті, інвертовані, передані, перевірені та використані в логічних операціях. Ця здатність досить важлива, оскільки для управління об'єктами часто застосовуються алгоритми, що містять операції над вхідними та вихідними булевими змінними, реалізація яких засобами звичайних мікропроцесорів пов'язана з певними труднощами.

Таким чином, ALU може оперувати чотирма типами інформаційних об'єктів: булевими (1 біт), цифровими (4 біта), байтними (8 біт) і адресними (16 біт). У ALU виконується 51 різна операція пересилання або перетворення цих даних. Так як використовується 11 режимів адресації (7 для даних і 4 для адрес), то шляхом комбінування операції і режиму адресації базове число команд 111 розширюється до 255 з 256 можливих при однобайтному коді операції.

2.2.2 Резидентна пам'ять програм/даних і регістри загального призначення

Резидентні (розміщені на кристалі) пам'ять програм (RPM) і пам'ять даних (RDM) фізично і логічно розділені, мають різні механізми адресації, працюють під управлінням різних сигналів і виконують різні функції.

2.2.2.1 Пам'ять програм.

Пам'ять програм RPM має ємність 4 Кбайта і призначена для зберігання команд, констант, керуючих слів ініціалізації, таблиць перекодування вхідних і вихідних змінних і т.п. Пам'ять має 16-бітну шину адреси.

При зверненні до зовнішньої пам'яті програм (EPM) всі мікроконтролери сімейства 8051 завжди використовують 16-розрядну адресу, що забезпечує їм доступ до 64 Кбайт ПЗУ. Мікроконтролер звертається до програмної пам'яті при читанні коду операції та операндів (використовуючи лічильник команд PC), а також при виконанні команд перенесення байта з пам'яті програм в акумулятор. При виконанні команд перенесення даних адресація комірки пам'яті програм, з якої будуть прочитані дані, може здійснюватися з використанням як лічильника PC, так і спеціального двухбайтового регістра-покажчика даних DPTR.

Пам'ять даних і регістри загального призначення. Пам'ять даних RDM призначена для зберігання змінних в процесі виконання прикладної програми, адресується одним байтом і має ємність 128 байт. Крім того, до її адресного простору примикають адреси регістрів спеціальних функцій, які перераховані в таблиці 1.2.

Пам'ять даних, так само як і пам'ять програм, може бути розширена до 64 Кбайт шляхом підключення зовнішніх мікросхем. Перші 32 байта організовані в чотири банки регістрів загального призначення (РОН), позначаються відповідно банк 0 - банк 3 (див. таблицю 1.1). Кожен з них складається з восьми регістрів R0 - R7. У будь-який момент програмі доступний тільки один банк регістрів, номер якого міститься в третьому і четвертому бітах слова стану програми PSW (див. Нижче).

Адресний простір може конфігуруватися розробником на свій розсуд: в ньому розташовуються стек, системні і призначені для користувача області даних. Звернення до осередків пам'яті даних можливо двома способами. Перший спосіб − пряма адресація комірки пам'яті. У цьому випадку адреса осередку є операндом відповідної команди. Другий спосіб − непряма адресація з допомогою регістрів R0 або R1: перед виконанням 17 відповідної команди в один з них повиненна бути занесена адреса комірки, до якої необхідно звернутися. Для звернення до зовнішньої пам'яті даних (EDM) використовується тільки непряма адресація з допомогою регістрів R0 і R1 або за допомогою 16-розрядного регістра-покажчика DPTR. Він належить до групи регістрів спеціальних функцій, і з його допомогою можна адресувати всі 64 Кбайта зовнішньої пам'яті. Частина пам'яті даних являє собою так звану бітову область, в ній є можливість за допомогою спеціальних бітових команд адресуватися до кожного розряду осередків пам'яті. Адреса прямо адресованих бітів може бути записана або у вигляді (адреса Байта). (Розряд), наприклад вираз 21.3 означає третій розряд комірки пам'яті з адресою 21H, або у вигляді абсолютної бітової адреси. Відповідність цих двох способів адресації можна визначити по таблиці.

Таблиця 2.1 – Адреси бітових областей пам'яті мікроконтролера 8051 і регістрів загального призначення.

Адреса біту	Адреси бітів за розрядами							
Adr	D7	D6	D5	D4	D3	D2	D1	D0
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	9	8

20H	7	6	5	4	3	2	1	0
1FH … 18H	Банк 3 РЗП							
17H … 10H	Банк 2 РЗП							
0FH … 08H	Банк 1 РЗП							
07H … 00H	Банк 0 РЗП							

Примітка. Адреса прямо адресованих бітів може бути записаний або у вигляді виразу (адреса Байта). (Розряд), наприклад вираз 21.3 означає адресу третього розряду комірки пам'яті з адресою 21H, або у вигляді абсолютного бітового адреси, який для даного біта дорівнює (див. Таблицю 2.1) 0B.

2.2.2.2 Регістри спеціальних функцій

До адресного простору пам'яті даних примикає адресний простір регістрів спеціальних функцій SFR (Special Function Register).

Таблиця 2.2 – Розміщення регістрів спеціальних функцій в просторі SFR Адреса Символ

Адрес	Символ	Найменування
0E0H	*ACC	Акумулятор (Accumulator)
0F0H	*B	Регістр розширювач акумулятора (Multiplication Register)
0D0H	*PSW	Слово стану програми (Program Status Word)
080H	*P0	Порт 0 (SFR P0)
090H	*P1	Порт 1 (SFR P1)
0A0H	*P2	Порт 2 (SFR P2)
0B0H	*P3	Порт 3 (SFR P3)
081H	SP	Регістр покажчик стека (Stack Pointer)
083H	DPH	Старший байт регістру покажчика даних DPTR (Data Pointer High)
082H	DPL	Молодший байт регістру покажчика даних DPTR (Data Pointer Low)
08CH	TH0	Старший байт таймера 0 ()
08AH	TL0	Молодший байт таймера 0 ()
08DH	TH1	Старший байт таймера 1 ()
08BH	TL1	Молодший байт таймера 1 ()
089H	TMOD	Регістр режимів таймерів лічильників (Timer/Counter Mode Control Register)
088H	*TCON	Регістр управління статусом таймерів (Timer/Counter Control Register)
0B8H	*IP	Регістр пріоритетів (Interrupt Priority Control Register)
0A8H	*IE	Регістр маски переривання (Interrupt Enable Register)
087H	PCON	Регістр управління потужністю (Power Control Register)

098H	*SCON	Регістр управління приймачем (Serial Port Control Register)
099H	SBUF	Буфер приймача (Serial Data Buffer)

Примітка. Регістри, символ яких відзначений знаком (*), допускають адресацію своїх окремих біт при використанні команд з групи команд операцій над бітами. Адреси, за якими розташовані ці регістри, наведені в таблиці 2. 3.

Таблиця 2.3 – Карта адресованих бітів в блоці регістрів спеціальних функцій

Адреса байту	Адреси бітів за розрядами								Ім’я регістру
Adr	D7	D6	D5	D4	D3	D2	D1	D0	Name
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
…	…								…
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
…	…								…
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
…	…								…
B8H	-	-	-	BC	BB	BA	B9	B8	IP
…	…								…
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
…	…								…
A8H	AF	-	-	AC	AB	AA	A9	A8	IE
…	…								…
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
…	…								…
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
…	…								…
90H	97	96	95	94	93	92	91	90	P1
…	…								…
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
…	…								…
80H	87	86	85	84	83	82	81	80	P0

Примітка. Адреса прямо адресованих бітів може бути записана або у вигляді виразу (Назва Регістру). (Розряд), наприклад вираз SCON.3 означає адресу третього розряду регістра SCON, або у вигляді абсолютної бітової адреси, яка для даного біта дорівнює (див. Таблицю 3) 9B. Крім того, деякі біти регістрів мають власні назви, так наприклад даний біт має назву TB8. Відзначимо, що регістри займають лише частину 128-байтового адресного простору. Тобто осередки пам'яті з адресами 80H-0FFH, які не зайняті регістрами,

фізично відсутні, на кристалах 20 мікроконтролерів сімейства 8051 при зверненні до них можна прочитати лише код команди повернення.

- Регістри спеціальних функцій керують роботою блоків, що входять в мікроконтролер.
- Регістри-засувки SFR паралельних портів P0 ... P3 - служать для введення-виведення інформації.
- Дві реєстрові пари з іменами TH0, TL0 і TH1, TL1 являють собою регістри, двох програмно-керованих 16-бітових таймерів-лічильників.
- Режими таймерів-лічильників задаються з використанням регістра TMOD, а управління ними здійснюється за допомогою регістра TCON.
- Для управління режимами енергоспоживання мікро-ЕОМ використовується
- регістр PCON.
- регістри IP і IE керують роботою системи переривань мікро-ЕОМ,
- регістри SBUF і SCON - роботою приймача послідовного порту.
- Регістр-покажчик стека SP в мікро-ЕОМ розглянутого сімейства - восьми бітний. Він може адресувати будь-яку область внутрішньої пам'яті даних. На відміну від мікропроцесора КР580ВМ80, у мікро-ЕОМ сімейства 8051 стек «росте вгору», тобто перед виконанням команди PUSH або CALL вміст SP інкременується, після чого проводиться запис інформації в стек. Відповідно при добуванні інформації з стека регістр SP декременується після вилучення інформації. В процесі ініціалізації мікро-ЕОМ після сигналу скидання або при включенні напруги живлення в SP заноситься код 07H. Це означає, що перший елемент стека буде розташовуватися в комірці пам'яті з адресою 08H.
- Регістр-покажчик даних DPTR найчастіше використовують для фіксації 16-бітної адреси в операціях звернення до зовнішньої пам'яті програм і даних. З точки зору програміста він може виступати як у вигляді одного 16-бітного регістра, так і у вигляді двох незалежних регістрів DPL і DPH.
- акумулятор (ACC) є джерелом операнда і місцем фіксації результату при виконанні арифметичних, логічних операцій і ряду операцій передачі даних. Крім того, тільки з використанням акумулятора можуть бути виконані операції зрушень, перевірка

на нуль, формування прапора паритету і т.п. У розпорядженні користувача є 8 регістрів загального призначення R0-R7 одного з чотирьох можливих банків. При виконанні багатьох команд в АЛУ формується ряд ознак операції (прапорів), які фіксуються в регістрі PSW.

- ❖ Регістр В використовується як джерело і як приймач при операціях множення і ділення, звернення до нього, як до регістру SFR, виробляється аналогічно акумулятору.
- ❖ При виконанні ряду команд в арифметико-логічному пристрої (АЛП) формуються ознаки операцій - прапори, які фіксуються в регістрі PSW. У таблиці 2.4 наводиться перелік прапорів PSW, даються їх символічні імена і описуються умови їх формування.

Таблиця 2.4 – Формат слова стану програми PSW.

<table>
<tr><th>Символ</th><th>Розряд</th><th>Ім’я та призначення</th></tr>
<tr><td>C</td><td>PSW.7</td><td>Флаг пріоритету. Встановлюється і скидається апаратно в кожному циклі команди і фіксує непарне / парне число одиничних біт в акумуляторі</td></tr>
<tr><td>AC</td><td>PSW.6</td><td>Не використовується</td></tr>
<tr><td>F0</td><td>PSW.5</td><td>Флаг переповнення. Встановлюється і скидається апаратно при виконанні арифметичних операцій</td></tr>
<tr><td>RS1
RS0</td><td>PSW.4
PSW.3</td><td>Біти вибору використовуваного банку регістрів. Можуть бути змінені програмним шляхом
<table>
<tr><th>RS0</th><th>RS1</th><th>Банк</th><th>Межі адрес ОЗУ</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00H - 07H</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>08H - 0FH</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>10H - 17H</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18H - 1FH</td></tr>
</table></td></tr>
<tr><td>OV</td><td>PSW.2</td><td>Флаг користувача. Може бути встановлений, скинутий або перевірений програмою користувача</td></tr>
<tr><td>-</td><td>PSW.1</td><td>Флаг допоміжного переносу. Встановлюється і скидається тільки апаратними засобами при виконанні команд додавання і віднімання і сигналізує про перенесення або позику в бите 3 акумулятора</td></tr>
<tr><td>P</td><td>PSW.0</td><td>Флаг перенесення. Встановлюється і скидається як апаратурно, так і програмним шляхом</td></tr>
</table>

Найбільш "активним" прапором PSW є прапор переносу, який бере участь і модифікується в процесі виконання безлічі операцій, включаючи додавання, віднімання і зрушення. Крім того, прапор переносу (C) виконує функції "булева акумулятора" в командах, що маніпулюють з бітами. Прапор переповнення (OV) фіксує

арифметичне переповнення при операціях над цілими числами зі знаком і робить можливим використання арифметики в додаткових кодах. ALU не керує прапорами селекції банку регістрів (RS0, RS1), їх значення повністю визначається прикладною програмою і використовується для вибору одного з чотирьох реєстрових банків.

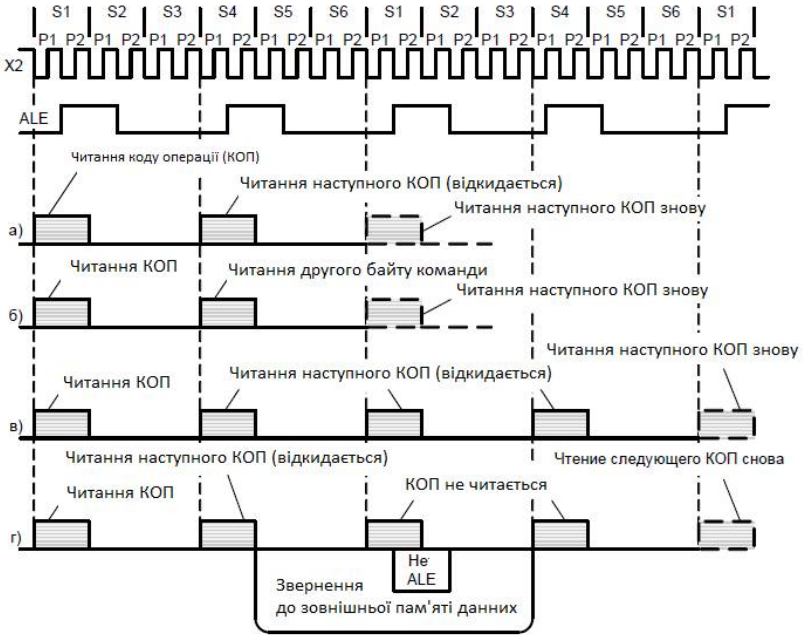
У мікропроцесорах, архітектура яких спирається на акумулятор, більшість команд працюють з ним, використовуючи неявну адресацію. В Intel 8051 інша справа. Хоча процесор має в своїй основі акумулятор, він може виконувати безліч команд і без його участі. Наприклад, дані можуть бути передані з будь-якого елементу RDM в будь-який регістр, будь-який регістр може бути завантажений безпосереднім операндом і т.д. Багато логічні операції можуть бути виконані без участі акумулятора. Крім того, змінні можуть бути інкрементовані, декрементовані і перевірені без використання акумулятора. Прапори та керуючі біти можуть бути перевірені і змінені аналогічно.

2.3 Пристрій управління і синхронізації

Кварцовий резонатор, що підключається до зовнішніх висновків мікроконтролера, керує роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації.

Пристрій управління (CU) на основі сигналів синхронізації формує машинний цикл фіксованої тривалості, рівної 12 періодам резонатора або шести станів первинного керуючого автомата (S1 - S6). Кожен стан керуючого автомата містить дві фази (P1, P2) сигналів резонатора. У фазі P1, як правило, виконується операція в АЛП, а в фазі P2 здійснюється міжрегістрова передача. Весь машинний цикл складається з 12 фаз, починаючи з фази S1P1 і закінчуючи фазою S6P2, як показано на рис. 3. Ця тимчасова діаграма ілюструє роботу пристрою управління MCS51 при вибірці і виконанні команд різного ступеня складності. Всі заштриховані сигнали є внутрішніми і недоступні користувачеві MCS51 для контролю. Зовнішніми, які спостерігаються сигналами є тільки сигнали резонатора і стрибають адреси зовнішньої пам'яті (ALE). Як видно з тимчасової діаграми, сигнал ALE формується двічі за один машинний цикл (S1P2 - S2P1 і S4P2 - S5P1) і використовується для управління процесом звернення до зовнішньої пам'яті.

Більшість команд мікроконтролера виконується за один машинний цикл. Деякі команди, які оперують з 2-байтними словами або пов'язані зі зверненням до зовнішньої пам'яті, виконуються за два машинних цикли. Тільки команди ділення і множення вимагають чотирьох машинних циклів. На основі цих особливостей роботи пристрою управління, проводиться розрахунок часу виконання прикладних програм.



а - команда 1 байт / 1 цикл, наприклад INC A; б - команда 2 байта / 1 цикл, наприклад ADD A, # d; в - команда 1 байт / 2 циклу, наприклад INC DPTR; г - команда 1 байт / 2 циклу, наприклад MOVX

Рисунок 2.3 – Послідовності вибірки і виконання команд в MCS51.

2.4 Організація портів введення/виводу мікроконтролера 8051

Всі чотири порти (P0-P3) призначені для введення або виведення інформації побайтно.

Кожен з портів містить регістр-засувку (SFR P0 - SFR P3), вхідний буфер і вихідний драйвер. Кожен з розрядів регістра- засувки SFR є D-тригером, інформація в якиго заноситься з внутрішньої шини даних мікроконтролера за сигналом «Запис в SFR Px» (x = 0, 1, 2, 3) від центрального процесорного елемента (CPU). З прямого виходу D-тригера інформація маже бути виведена на внутрішню шину по сигналу «Читання SFR Px» від CPU, а з виведення мікросхеми («із зовнішнього світу») за сигналом «Читання виводів Px». Одні команди активізують сигнал «Читання SFR PI», інші - «Читання виводів PI». Вихідні драйвери портів 0 і 2, а також вхідний буфер порту 0 використовуються при зверненні до зовнішньої пам'яті (ВП). При цьому через порт 0 в режимі тимчасового мультиплексування спочатку виводиться молодший байт адреси ВП, а потім видається або приймається байт даних. Через порт 2 виводиться старший байт адреси в тих випадках, коли розрядність адреси дорівнює 16 біт.

Всі виводи порту 3 можуть бути використані для реалізації альтернативних функцій, перерахованих в таблиці 6. Ці функції можуть бути задіяні шляхом запису 1 у відповідні біти регістра-засувки (P3.0-P3.7) порту 3.

Порт 0 є двонаправленим, а порти 1-3 - квазідвонаправленими. Кожна лінія портів може бути використана незалежно для введення або виведення.

За сигналом RST в регістри-засувки всіх портів автоматично записуються одиниці, які налаштовують їх тим самим на режим введення.

Всі порти можуть бути використані для організації введення / виведення інформації по двонаправленим лініям передачі. Однак порти 0 і 2 не можуть бути використані для цієї мети в разі, якщо система має зовнішню пам'ять, зв'язок з якою організовується через загальну поділювану шину адреси / даних, що працює в режимі тимчасового мультиплексування.

2.4.1 Особливості електричних характеристик портів

Вихідні каскади тригерів SFR портів P1 − P3 виконані на польових транзисторах з внутрішнім навантаженням, в той час як аналогічні каскади тригерів SFR P0 – на транзисторах з відкритим

стоком. Кожна лінія будь-якого з портів може незалежно використовуватися як для введення, так і для виведення інформації (для ліній портів P0 і P2 це справедливо тоді, коли вони не використовуються для звернення до зовнішньої пам'яті).

Для перекладу будь-якої лінії портів P1 - P3 в режим введення інформації необхідно в відповідний розряд SFR занести 1. При цьому вихідний польовий транзистор відключається. Внутрішній навантажувальний резистор як би «підтягує» потенціал виведення до напруги живлення, в той час як зовнішнє навантаження може зробити його нульовим. Вихідні каскади порту P0 мають іншу структуру. Навантажувальний польовий транзистор лінії порту включений тільки тоді, коли порт виводу звертається до зовнішньої пам'яті. В інших випадках навантажувальний транзистор відключений. Таким чином, при роботі в режимі звичайного введення-виведення інформації (як, наприклад, порт P1) вихідні каскади порту P0 є ступені на транзисторах з відкритим стоком. Запис 1 в відповідний біт SFR відключає і другий транзистор, що призводить до того, що вивід БІС виявляється під «плаваючим» потенціалом.

Оскільки вихідні каскади портів P1 – P3 мають внутрішнє навантаження, при переведенні в режим введення інформації вони стають джерелами струму для мікросхеми або транзистора, навантажених на даний вивід.

Таблиця 2.5 – Альтернативні функції виводів порту P3

Символ	Розряд	Ім’я та призначення
RD	P3.7	Читання. Активний сигнал низького рівня формується апаратно при зверненні до зовнішньої пам'яті даних
WR	P3.6	Запис. Активний сигнал низького рівня формується апаратно при зверненні до зовнішньої пам'яті даних
T1	P3.5	Вхід таймера / лічильника 1 або тест-вхід
T0	P3.4	Вхід таймера / лічильника 0 або тест-вхід
INT1	P3.3	Вхід запиту переривання 1. Сприймається сигнал низького рівня або зріз
INT0	P3.2	Вхід запиту переривання 0. Сприймається сигнал низького рівня або зріз
TXR	P3.1	Вихід передавача послідовного порту в режимі UART. Вихід синхронізації в режимі регістра зсуву.
RXD	P3.0	Вхід приймача послідовного порту в режимі UART. Ввід / вивід даних в режимі регістра зсуву.

2.4.2 Особливості роботи портів

Звернення до портів введення/виводу можливо з використанням команд, що оперують з байтом, окремим бітом, довільною комбінацією бітів. При цьому в тих випадках, коли порт є одночасно операндом і місцем призначення результату, пристрій управління автоматично реалізує спеціальний режим, який DC T P3.X Лінія внутр. шини Запис в SFR + Ucc читання SFR читання виведення & Альтернативний сигнал виходу Альтернативний вхідний сигнал.

Навантаження називається "читання-модифікація-запис". Цей режим звертання припускає введення сигналів не з зовнішніх виводів порту, а з його регістра-засувки, що дозволяє виключити неправильне зчитування раніше виведеної інформації [1]. Цей механізм звернення до портів реалізований в командах:

- ANL - логічне І, наприклад, ANL P1, A;
- ORL - логічне АБО, наприклад, ORL P2, A;
- XRL - виключає АБО, наприклад, XRL P3, A;
- JBC - перехід, якщо в адресованому біті встановлена одиниця, і подальше скидання біта, наприклад, JBC P1.1, LABEL;
- CPL - інверсія біта, наприклад, CPL P3.3;
- INC - інкремент порту, наприклад, INC P2;
- DEC - декремент порту, наприклад, DEC P2;
- DJNZ - декремент порту і перехід, якщо його вміст не дорівнює нулю, на-приклад, DJNZ r, LABEL;
- MOV PX.Y, C - передача біта перенесення в біт X порту Y;
- SET PX.Y - установка біта X порту Y;
- CLR PX.Y - скидання біта X порту Y.

Зовсім не очевидно, що останні три команди в наведеному списку є командами "читання-модифікація-запис". Однак це саме так. За цими командами спочатку зчитується байт з порту, а потім записується новий байт в регістр-засувку [1].

Причиною, через яку команди "читання-модифікація-запис" забезпечують роздільний доступ до регістру-клямки порту і до зовнішніх виводів порту, є необхідність виключити можливість неправильного прочитання рівнів сигналів на зовнішніх виводах.

Припустимо для прикладу, що лінія X порту Y з'єднується з базою потужного транзистора і вихідний сигнал на ній призначений для його управління. Коли в даний біт записана 1, то транзистор включається. Якщо для перевірки стану виконавчого механізму (в

нашому випадку – потужного транзистора) прикладної програми потрібно прочитати стан вихідного сигналу в тому ж біті порту, то зчитування сигналу (наприклад, командою MOV ACC, PY) з зовнішнього виведення порту, а не з D-тригера регістра-засувки порту призведе до неправильного результату: одиничний сигнал на базі транзистора має відносно низький рівень і буде інтерпретований в МК як сигнал 0. Команди "читання-модіфікація- запис" реалізують зчитування з регістру-засувки, а не з зовні на виведення порту.

2.5 Завдання для практичної роботи

Відповісти на наступні питання:

1. Як відбувається виконання програми в МП, пояснити на структурній схемі МП (рис. 2.2).

2. Які є регістри в МП 8051, яке їх призначення?

3. Перерахуйте характерні риси архітектури мікроконтролерів, спрямовані на взаємодію з об'єктами управління.

4. Вкажіть призначення регістрів спеціальних функцій.

5. Перерахуйте альтернативні функції паралельних портів.

6. В якому стані знаходяться паралельні порти після формування сигналу RST?

7. Чи може порт одночасно бути джерелом операнда і приймачем результату операції?

8. Які регістри можна використовувати для наступних цілей:

a. Додавання і віднімання

b. Підрахунку числа циклів

c. Множення і ділення

d. Індикації нульового результату

f. Адресації виконуваної команди

8. Вкажіть, які з регістрів спеціальних функцій допускають бітову адресацію.

9. Перерахуйте команди операцій з бітами.

3. ПРАКТИЧНА РОБОТА №3
«ОТРИМАННЯ ІНФОРМАЦІЇ З ДАТЧИКІВ»

3.1 Опитування двійкового датчика. Очікування події

У пристроях і системах управління об'єктами події фіксуються з використанням різноманітних датчиків цифрового і аналогового типів. Найбільшого поширення мають виконавчі датчики типу так / ні.

Очікування статичного сигналу. Типова процедура очікування події (WAIT) складається з наступних дій: введення сигналу від датчика, аналізу значення сигналу і передачі управління в залежності від стану датчика. Конкретна програмна реалізація процедури залежить від того, яким чином датчик підключений до мікроконтролеру. Наприклад, при підключенні датчика до лінії біта 3 порту 1 програма очікування замикання контакту матиме вигляд:

```
WAIT0:  JNB P1.3,   WAIT0   ;очікування розімкнення датчика
```

Іншим окремим випадком є процедура очікування розмикання контакту, яка може бути реалізована наступним чином:

```
WAIT0:  JB P1.3,   WAIT0   ;очікування замкнення датчика
```

Для опитування особливо важливих датчиків з метою зменшення часу реакції на виняткову ситуацію в об'єкті доцільно використовувати режим переривання.

Очікування імпульсного сигналу. Особливість процедури очікування імпульсного сигналу полягає в тому, що мікроконтролер повинен виявити не тільки факт появи, а й факт закінчення сигналу.

Для програмування цієї процедури зручно скористатися розглянутими вище прикладами, змонтувавши їх послідовно в лінійну програму. Оформляти процедури WAITC і WAIT0 у вигляді підпрограм недоцільно, так як це подовжує програму, а довжина і, отже, час виконання програми визначають мінімальну тривалість імпульсу, який може бути виявлений програмою.

Послідовність склеювання процедур WAITC і WAIT0 залежить від форми імпульсу. Для "негативного" імпульсу (1 → 0 → 1)

процедура WAITC передує процедурі WAIT0, для "позитивного" (0 → 1 → 0) слідує за нею.

Нижче наведено приклад програмної реалізації процедури очікування "негативного" імпульсного сигналу при підключенні датчика до біту 3 порту 1 за умови, що початковий стан входу - одиничне:

```
WAITIMP:
WAITC:      JB P1.3, WAITC  ;очікування P1.3 = 0
WAIT0:      JNB P1.3, WAIT0 ;очікування P1.3 = 1
```

Програмна реалізація циклу очікування накладає обмеження на тривалість імпульсу: імпульси тривалістю менше часу виконання циклу очікування можуть бути «не помічені» мікроконтролером. Для виявлення короткочасних імпульсів зазвичай використовують спосіб фіксації імпульсу на зовнішньому тригері прапора. На вхід в цьому випадку надходить не короткочасний сигнал з датчика, а прапор, що формується тригером. Тригер встановлюється по фронту імпульсу, а скидається програмним шляхом - видачею спеціального керуючого впливу. Тривалість імпульсу при цьому буде обмежена знизу тільки швидкодією тригера.

3.2 Усунення брязкоту контактів

При роботі з датчиками, що мають механічні або електромеханічні контакти (кнопки, клавіші, реле і клавіатури), виникає явище, зване брязкотом. Він полягає в тому, що при замиканні контактів можлива поява відскоку (BOUNCE) контактів, яке призводить до перехідного процесу. При цьому сигнал з контакту може бути прочитаний мікро контролером як випадкова послідовність нулів і одиниць. Придушити це небажане явище можна схемотехнічними засобами, але частіше це робиться програмним шляхом.

Найбільшого поширення набули два програмних способу очікування сталого значення:

1. Підрахунок заданого числа співпадаючих значень сигналу;
2. Часова затримка.

Суть першого методу полягає в багаторазовому зчитуванні сигналу з контакту. Підрахунок вдалих опитувань, які виявили, що контакт стійко замкнутий, ведеться програмним лічильником. Якщо після серії вдалих опитувань зустрічається невдалий, то підрахунок починається спочатку. Контакт вважається стійко замкнутим, якщо послідувало N вдалих опитувань. Число N підбирається експериментально для кожного типу використовуваних датчиків і лежить в межах від 5 до 50.

Приклад програмного придушення брязкоту контакту наводиться для випадку, коли датчик імпульсного сигналу підключений до входу T0, рахунок вдалих опитувань ведеться в регістрі R3, N = 20:

```
DBNC:   MOV R3, #20     ;ініціалізація лічильника
DBNC1:  JB P3.4, DBNC   ;якщо контакт розімкнутий,
                        ;то почати підрахунок спочатку
        DJNZ R3, DBNC1  ;повторювати, поки R3
                        ;не стане рівним 0.
```

Введення затримки. Усунення брязкоту контакту шляхом введення тимчасової затримки полягає в наступному. Програма, виявивши замикання контакту, забороняє опитування його стану на час, свідомо більше тривалості перехідного процесу. Програма написана для випадку підключення датчика до входу T0 і програмної реалізації тимчасової затримки:

```
DBNCDL:     JB P3.4, DBNCDL     ;очікування нуля на вході
            CALL DELAY          ;виклик підпрограми затримки
EXIT:       ...                 ;визід із процедури
```

Тимчасова затримка в межах 1-10 мс підбирається експериментально для кожного типу датчиків і реалізується підпрограмою DELAY.

3.3 Підрахунок числа імпульсів

Часто в керуючих програмах виникає необхідність очікування ланцюжка подій, що подається послідовністю імпульсних сигналів від датчиків. Розглянемо дві типові процедури: підрахунок числа

імпульсів між двома подіями і підрахунок числа імпульсів в заданий інтервал часу.

Підрахунок числа імпульсів між двома подіями. Один з можливих варіантів процедури підрахунку може бути реалізований, якщо використовувати вхід T1 як вхід лічильника подій. В акумуляторі фіксується число імпульсів, представлене в двійковому коді (максимальна кількість 255).

```
CNTEVNT:    MOV TMOD, #0100000B ;налаштування лічильника 1
            MOV TH1, #0         ;скидання лічильника імпульсів
WAIT0:      JB P3.4, WAIT0      ;очикування ввімкнення лічильника
            SETB TCON.6         ;пуск лічильника 1
WAIT1:      JNB P3.4, WAIT1     ;очикування вимкнення лічильника
            CLR TCON.6          ;зупинка лічильника 1
            MOV A, TH1          ;фіксація 1/22 числа імпульсів
EXIT:       ...                 ;вихід із процедури.
```

Підрахунок числа імпульсів за заданий проміжок часу. При вирішенні завдання перетворення числа імпульсного коду в двійковий код, а також в ряді інших завдань може виникнути необхідність підрахунку числа імпульсів за заданий інтервал часу. Ця процедура може бути реалізована різними способами:

1. Програмною реалізацією тимчасового інтервалу і програмним підрахунком числа імпульсів на вході;
2. Програмною реалізацією тимчасового інтервалу і апаратним підрахунком числа імпульсів (на внутрішньому таймері / лічильнику);
3. Апаратної реалізацією тимчасового інтервалу і програмним підрахунком числа імпульсів;
4. Апаратна реалізація тимчасового інтервалу з апаратним підрахунком числа імпульсів.

Четвертий спосіб підрахунку імпульсів вимагає використання двох лічильників. На T/C1 можна виконувати підрахунок числа імпульсів, а на T/C0 − відлік заданого інтервалу. Датчик імпульсів повинен бути підключений до входу T1:

```
TIME = NOT(10000)+1             ;визначення константи TIME для
                                ;відрахунку інтервалу 10 мс
CNTTIME:    MOV TMOD, #0x51     ;налаштування T/C, 16 біт
                                ;1 - лічильник, 0 - таймер
            CLR A               ;скидання аккумулятора
            MOV TH1, A          ;скидання T/C1
```

```
            MOV TL1, A
            MOV TH0, #HIGH(TIME);завантаження у T/C0
            MOV TL0, #LOW(TIME) ;константи TIME
            ORL TCON,#0x50      ;пуск T/C1 та T/C0
WAIT:       JBC TCON.5, EXIT    ;перевірка переповнення
            SJMP WAIT           ;повтор поки TF=0
EXIT:       MOV B,  TH1         ;фіксація числа імпульсів
            MOV A,  TL1
            ...                 ;вихід із процедури
```

3.4 Опитування групи двійкових датчиків

Мікроконтролери найчастіше мають справу не з одним датчиком, як в розглянутих вище прикладах, а з групою автономних, логічно незалежних або взаємозв'язаних, які формують двійковий код датчиків. При цьому мікроконтролер може виконувати процедуру опитування датчиків і передачі управління окремими фрагментами прикладної програми в залежності від прийнятого коду.

Програмну реалізацію процедури очікування заданого коду (WTCODE) розглянемо для випадку підключення групи з восьми взаємопов'язаних статичних датчиків до входів порту 1:

```
WTCODE: MOV A, #10          ;завантаження в аккумулятор еталонного
                             ;коду
WAIT:  CJNE A, P1. WAIT  ;якщо кодова комбінация на входах порту 1
                           ;не співпала з еталонним значенням, то
                           ;чекати
EXIT:   ...              ;вихід з процедури.
```

Передача управління по коду. При опитуванні довічних датчиків передачу управління зручно здійснювати за таблицею переходів. Нижче наводиться текст програми, що здійснює передачу управління однієї з восьми прикладних програм PROG0 - PROG7. Передача проводиться в залежності від кодової комбінації на входах P1.0 - P1.2.

Адреса рядка таблиці, в якій зберігаються адреси переходів, обчислюється як сума відносного (всередині поточної сторінки резидентної пам'яті програм) початкового адреси таблиці BASE і коду, прийнятого від датчиків. Команди MOVC A, @A+DPTR, JMP @A+DPTR, таблиця BASE і програми PROG0 - PROG7 повинні розташовуватися в одній сторінці пам'яті програм.

```
GOCODE: MOV DPTR, #LOW BASE ;завантаження в DPTR початкової
                            ;адреси таблиці переходів
        IN A, P1            ;ввід байту
        ANL A, #0x07        ;виділення молоших бітів
        MOVC A,@A+DPTR      ;читання з пам'яті програм
                            ;(з таблиці переходів) адреси
переходу
        MOV DPTR, #0x0000   ;обнулення DPTR
        JMP @A+DPTR         ;передача керування
BASE:   DB LOW PROGO        ;таблиця переходів
        DB LOW PROG1
        ...
        DB LOW PROG7
PROG0:  ...                 ;прикладні програми
        ...
PROG7:  ...
```

Опитування групи імпульсних датчиків. Ця процедура складається з послідовності дій: очікування замикання одного з контактів, усунення брязкоту, очікування розмикання замкнутого контакту. Програмна реалізація процедури (KBRD) для випадку підключення чотирьох імпульсних датчиків до входів 0 - 3 порту 1 буде мати вигляд:

```
KBRD:   MOV A, P1   ;введення коду
        CPL A       ;інверсія коду
        ANL A, #0x0F;перевірка замкнутості контакту
        JZ  KBRD    ;якщо не один контакт не замкнутий
                    ;то чекати
        MOV R2, A   ;передача прийнятого коду у R2
DBNC:   CALL DELAY  ;усунення брязкоту
WAIT:   MOV A, P1   ;введення коду
        CPL A       ;інверсія коду
        ANL A, #0x0F;перевірка замкнутості контакту
        JNZ WAIT    ;якщо контакт замкнутий, то чекати,
EXIT:   ...         ;інакше вихід з процедури
```

Аналіз стану контактів здійснюється накладенням маски на прийнятий від датчиків код. Для датчиків, що формують «негативний» імпульс, прийнятий код заздалегідь інвертується.

Для групи імпульсних датчиків, які представляють собою клавішний регістр, процедура KBRD повинна бути доповнена

процедурами ідентифікації натиснутої клавіші і захисту від одночасного натискання двох і більше клавіш.

Ідентифікація натиснутої клавіші може здійснюватися двома способами: по таблиці або програмно. При табличному способі перекодування в пам'яті програм повинна знаходитися таблиця довічних еквівалентів кодів клавіш. Програмне перетворення унітарної коду, прийнятого від клавіатури, в двійковий код може бути виконано методом зрушень вихідного унітарного коду і підрахунком числа зрушень на лічильнику до появи першого перенесення.

3.5 Завдання для практичної роботи

Відповісти на наступні питання:

1. Як організувати процедуру очікування події за допомогою однієї команди?
2. Які обмеження накладаються на тривалість виявленого імпульсного сигналу при програмній реалізації циклу очікування?
3. Поясніть принципи усунення брязкоту контактів.
4. Поясніть принцип організації процедур, які підраховують число імпульсів між двома подіями і за заданий проміжок часу.
5. Поясніть принцип організації передачі управління в програмі за кодом.
6. У чому полягає табличний спосіб генерації мікро контролером складних послідовностей сигналів?

Таблиця 3.1 – Завдання для практичної роботи

Варіант	Завдання до роботи

Варіант	Завдання до роботи
	1. Написати підарограму очікування статичного сигналу вказаного рівня для вказаного піну. 2. Написати підарограму очікування імпульсного сигналу вказаної полярності для вказаного піну. 3. Написати підарограми для усунення брязкоту контактів двома способами для вказаного піну. 4. Написати підарограми для підрахунку числа імпульсів між двома вказаними подіями та за проміжок часу. 5. Написати підарограму для очікування вказаної комбінації на вказаному порті. 6. Написати підарограму для передачі керування за вказаним кодом. 7. Написати підарограму для очікування вказаної комбінації для вказаної кількості пінів.
1	1. Очікування «логічної одниниці» на піні 4, порту 1 2. Очікування «позитивного імпульсу» на піні 1, порту 2 3. Усунення брязкоту на піні 6, порту 3 4. Підрахувати число імпульсів між логічною одиницею на на піні 1, порту 1 та логічною одиницею на піні 3, порту 1 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 молодших пінах порту 0

Варіант	Завдання до роботи
2	1. Очікування «логічного нуля» на піні 3, порту 2 2. Очікування «негативного імпульсу» на піні 5, порту 1 3. Усунення брязкоту на піні 7, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 7, порту 2 та логічною одиницею на на піні 3, порту 3 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 1–5 порту 1
3	1. Очікування «логічної одниниці» на піні 6, порту 2 2. Очікування «позитивного імпульсу» на піні 4, порту 1 3. Усунення брязкоту на піні 3, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 1, порту 1 та логічною одиницею на на піні 3, порту 3 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 старших пінах 2 порту
4	1. Очікування «логічного нуля» на піні 2, порту 1 2. Очікування «негативного імпульсу» на піні 3, порту 3 3. Усунення брязкоту на піні 4, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 3, порту 1 та логічною одиницею на на піні 0, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 2–6 порту 3

Варіант	Завдання до роботи
5	1. Очікування «логічної одниниці» на піні 5, порту 0 2. Очікування «позитивного імпульсу» на піні 7, порту 1 3. Усунення брязкоту на піні 3, порту 2 4. Підрахувати число імпульсів між логічною одиницею на на піні 7, порту 0 та логічною одиницею на на піні 1, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 6 молодших пінах 0 порту
6	1. Очікування «логічного нуля» на піні 5, порту 0 2. Очікування «негативного імпульсу» на піні 7, порту 2 3. Усунення брязкоту на піні 4, порту 1 4. Підрахувати число імпульсів між логічною одиницею на на піні 2, порту 1 та логічною одиницею на на піні 2, порту 1 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 3–7 порту 1
7	1. Очікування «логічної одниниці» на піні 5, порту 1 2. Очікування «позитивного імпульсу» на піні 2, порту 2 3. Усунення брязкоту на піні 2, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 4, порту 0 та логічною одиницею на на піні 5, порту 1 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 6 старших пінах 2 порту

Варіант	Завдання до роботи
8	1. Очікування «логічного нуля» на піні 2, порту 0 2. Очікування «негативного імпульсу» на піні 4, порту 1 3. Усунення брязкоту на піні 5, порту 2 4. Підрахувати число імпульсів між логічною одиницею на на піні 1, порту 2 та логічною одиницею на піні 3, порту 5 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 молодших пінах порту 3
9	1. Очікування «логічної одниниці» на піні 2, порту 1 2. Очікування «негативного імпульсу» на піні 4, порту 2 3. Усунення брязкоту на піні 3, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 5, порту 1 та логічною одиницею на піні 4, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 1–5 порту 0
10	1. Очікування «логічного нуля» на піні 5, порту 1 2. Очікування «позитивного імпульсу» на піні 3, порту 1 3. Усунення брязкоту на піні 2, порту 2 4. Підрахувати число імпульсів між логічною одиницею на на піні 4, порту 2 та логічною одиницею на піні 2, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 старших пінах 1 порту

Варіант	Завдання до роботи
11	1. Очікування «логічної однинициі» на піні 1, порту 3 2. Очікування «негативного імпульсу» на піні 7, порту 2 3. Усунення брязкоту на піні 6, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 5, порту 1 та логічною одиницею на на піні 5, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 2–6 порту 2
12	1. Очікування «логічного нуля» на піні 6, порту 0 2. Очікування «позитивного імпульсу» на піні 1, порту 2 3. Усунення брязкоту на піні 4, порту 1 4. Підрахувати число імпульсів між логічною одиницею на на піні 4, порту 1 та логічною одиницею на піні 2, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 6 молодших пінах 3 порту
13	1. Очікування «логічної однинициі» на піні 4, порту 1 2. Очікування «негативного імпульсу» на піні 5, порту 1 3. Усунення брязкоту на піні 3, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 2, порту 0 та логічною одиницею на піні 1, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 3–7 порту 0

Варіант	Завдання до роботи
14	1. Очікування «логічного нуля» на піні 2, порту 0 2. Очікування «позитивного імпульсу» на піні 7, порту 2 3. Усунення брязкоту на піні 5, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 2, порту 1 та логічною одиницею на піні 4, порту 1 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 молодших пінах порту 2
15	1. Очікування «логічної одниниці» на піні 6, порту 1 2. Очікування «негативного імпульсу» на піні 7, порту 0 3. Усунення брязкоту на піні 2, порту 2 4. Підрахувати число імпульсів між логічною одиницею на на піні 7, порту 3 та логічною одиницею на піні 3, порту 3 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 1–5 порту 3
16	1. Очікування «логічного нуля» на піні 5, порту 0 2. Очікування «позитивного імпульсу» на піні 6, порту 2 3. Усунення брязкоту на піні 4, порту 1 4. Підрахувати число імпульсів між логічною одиницею на на піні 1, порту 0 та логічною одиницею на піні 5, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 5 старших пінах 0 порту

Варіант	Завдання до роботи
17	1. Очікування «логічної однинициі» на піні 2, порту 2 2. Очікування «негативного імпульсу» на піні 2, порту 1 3. Усунення брязкоту на піні 2, порту 0 4. Підрахувати число імпульсів між логічною одиницею на на піні 3, порту 1 та логічною одиницею на піні 0, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на пінах 2–6 порту 1
18	1. Очікування «логічного нуля» на піні 4, порту 0 2. Очікування «позитивного імпульсу» на піні 5, порту 1 3. Усунення брязкоту на піні 6, порту 2 4. Підрахувати число імпульсів між логічною одиницею на на піні 7, порту 3 та логічною одиницею на піні 0, порту 0 5. Комбінація 0xNN на першому порті (де NN – номер варіанту) 6. Передача керування підпрограмі PROGO_N, де N – остання цифра номеру варіанта 7. Комбінація 0xNN на 6 молодших пінах 2 порту

4. ПРАКТИЧНА РОБОТА №4
«РЕАЛІЗАЦІЯ ФУНКЦІЙ ЧАСУ»

У базових моделях сімейства є два програмованих 16- бітних таймера/лічильника (T/C0 і T/C1), які можуть бути використані як в якості таймерів, так і в якості лічильників зовнішніх подій. У першому випадку вміст відповідного таймера / лічильника (далі для стислості T/C) інкременується в кожному машинному циклі, тобто через кожні 12 періодів коливань кварцевого резонатора, у другому воно інкременується під впливом переходу з 1 в 0 зовнішнього вхідного сигналу, що подається на відповідний (T0, T1) висновок мікро-ЕОМ 8051. Опитування значення зовнішнього вхідного сигналу виконується в момент часу S5P2 кожного машинного циклу.

Таблиця 4.1 – Регістр режиму роботи таймера / лічильника TMOD

Символ	Позиція	Ім'я та призначення		
GATE	TMOD.7 для T/C1 і TMOD.3 для T/CO	Управління блокуванням. Якщо біт встановлено, то таймер / лічильник "x" дозволений до тих пір, поки на вході "lNTx" високий рівень і біт управління "TRx" встановлено. Якщо біт скинутий, то T/C дозволяється, як тільки біт керування "TRx" встановлюється		
C/T	TMOD.6 для T/C1 і TMOD.2 для T/CO	Біт вибору режиму таймера або лічильника подій. Якщо біт скинутий, то працює таймер від внутрішнього джерела сигналів синхронізації. якщо встановлено, то працює лічильник від зовнішніх сигналів на вході "Tx"		
M1	TMOD.5 для T/C1 і TMOD.1 для T/CO	Режим роботи		
		M1	M0	
		0	0	"TLx" працює як 5- бітний переддільник
		0	1	16 бітний таймер / лічильник. "THx" і "TLx" включений послідовно
M0	TMOD.4 для T/C1 і TMOD.0 для T/CO	1	0	8- бітний авто перезавантажуваний таймер / лічильник. "THx" зберігає значення, яке має бути перезавантажене в "TLx" кожен раз по переповненню
		1	1	Таймер / лічильник 1 зупиняється. Таймер / лічильник 0: TLO працює як 8- бітний таймер / лічильник, і його режим визначається керуючими бітами таймера 0. TH0 працює тільки як 8 бітний таймер, і його режим визначається керуючими бітами таймера 1

Вміст лічильника буде збільшено на 1 в тому випадку, якщо в попередньому циклі був лічений вхідний сигнал високого рівня (1), а в наступному - сигнал низького рівня (0). Нове (інкрементувати)

значення лічильника буде сформовано в момент S3P3 в циклі, наступному за тим, в якому був виявлений перехід сигналу з 1 в 0. Так як на розпізнавання періоду потрібні два машинних цикли, максимальна частота підрахунку вхідних сигналів дорівнює 1/24 частоти резонатора . На тривалість періоду вхідних сигналів обмежень зверху немає.

Для гарантованого прочитання вхідний сигнал повинен утримувати значення 1, як мінімум, протягом одного машинного циклу мікро ЕОМ.

Для управління режимами роботи Т/С і для організації їх взаємодії з системою переривань використовуються два регістри спеціальних функцій (TMOD і TCON), опис яких наведено в табл. 4.1 і 4.2 відповідно.

Таблиця 4.2– Регістр керування / статусу таймера TCON.

Символ	Позиція	Ім'я та призначення
TF1	TCON.7	Флаг переповнення таймера 1. Встановлюється апаратно при переповненні таймера / лічильника. Скидається при обслуговуванні переривання апаратно
TR1	TCON.6	Біт управління таймера 1. Встановлюється, / скидається програмою для пуску / зупинки
TF0	TCON.5	Флаг переповнення таймера 0. Встановлюється апаратно. Скидається при обслуговуванні переривання
TR0	TCON.4	Біт управління таймера 0. Встановлюється / скидається програмою для пуску / зупинки таймера / лічильника
IE1	TCON.3	Флаг фронту переривання 1. Встановлюється апаратно, коли детектується зріз зовнішнього сигналу INT1. Скидається при обслуговуванні переривання
IT1	TCON.2	Біт управління типом переривання 1. Встановлюється / скидається програмно для специфікації запиту INT1 (зріз / низький рівень)
IE0	TCON.1	Флаг фронту переривання 0. Встановлюється по зрізі сигналу INT0. Скидається при обслуговуванні переривання
IT1	TCON .0	Біт управління типом переривання 0. Встановлюється / скидається програмно для специфікації запиту INT0 (зріз / низький рівень)

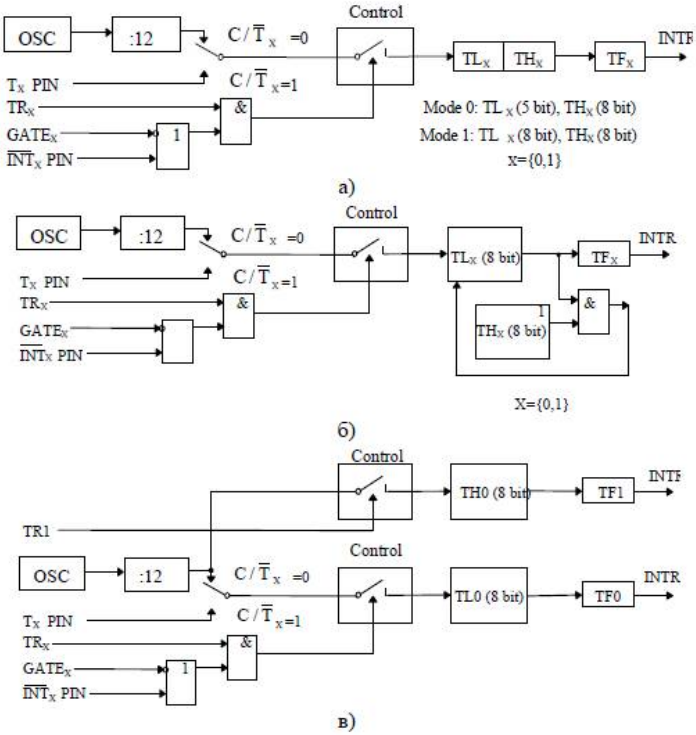
4.1 Режими роботи таймерів-лічильників

Як випливає з опису керуючих біт TMOD, для обох Т / З режими роботи 0, 1 і 2 однакові. Режими 3 для Т/З і Т/С1 різні. Розглянемо коротко роботу Т / С в кожному з режимів [1].

- Режим 0. Переклад будь-якого Т/С в режим 0 робить його схожим на таймер - восьми бітний лічильник, до входу якого підключений 5 бітний переддільник частоти на 32. Роботу Т/С в режимі 0 на прикладі Т/С1 ілюструє рис 12, а. В цьому режимі таймерний регістр має розрядність 13 біт. При переході зі стану "всі одиниці" в стан "всі нулі" встановлюється прапор переривання від таймера TF. Вхідний синхросигнал таймера 1 дозволений (надходить на вхід Т/С1), коли керуючий біт TR1 встановлений в 1 або керуючий біт GATE (блокування) дорівнює 0, або на зовнішній висновок запиту переривання INT1 надходить рівень 1. Відзначимо попутно, що установка біта GATE в 1 дозволяє використовувати таймер для вимірювання тривалості імпульсного сигналу подається на вхід запиту переривання.
- Режим 1. Робота будь-якого Т/С в цьому режимі така ж, як і в режимі 0, за винятком того, що таймерний регістр має розрядність 16 біт.
- Режим 2. У цьому режимі робота організована таким чином, що переповнення (перехід зі стану "всі одиниці" в стан, "всі нулі") 8-бітного лічильника TL1 приводить не тільки до установки прапора TF1 (див. Рис. 4.1, б), а й автоматично перезавантажує в TL1 вміст старшого байта (TH 1) таймерного регістра, яке попередньо було задано програмним шляхом. Перевантаження залишає вміст TH1 незмінним. У режимі 2 Т/С0 і Т/С1 також працюють абсолютно однаково.
- Режим 3. У режимі 3 Т/С0 і Т/С1 працюють по-різному. Т/С1 зберігає незмінним свій поточний зміст. Іншими словами, ефект такий же як і при скиданні керуючого біта TR1 в 0. Роботу Т/С0 ілюструє рис. 4.1. У режимі 3 TL0 і TH0 функціонують як два незалежних 8-бітових лічильника. Роботу TL0 визначають керуючі біти Т / С0 (С / Т, GATE TR0), вхідний сигнал INT0 і прапор переповнення TF0. Роботу TH0, який може виконувати тільки функції таймера (підрахунок машинних циклів мікро-ЕОМ), визначає керуючий біт TR1. При цьому TH0 використовує прапор переповнення TF1. Режим 3 використовується в тих випадках, коли потрібна наявність додаткового восьми бітного таймера або лічильника подій. Можна вважати, що в цьому режимі мікро-ЕОМ 8051 має в своєму складі три таймера / лічильника. У разі ж, якщо Т / С0 використовується в 38 режимі 3,

Т / С1 може бути або вимкнений, або переведений в режим 0, 1 або 2, або може бути використаний послідовним портом як генератор частоти передачі.

У модернізованих моделях мікроконтролерів сімейства MCS-51 може бути третій таймер лічильник Т/С2 і (або) блок програмних лічильників РСА, які теж можуть бути використані для відліку часових інтервалів.



а – Т/С0 та Т/С1 у режимах 0 та 1; б – Т/С0 та Т/С1 у режимі 2; в – Т/С0 у режимі 3.

Рисунок 4.1 – таймери-лічильники Т/С0 та Т/С1 у режимах 0, 1, 2 та 3.

4.2 Реалізація функцій часу

4.2.1 Програмне формування тимчасової затримки

Затримка малої тривалості. Процедура реалізації тимчасової затримки використовує метод програмних циклів. При цьому в певний робочий регістр завантажується число, яке потім в кожному проході циклу зменшується на 1. Так триває до тих пір, поки вміст робочого регістра не стане рівним нулю, що інтерпретується програмою як момент виходу з циклу. Час затримки при цьому визначається числом, завантаженим в робочий регістр, і часом виконання команд, що утворюють програмний цикл.

Припустимо, що в керуючій програмі необхідно реалізувати тимчасову затримку 99 мкс. Фрагмент програми, що реалізує тимчасову затримку, потрібно оформити у вигляді підпрограми, так як передбачається, що основна керуюча програма буде виробляти до неї багаторазові звернення для формування вихідних імпульсних сигналів, тривалість яких кратна 99 мкс:

```
DELAY:  MOV R2, #X      ;загрузка числа X
COUNT:  DJNZ R2, COUNT  ;декремент R2 и цикл, якщо не рівне 0
RET                     ;возврат
```

Для отримання необхідної тимчасової затримки необхідно визначити число X, завантажувати в робочий регістр. Визначення числа X виконується на основі розрахунку часу виконання команд, що утворюють дану підпрограму. При цьому необхідно враховувати, що команди MOV і RET виконуються одноразово, а число повторень команди DJNZ дорівнює числу X. Крім того, звернення до підпрограми тимчасової затримки здійснюється по команді CALL DELAY, время виконання якої також необхідно враховувати при підрахунку невеликою затримкою.

В описі команд мікроконтролера вказується, за скільки машинних циклів (МЦ) виконується кожна команда. На підставі цих даних визначається сумарна кількість машинних циклів в підпрограмі: CALL - 2 МЦ, MOV - 1 МЦ, DJNZ - 2 МЦ, RET - 2 МЦ.

На основі цих данних визначається сумарна кількість машинних циклів в підпрограмі. При тактовій частоті 12 МГц кожен машинний цикл виконується за 1 мкс. Таким чином, підпрограма виконується за

час 2 + 1 + 2 X + 2 = 5 + 2 X мкс. Для реалізації тимчасової затримки 99 мкс число X = (99 - 5)/ 2 = 47.

В даному випадку при завантаженні в регістр R2 числа 47 необхідна тимчасова затримка (99 мкс) реалізується точно. Якщо число X виходить дробовим, то тимчасову затримку можна реалізувати лише приблизно. Для більш точного підстроювання в підпрограму можуть бути включені команди NOP, час виконання кожної з яких дорівнює 1 мкс.

Мінімальна тимчасова затримка, реалізована підпрограмою DELAY, становить 7 мкс (X = 1).

Тимчасову затримку меншої тривалості програмним шляхом можна реалізувати, включаючи в програму ланцюжка команд NOP.

Максимальна тривалість затримки, що реалізується підпрограмою DELAY, становить 515 мкс (X = 255)

Для реалізації затримки більшої тривалості можна рекомендувати збільшити тіло циклу включенням додаткових команд або використовувати метод вкладених циклів. Так, наприклад, якщо в підпрограму DELAY перед командою DJNZ вставити додатково дві команди NOP, то максимальна затримка складе 5 + X (2 + 1) = 5 + 3 * 255 = 770 мкс (тобто майже в півтора рази більше).

Затримка великої тривалості. Затримка великої тривалості може бути реалізована шляхом вкладених циклів. Числа X і Y вибираються з співвідношення T = 2 + 1 + X (1 + 2Y + 2) + 2, де T - реалізований часовий інтервал в мікросекундах. Максимальний часовий інтервал, який реалізується таким способом, при X = Y = 255 становить 130.82 мс, тобто приблизно 0.13 с.

Як приклад розглянемо підпрограму, що реалізує тимчасову затримку 100 мс:

Тут два вкладених цикла реалізують тимчасову затримку тривалістю 5 + 195 (3 + 2 * 254) = 99 650 мкс, а додатковий цикл LOOPAD і команда NOP реалізує затримку 350 мкс і тим самим забезпечує точну настройку тимчасового інтервалу.

```
DLY100: MOV     R1, #195    ;завантаження
LOOPEX: MOV     R2, #254    ;завантаження
LOOPIN: DJNZ    R2, LOOPIN  ;декремент R2 та внутрішній
                            ;цикл, якщо (R2) не ріне 0
        DJNZ    R1, LOOPEX  ;декремент R1 та внутрішній
                            ;цикл, якщо (R1) не ріне 0
```

```
        MOV     R3, #174    ;точне підлаштування
LOOPAD: DJNZ    R2, LOOPAD  ;часової
        NOP                 ;затримки
        RET                 ;повернення
```

Тимчасова затримка тривалістю 1 с. З розглянутого прикладу видно, що секунда є дуже великим інтервалом часу в порівнянні з тактовою частотою мікроконтролера. Такі затримки складно реалізувати методом вкладених циклів, тому їх зазвичай набирають з точно підстроєних затримок меншою тривалості. Наприклад, затримку в 1 с можна реалізувати десятикратним викликом підпрограми, що реалізує затримку 100 мс:

```
DLY1:   MOV R4, #10         ;загрузка в R4 числа визовів DE-LAY
LOOP:   CALL DLY100         ;затримка 100мс
       DJNZ R4, LOOP;R4 - 1 і цикл, якщо (R4) не равно 0
```

Похибка підпрограми становить 21 мкс. Для дуже багатьох застосувань це досить висока точність, хоча реалізовані на основі цієї програми годинники астрономічного часу за добу "втечуть" приблизно на 1.8 с.

4.2.2 Формування часової затримки таймером

Затримка малої тривалості. Недоліком програмного способу реалізації тимчасової затримки є нераціональне використання ресурсів мікроконтролера: під час формування затримки він практично простоює, так як не може вирішувати ніяких завдань управління об'єктом. У той же час апаратурні засоби дозволяють реалізувати тимчасові затримки на фоні основної програми роботи.

На вхід таймера / лічильника (T/C) можуть надходити сигнали синхронізації з частотою 1 МГц (T/C в режимі таймера) або сигнали від зовнішнього джерела (T/C в режимі лічильника). Обидва ці режиму можуть бути використані для формування затримок. Якщо використовувати T/C в режимі таймера повного формату (16 біт), то можна отримати затримки в діапазоні 1 - 65 536 мкс.

Як приклад розглянемо організацію тимчасової затримки 50 мс (TIMER). Передбачається, що біт IE.7 встановлено, тобто знято блокування всіх переривань. Зверніть увагу, що тут використана

команда переводу мікроконтролера в режим холостого ходу, який припиняється після закінчення 50 мс. Цей режим реалізований в мікроконтролері Intel 80C51 (вітчизняний аналог КР1830ВЕ51), виготовлений за технологією КМОП [3]. Тому для налагодження цієї програми через пункт Debug меню Options виберіть вказаний тип мікроконтролера.

```
;Організація переходу до мітки NEXT при переповненні T/C0
       ORG 0x0B                    ;адреса вектора
переривання від T/C0
       CLR TCON.4                  ;зупинка T/C0
       RETI                        ;вихід з підпрограми
                                   ;обробки переривання
       ORG 30                      ;початкова адреса
програми
TIMER: MOV TMOD, #0x01             ;налаштування T/C0
       MOV TL0,  #LOW(NOT(50000-1)) ;завантаження таймеру для
       MOV TH0,  #HIGH(NOT(50000-1));затримки у 50 мс
       SETB    TCON.4              ;старт T/C0
       SETB    IE.1                ;дозвіл переривання від
T/C0
       SETB    PCON.0              ;перехід у режим
холостого ходу
NEXT:  ...                         ;
```

4.3 Вимірювання часових інтервалів

У завданнях управління часто виникає необхідність вимірювання проміжку часу між двома подіями. Зазвичай події в об'єкті управління представляються сигналами від довічних датчиків. Вважаючи подіями фронт і спад імпульсу, можна визначати тимчасові характеристики імпульсних сигналів: тривалість, період і шпаруватість.

Найпростішим способом виміру тривалості імпульсу є програмний. Для виявлення подій (фронт і спад імпульсного сигналу) в цьому випадку використовуються типові процедури WAIT, а відлік часу ведеться програмним способом. Для "позитивного" імпульсного сигналу, що надходить на вхід T0, програма вимірювання його тривалості матиме вигляд:

```
MSCONT: MOV R7, #0       ;Скидання лічильника
```

```
WAIT0:  JNB T0, WAIT0   ;Очікування фронту сигналу
COUNT:  INC R7          ;Інкремент лічильника
        JB  T0, COUNT   ;очікування спаду сигналу
EXIT:   . . .           ;Вихід з процедури
```

Після виходу з процедури вміст лічильника R7 пропорційно тривалості імпульсу.

Для нормальної роботи цієї програми необхідно, щоб звернення до неї вироблялося в моменти, коли на вході T0 присутній сигнал нульового рівня. Верхня межа вимірюваної тривалості «позитивного» імпульсу складе 255*(1 + 2) мкс = 765 мкс. Ця межа може бути збільшена включенням в цикл COUNT додаткових команд NOP. Максимальна похибка вимірювань 3 мкс.

Вимірювання часових інтервалів. Застосування таймера. Для вимірювання тривалості сигналу може бути використаний таймер. Особливо ефективно використання для цієї мети таймера в 8051 Intel, має вхід дозволу рахунку (альтернативна функція входу INT). Вимірюваний сигнал можна, наприклад, подавати на вхід INT0, а вимір тривалості при цьому буде виконуватися в T/C0. Програма вимірювання тривалості «позитивного» імпульсу (MSTIMER) буде виглядати так:

```
MSTIMER:    MOV TMOD, #0x09 ;Настройка T/C0
            MOV TH0,  #0    ;Скидання таймера
            MOV TL0,  #0
            SETB TCON.4     ;Старт T/C0
WAIT0:      JNB P3.2, WAIT0 ;Очікування 1
WAITC:      JB  P3.2, WAITC ;Очікування 0
            CLR TCON.4      ;Стоп T/C0
EXIT:       ...             ;Вихід з процедури
```

Управління програмою повинно бути передано за умови, що на вході INT0 присутній низький рівень. Переривання від T/C0 і зовнішнє переривання по входу INT0 повинні бути заборонені.

По завершенні програми в T/C0 буде знаходитися число, пропорційне тривалості «позитивного» імпульсу на вході INT0. Верхня межа вимірювання дорівнює 65 536 мкс, а максимальна похибка 1 мкс.

При необхідності вимірювання тимчасових інтервалів більшої тривалості можна програмним способом підраховувати число переповнень від таймера, тобто розширювати його розрядність за рахунок робочого регістра або комірки резидентної пам'яті даних.

4.4 Завдання для практичної роботи

Відповісти на наступні питання:

1. З якою частотою інкрементується вміст таймера/лічильника при роботі в якості таймера?

2. Чому дорівнює максимальна частота підрахунку вхідних сигналів при роботі таймера/лічильника в режимі лічильника?

3. Охарактеризуйте режими роботи таймера/лічильника.

4. Як організувати процедуру очікування події за допомогою однієї команди?

5. Як програмно і апаратно формуються затримки різної тривалості?

6. Як за допомогою мікроконтролера виміряти часовий інтервал і оцінити точність та діапазон виміру?

7. Розшифруйте команду MOV TMOD, #01000000B.

Таблиця 4.3 – Завдання для практичної роботи

Варіант	Завдання до роботи
	1. Написати підараграму для затримки малої тривалості на вказаний інтервал часу. 2. Написати підараграму для затримки великої тривалості на вказаний інтервал часу. 3. Написати підараграму для затримки вказаної тривалості за допомогою таймера. 4. Написати підараграму для виміру тривалості імпульсу заданої полярності на вказаному піні програмним способом. 5. Написати підараграму для виміру тривалості імпульсу заданої полярності на вказаному піні за допомогою таймера.
1	1. Написати підараграму для затримки на 90 мкс 2. Написати підараграму для затримки на 90 мс 3. Написати підараграму для затримки на 23 мс 4. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера

Варіант	Завдання до роботи
2	1. Написати підарограму для затримки на 97 мкс 2. Написати підарограму для затримки на 85 мс 3. Написати підарограму для затримки на 26 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
3	1. Написати підарограму для затримки на 96 мкс 2. Написати підарограму для затримки на 98 мс 3. Написати підарограму для затримки на 27 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
4	1. Написати підарограму для затримки на 92 мкс 2. Написати підарограму для затримки на 170 мс 3. Написати підарограму для затримки на 13 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
5	1. Написати підарограму для затримки на 117 мкс 2. Написати підарограму для затримки на 57 мс 3. Написати підарограму для затримки на 21 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера

Варіант	Завдання до роботи
6	1. Написати підарограму для затримки на 112 мкс 2. Написати підарограму для затримки на 98 мс 3. Написати підарограму для затримки на 18 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
7	1. Написати підарограму для затримки на 119 мкс 2. Написати підарограму для затримки на 112 мс 3. Написати підарограму для затримки на 30 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
8	1. Написати підарограму для затримки на 121 мкс 2. Написати підарограму для затримки на 108 мс 3. Написати підарограму для затримки на 28 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
9	1. Написати підарограму для затримки на 122 мкс 2. Написати підарограму для затримки на 107 мс 3. Написати підарограму для затримки на 26 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера

Варіант	Завдання до роботи
10	1. Написати підараграму для затримки на 94 мкс 2. Написати підараграму для затримки на 70 мс 3. Написати підараграму для затримки на 37 мс 4. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
11	1. Написати підараграму для затримки на 118 мкс 2. Написати підараграму для затримки на 127 мс 3. Написати підараграму для затримки на 23 мс 4. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
12	1. Написати підараграму для затримки на 103 мкс 2. Написати підараграму для затримки на 75 мс 3. Написати підараграму для затримки на 44 мс 4. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
13	1. Написати підараграму для затримки на 137 мкс 2. Написати підараграму для затримки на 67 мс 3. Написати підараграму для затримки на 42 мс 4. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підараграму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера

Варіант	Завдання до роботи
14	1. Написати підарограму для затримки на 145 мкс 2. Написати підарограму для затримки на 118 мс 3. Написати підарограму для затримки на 33 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
15	1. Написати підарограму для затримки на 115 мкс 2. Написати підарограму для затримки на 85 мс 3. Написати підарограму для затримки на 38 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера
16	1. Написати підарограму для затримки на 100 мкс 2. Написати підарограму для затримки на 77 мс 3. Написати підарограму для затримки на 55 мс 4. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 5 порту 3 програмним шляхом 5. Написати підарограму для виміру тривалості «позитивного» імпульсу на піні 4 порту 2 за допомогою таймера

5. ПРАКТИЧНА РОБОТА №5 «СИСТЕМА ПЕРЕРИВАНЬ ТА ТАЙМЕРИ/ЛІЧИЛЬНИКИ МІКРОКОНТРОЛЕРА 8051»

5.1 Система переривань мікроконтролера 8051

Спрощена схема переривань мікро-ЕОМ 8051 показана на рис. 5.1.

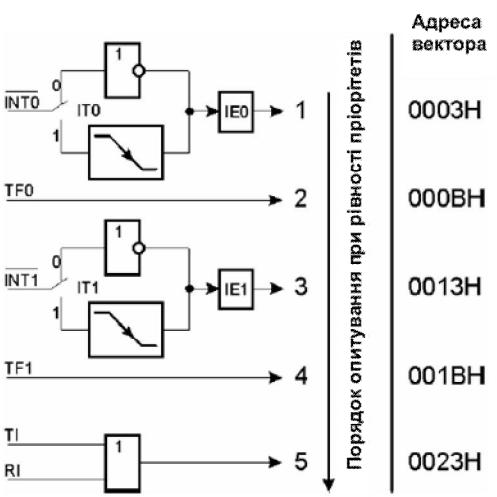


Рисунок 5.1 – Схема переривань

Зовнішні переривання INT 0 і INT 1 можуть бути викликані або рівнем, або переходом сигналу з 1 в 0 на входах 8051 в залежності від значень керуючих біт IT0 і IT1 в регістрі TCON. Від зовнішніх переривань встановлюються прапори IE0 і IE1 в регістрі TCON, які ініціюють виклик відповідної програми обслуговування переривання. Скидання цих прапорів виконується апаратно тільки в тому випадку, якщо переривання було викликано по переходу (зрізу) сигналу. Якщо ж переривання викликано рівнем вхідного сигналу, то скиданням прапора I повинна управляти відповідна підпрограма обслуговування переривання шляхом впливу на джерело переривання з метою зняття ним запиту.

Прапори запитів переривання від таймерів TF0 і TF1 скидаються автоматично при передачі управління підпрограмі обслуговування. Прапори запитів переривання RI і TI встановлюються блоком управління приймача апаратно, але скидатися повинні програмним шляхом. Переривання можуть бути викликані або скасовані програмою, так як всі названі прапори програмно доступні і можуть бути Рис. 5.1. Схема переривань встановлені / скинуті програмою з тим же результатом, як би вони були встановлені / скинуті апаратними засобами.

У блоці регістрів спеціальних функцій є два регістри, призначених для управління режимом переривань IE і рівнями пріоритету IP, описані в таблиці 5.1 і 5.2 відповідно.

Можливість програмної установки / скидання будь-якого керуючого біта в цих двох регістрах робить систему переривань 8051 виключно гнучкою. У більш складних модифікаціях мікроконтролерів сімейства MCS-51 кількість периферійних пристроїв збільшено, що призводить до необхідності використовувати один вектор переривання для декількох пристроїв (поділ підпрограм обслуговування переривань в цьому випадку необхідно реалізувати програмно), або додати ще два регістри - режиму (маски) і пріоритету переривань.

Таблиця 5.1 – Регістр масок переривання (IE)

Символ	Розряд	Ім'я та призначення
EA	IE.7	Розблокування переривань. Скидається програмно для заборони всіх переривань незалежно від станів IE4-IE0
	IE.6	Не використовуються
	IE.5	Не використовуються
ES	IE.4	Біт дозволу переривання від UART. Установка / скидання програмою для дозволу / заборони переривань від прапорів TI, RI
ET1	IE.3	Біт дозволу переривання від таймера 1. Установка / скидання програмою для дозволу / заборони переривань від таймера 1
EX1	IE.2	Біт дозволу зовнішнього переривання 1. Установка / скидання програмою для дозволу / заборони переривань
ET0	IE.1	Дозвіл переривання від таймера 0. Працює аналогічно IE.3
EX0	IE.0	Дозволи зовнішнього переривання 0. Працює аналогічно IE.2

Таблиця 5.2 – Регістр пріорітетів переривань

Символ	Розряд	Ім'я та призначення
-	IP.7- IP.5	Не використовується
PS	IP.4	Біт пріоритету UART. Установка / скидання програмою для призначення переривання від UART вищого / нижчого пріоритету

PT1	IP.3	Біт пріоритету таймера 1. Установка / скидання програмою для призначення переривання від таймера 1 вищого / нижчого пріоритету
PX1	IP.2	Біт пріоритету зовнішнього переривання 1. Установка / скидання програмою для призначення переривання INT1 вищого / нижчого пріоритету
PT0	IP.1	Біт пріоритету таймера 0. Працює аналогічно IP.3
PX0	IP.0	Пріоритет зовнішнього переривання 0. Працює аналогічно IP.2

5.2 Виконання підпрограми переривання

Система переривань формує апаратний виклик (LCALL) відповідної підпрограми обслуговування, якщо вона не заблокована одним з наступних умов:

- в даний момент обслуговується запит переривання рівного або високого рівня пріоритету;
- поточний машинний цикл − не останній в циклі виконуваної команди;
- виконується команда RETI або будь-яка команда, пов'язана з обігом до регістрів IE або IP.

Відзначимо, що якщо прапор переривання був встановлений, але по одному із зазначених вище умов не отримав обслуговування і до моменту закінчення блокування вже скинутий, то запит переривання втрачається і ніде не запам'ятовується.

За апаратно сформованого коду LCALL система переривання поміщає в стек тільки вміст лічильника команд (PC) і завантажує в нього адреса вектора відповідної підпрограми обслуговування. За адресою вектора повинна бути розташована команда безумовної передачі керування (JMP) до початкової адреси підпрограми обслуговування переривання. У разі необхідності вона повинна починатися командами запису в стек (PUSH) слова стану програми (PSW), акумулятора, розширювача, покажчика даних і т.д. і повинна закінчуватися командами відновлення зі стека (POP).

Підпрограми обслуговування переривання повинні завершуватися командою RETI, по якій в лічильник команд перезавантажується з стека збережена адреса повернення в основну програму. Команда RET також повертає управління перерваної основній програмі, але при цьому не зніме блокування переривань, що приводить до необхідності мати програмний механізм аналізу закінчення процедури обслуговування даного переривання.

5.3 Переривання

Підпрограма обробки переривання повинна зберегти в стеці вміст тих регістрів, які будуть в ній використані, а перед поверненням в перервану програму повинна відновити їх значення. Підпрограма обробки зовнішнього переривання рівня 0 (SUBINO) може, наприклад, мати наступну структуру:

```
        ORG 0x3          ;задати адресу вектора переривання
        SJMP SUBINO      ;перехід на підпрограму обробки
        ORG 0x30
SUBINO:
        PUSH ACC         ;збереження акумулятора
        PUSH PSW         ;збереження в стеку PSW
        PUSH B           ;збереження B
        PUSH DPL         ;збереження DPTR і ,можливо ,
        PUSH DPH         ;інших регістрів
        MOV PSW, #0x8    ;вибір регістрів банка 1
        . . .            ;безпосередньо обробка переривання
        POP DPH          ;відновлення DPH
        POP DPL          ;відновлення DPL
        POP B            ;відновлення B
        POP PSW          ;відновлення PSW
        POP ACC          ;відновлення ACC
        RETI             ;повернення
END
```

Зверніть увагу, що обробник закінчується командою повернення з підпрограми RETI, а перехід на нього відбувається по команді безумовного переходу SJMP. Справа в тому, що система переривань формує виклик LCALL апаратно [3].

З метою налагодження програму необхідно доповнити наступними командами:

```
        ORG 0H
        SJMP START
        ORG 20H      ;без наступних команд можна обійтись,
якщо встановити
                     ;ці біти безпосередньо у вікні Main
Registers
START:  SETB IT0     ;тип преривання 0 – «по зрізу»
        SETB EX0     ;дозвіл зовнішнього переривання 0
        SETB EA      ;зняття блокування переривання
START1: NOP
```

5.4.Вивід керуючих сигналів

Формування статичних сигналів

Для управління виконавчим пристроєм, що працює за принципом увімкнення / на відповідній вихідний лінії порту необхідно сформувати статичний сигнал 1 або 0, що реалізується командами установки, скидання або інверсії біта порту (OUT) [1].

У разі паралельного управління групою автономних виконавчих пристроїв, підключених до порту, формується не двійковий керуючий вплив, а керуюче слово, кожному з розрядів якого ставиться у відповідність 1 або 0 в залежності від того, які виконавчі пристрої повинні бути включені, а які вимкнені.

Керуючі слова зручно формувати командами логічних операцій над вмістом порту [1]. Командою XRL здійснюється інверсія тих бітів, які в масці задані одиницею. Команда ORL використовується для установки бітів слова. Команда ANL використовується для скидання бітів.

```
OUT:    MOV P1,#0xAA   ;вивід байта в порт P1
        SETB P1.0      ;установка нульового розряду
        CLR P1.3       ;скидання 3-го разряду
        CLR P1.4       ;скидання 4-го разряду
        CLR P1.5       ;скидання 5-го разряду
        CLR P1.6       ;скидання 6-го разряду
        CLR P1.7       ;скидання 7-го разряду
        XRL P1, #0xF0  ;інверсія разрядов 4 – 7
        ORL P1, #0x08  ;установка третього разряду
        ANL P1, #0xFE  ;скидання нульового разряду
```

Для формування складних послідовностей управляючих слів зазвичай використовують табличний спосіб, при якому всі можливі слова упаковані в таблицю, а прикладна програма обчислює адресу потрібного слова, вибирає його з таблиці і передає в порт.

Формування імпульсних сигналів

Імпульс. Його можна отримати послідовною видачею сигналів, включаючи і відключаючи (PULS, з проміжним викликом підпрограми тимчасової затримки:

```
PULS:                  ;Видача імпульсу в лінію 3 порту 1
ON:     SETB P1.3      ;включення
        CALL DELAY     ;Тимчасова затримка
```

```
OFF:    CLR  P1.3      ;відключення
        ...
```

Тривалість імпульсу визначається часовою затримкою, яка реалізується підпрограмою DELAY.

Генерація меандру. В цьому випадку можна скористатися підпрограмою затримки, яка дорівнює половині періоду сигналу:

```
MEANDR: CPL P1.3
        CALL DELAY
        SJMP MEANDR
```

Нескінченний періодичний сигнал формується в лінії 3 порту 1. На інших лініях сигнали залишаються незмінними.

Формування аперіодичних керуючих сигналів. Послідовність імпульсних сигналів з довільною тривалістю та скважністю може бути отримана аналогічним чином, тобто шляхом чергування процедур видачі значення 0 або 1 і виклику підпрограм тимчасових затримок заданих тривалостей.

5.5 Завдання для практичної роботи

Відповісти на наступні питання:

1. Перерахуйте і охарактеризуйте типи переривань. Як змінити пріоритети переривань?
2. Як задати вектор переривання в програмі на мові асемблера?
3. Чому в прикладі SUBINO перехід на обробник переривання здійснюється за командою SJMP, а не CALL?
4. Як співвідноситься порядок збереження вмісту регістрів в стеку з порядком відновлення зі стека в підпрограмі обробки переривання?
5. Чому підпрограма обробки переривання завершується командою RETI, а не RET?
6. Поясніть принцип генерації статичних, періодичних і аперіодичних сигналів.
7. Який час реакції мікроконтролера на подію при програмній реалізації процедури очікування і за перериванням?

Таблиця 5.3 – Завдання для практичної роботи

Вариант	Завдання до роботи
	1. Написати підарограму обробки переривання, що буде працювати з вказаними регістрами, вказаного банку регістрів 2. Написати підпрограму, що буде проводити вивід інкрементованого значення акумулятора у ваказаний порт із вказаною затримкою 3. Написати підпрограму, для генерації меандру обома приведеними вище способами, із вкащаним періодом
1	1. ACC, B, R0, R1, R4, R5, R6, R7, DPl, DPH, банк пам'яті 0 2. Виводити в порт 0 значення NN, 1, … 120, із інтервалом 10мс 3. Пін 3 порт 3 із періодом 25 мс
2	1. ACC, B, R0, R2, R3R5, R6, R7, DPl, DPH, банк пам'яті 1 2. Виводити в порт 1 значення NN, 1, … 127, із інтервалом 11мс 3. Пін 1 порт 2 із періодом 26 мс
3	1. ACC, B, R2, R3, R4, R5, R6, R7, DPl, DPH, банк пам'яті 2 2. Виводити в порт 2 значення NN, 1, … 118, із інтервалом 13мс 3. Пін 5 порт 2 із періодом 24 мс
4	1. R0, R1, R2, R3, R4, R5, R6, R7, DPl, DPH, банк пам'яті 3 2. Виводити в порт 3 значення NN, 1, … 108, із інтервалом 17мс 3. Пін 2 порт 1 із періодом 27 мс
5	1. ACC, B, R0, R1, R2, R3, R4, R5, R6, R7, банк пам'яті 0 2. Виводити в порт 0 значення NN, 1, … 150, із інтервалом 7мс 3. Пін 0 порт 2 із періодом 23 мс

Вариант	Завдання до роботи
6	1. ACC, B, R0, R2, R3, R4R6, R7, DPl, DPH, банк пам'яті 1 2. Виводити в порт 1 значення NN, 1, … 180, із інтервалом 15мс 3. Пін 2 порт 0 із періодом 28 мс
7	1. ACC, B, R0, R1, R2, R3, R4, R5, DPl, DPH, банк пам'яті 2 2. Виводити в порт 2 значення NN, 1, … 184, із інтервалом 18мс 3. Пін 2 порт 1 із періодом 22 мс
8	1. ACC, B, R0, R1, R2, R4, R5, R7, DPl, DPH, банк пам'яті 3 2. Виводити в порт 3 значення NN, 1, … 169, із інтервалом 19мс 3. Пін 6 порт 3 із періодом 29 мс
9	1. ACC, B, R0, R1, R2, R4, R5, R6, DPl, DPH, банк пам'яті 3 2. Виводити в порт 0 значення NN, 1, … 134, із інтервалом 23мс 3. Пін 5 порт 0 із періодом 21 мс
10	1. ACC, R1, R2, R3, R4, R5, R6, R7, DPl, DPH, банк пам'яті 2 2. Виводити в порт 1 значення NN, 1, … 115, із інтервалом 37мс 3. Пін 6 порт 1 із періодом 31 мс
11	1. ACC, R0, R1, R2, R4, R5, R6, R7, DPl, DPH, банк пам'яті 1 2. Виводити в порт 2 значення NN, 1, … 185, із інтервалом 45мс 3. Пін 7 порт 2 із періодом 18 мс
12	1. ACC, B, R0, R2, R3, R4, R5, R7, DPl, DPH, банк пам'яті 0 2. Виводити в порт 3 значення NN, 1, … 165, із інтервалом 60мс 3. Пін 0 порт 3 із періодом 35 мс

Вариант	Завдання до роботи
13	1. ACC, B, R0, R1, R4, R5, R6, R7, DPl, DPH, банк пам’яті 3 2. Виводити в порт 0 значення NN, 1, … 142, із інтервалом 47мс 3. Пін 1 порт 0 із періодом 47 мс
14	1. ACC, B, R1, R2, R3, R4, R5, R7, DPl, DPH, банк пам’яті 2 2. Виводити в порт 1 значення NN, 1, … 157, із інтервалом 35мс 3. Пін 2 порт 1 із періодом 38 мс
15	1. ACC, R0, R1, R2, R4, R5, R6, R7, DPl, DPH, банк пам’яті 0 2. Виводити в порт 2 значення NN, 1, … 186, із інтервалом 38мс 3. Пін 3 порт 2 із періодом 28 мс
16	1. ACC, B, R0, R1, R2, R4, R5, R6, DPl, DPH, банк пам’яті 1 2. Виводити в порт 3 значення NN, 1, … 153, із інтервалом 48мс 3. Пін 3 порт 3 із періодом 36 мс

*NN – номер варіанта за списком

6. ПРАКТИЧНА РОБОТА №6 «ПОСЛІДОВНИЙ ІНТЕРФЕЙС МІКРОКОНТРОЛЕРА 8051»

6.1 Регістр керування/статусу приймача SCON

Через універсальний асинхронний приймач UART (Universal Asynchronous Receiver-Transmitter) здійснюються прийом і передача інформації, представленої послідовним кодом (молодшими бітами вперед), в повному дуплексному режимі обміну. До складу приймача, званого часто послідовним портом входять приймаючий і передавальний зсувний регістри, а також спеціальний буферний регістр (SBUF) приймача. Крім того, роботою послідовного порту управляють два службових регістри:

• Регістр керування / статусу приймача SCON;

• Біт SMOD регістра керування потужністю PCON. Запис байта в буфер призводить до автоматичного перепису байта в зсувний регістр передавача і ініціює початок передачі байта. Наявність буферного регістра приймача дозволяє поєднувати операцію читання раніше прийнятого байта з прийомом чергового. Але якщо до моменту закінчення прийому байта попередній не був лічений з SBUF, то він буде втрачений.

Послідовний порт 8051 може працювати в чотирьох різних режимах.

• Режим 0. Інформація передається і приймається через вивід входу приймача (RXi TXi). Приймаються або передаються 8 біт даних. Через вивід виходу передавача (TXD) видаються імпульси зсуву, які супроводжують кожен біт.

Частота передачі біта інформації дорівнює 1/12 частоти кварцевого резонатора

• Режим 1. У цьому режимі передаються через вивід TXD або приймаються через RXD 10 біт інформації: старт-біт (0), 8 біт даних і стоп-біт (1) при прийомі інформації в біт RB8 регістра управління / статусу приймача SCON заносяться стоп-біт Швидкість прийому / передачі - величина змінна і задається таймером.

• Режим 2. У цьому режимі через вивід TXD передаються або через RXD приймаються 11 біт інформації: старт-біт, 8 біт даних, програмований дев'ятий біт і стоп-біт. При передачі дев'ятий біт даних

може приймати значення 0 або 1 або, наприклад, для підвищення достовірності передачі шляхом контролю по парності в нього може бути вміщено значення ознаки паритету зі слова стану програми (PSW.0). При прийомі дев'ятий біт даних міститься в біт RB8 SCON, а 40 стоп-біт, на відміну від режиму 1, втрачається. Частота прийому / передачі вибирається програмою і може дорівнювати або 1/32, або 1/64 частоти резонатора в залежності від керуючого біта SMOD.

• Режим 3. збігається з режимом 2 у всіх деталях, за винятком частоти прийому / передачі, яка є величиною змінною і задається таймером.

У всіх випадках передача ініціюється інструкцією, в якій дані переміщаються в SBUF. Прийом ініціюється при виявленні перепаду з 1 в 0 на вході приймача.

При цьому в режимі 0 цей перехід повинен супроводжуватися виконанням умов R1 = 0 і REN = 1 (див. Табл. 5.1), а для інших режимів - REN = 1.

Регістр керування/статусу приймача SCON

Управління режимом роботи приймача здійснюється через спеціальний регістр з символічним ім'ям SCON. Цей регістр містить не тільки керуючі біти, що визначають режим роботи послідовного порту, але і дев'ятий біт прийнятих або переданих даних (RB8 і TB8) і біти переривання приймача (R1 і T1).

Функціональне призначення біту регістра керування / статусу UART наводиться в таблиці 5.2.

Прикладна програма шляхом завантаження в старші біти регістра SCON двухбітного коду визначає режим роботи приймача. У всіх чотирьох режимах роботи передача ініціюється будь-якою командою, в якій буферний регістр SBUF зазначений як одержувач байта. Як уже зазначалося, прийом в режимі 0 здійснюється за умови, що R1 = 0 і REN = 1, в інших режимах - за умови, що REN = 1.

У біті TB8 програмно встановлюється значення дев'ятого біта даних, який буде переданий 8 режимі 2 або 3. У біті RB8 в цих режимах фіксується дев'ятий прийнятий біт даних. У режимі 1 у біт RB8 заноситься стоп-біт. У режимі 0 біт RB8 не використовується.

Прапор переривання передавача TI установлюється апаратно наприкінці періоду передачі стоп-біта у всіх режимах. Відповідна підпрограма обслуговування переривання повинна скидати біт TL.

Прапор переривання приймача RI встановлюється апаратно в кінці періоду прийому восьмого біта даних в режимі 0 і в середині періоду прийому стоп-біта в режимах 1, 2 і 3. Підпрограма обслуговування переривання повинна скидати біт RI.

Таблиця 6.1 – Функціональне призначення біт регістру керуваня/статусу прийомопередавача SCON

<table>
<tr><th>Символ</th><th>Позиція</th><th colspan="3">Ім’я та призначення</th></tr>
<tr><td rowspan="6">SM0
SM1</td><td rowspan="6">SCON.7
SCON.6</td><td colspan="3">Біти керування режимом роботи прийомопередавача.
Встановлюються та скидаються апаратно</td></tr>
<tr><td>SM0</td><td>SM1</td><td>Режим роботи прийомопередавача</td></tr>
<tr><td>0</td><td>0</td><td>Регістр зсуву вводу/виводу</td></tr>
<tr><td>0</td><td>1</td><td>8-бітний прийомопередавач, змінювана швидкість передачі</td></tr>
<tr><td>1</td><td>0</td><td>9-бітний прийомопередавач, фіксована швидкість передачі</td></tr>
<tr><td>1</td><td>1</td><td>9-бітний прийомопередавач, змінювана швидкість передачі</td></tr>
<tr><td>SM2</td><td>SCON.5</td><td colspan="3">Біт керування режимом прийомопередавача. Встановлюється програмно для заборони прийому повідомлення, в якому дев’ятий біт має значення 0</td></tr>
<tr><td>REN</td><td>SCON.4</td><td colspan="3">Біт дозволу прийому. Встановлюється/скидається програмно для дозволу/заборони прийому послідовних данних</td></tr>
<tr><td>TB8</td><td>SCON.3</td><td colspan="3">Передача біту 8. Встановлюється/скидається програмно для задання дев’ятого біту, що передається, у режимі 9 бітового передавача</td></tr>
<tr><td>RB8</td><td>SCON.2</td><td colspan="3">Приймання біту 8. Встановлюється/скидається програмно для фіксації дев’ятого біту, що приймається, у режимі 9 бітового приймача</td></tr>
<tr><td>TI</td><td>SCON.1</td><td colspan="3">Прапор переривання передавача. Встановлюється апаратно при закінченні передачі байту. Скидається програмно після обслуговування переривання.</td></tr>
<tr><td>RI</td><td>SCON.0</td><td colspan="3">Прапор переривання приймача. Встановлюється апаратно при закінченні прийому байту. Скидається програмно після обслуговування переривання.</td></tr>
</table>

6.2 Робота UART в мультиконтролерних системах

У системах децентралізованого управління, які використовуються для управління і регулювання в топологічно розподілених об'єктах, виникає задача обміну інформацією між безліччю мікроконтролерів, об'єднаних в локальну обчислювально-керуючу мережу. Як правило, локальні мережі на основі Intel 8051 мають магістральну архітектуру з роздільним моноканалом (коаксіальний кабель, вита пара, оптичне волокно), по якому здійснюється обмін інформацією між контролерами.

У регістрі спеціальних функцій SCON мікроконтролера є керуючий біт SM2, який в режимах 2 і 3 UART дозволяє простими засобами реалізувати міжконтролерний обмін інформацією в локальних керуючих мережах.

Механізм обміну побудований на тому, що в режимах 2 і 3 програмований дев'ятий біт даних при прийомі фіксується в біті RB8. UART може бути запрограмований таким чином, що при отриманні стоп-біта переривання від приймача буде можливо тільки за умови RB8 = 1. Це виконується установкою керуючого біта SM2 в регістрі SCON.

Пояснимо процес міжконтролерного обміну інформацією на прикладі. Нехай ведучому МК потрібно передати блок даних деякому (або декільком) відомому МК.

Ведучий МК всім відомим передає широкомовне повідомлення з байтом-ідентифікатором абонента, яке відрізняється від байтів даних тільки тим, що в його дев'ятому біті міститься 1. Програма реалізації протоколу мережевого обміну інформацією повинна бути побудована таким чином, щоб при отриманні байта-ідентифікатора (RB8 = 1) у всіх відомих МК відбулися переривання прикладних програм і виклик підпрограми порівняння байта-ідентифікатора з кодом власної мережевої адреси. Адресний МК скидає свій SM2 і готується до прийому блоку даних. Решта ведені мікроконтролери залишають незмінними свої SM2 = 1 і передають управління основній програмі. При SM2 = 1 інформаційні байти в мережі, передаються по моноканалу і надходять в UART ведених МК, переривання не викликають, тобто ігноруються.

У режимі 1 UART автономного мікроконтролера біт SM2 використовується для контролю істинності стоп-біта (при SM2 = 1 переривання не відбудеться до тих пір, поки не буде отримано 43 справжнє (одиничне) значення стоп-біта). У режимі 0 біт SM2 не використовується і повинен бути скинутий.

Швидкість прийому/передачі інформації через послідовний порт швидкість прийому/передачі, тобто частота роботи приймача в різних режимах, визначається різними способами [1].

У режимі 0 частота передачі залежить тільки від резонансної частоти кварцевого резонатора. За один машинний цикл послідовний порт передає один біт інформації.

У режимах 1, 2 і 3 швидкість прийому / передачі залежить від значення керуючого біта SMOD в регістрі спеціальних функцій PCON (таблиці 6.2).

Таблиця 6.2 – Регістр керування потужністю PCON

Символ	Позиція	Ім'я та призначення
SMOD	PCON.7	Подвоєна швидкість передачі. Якщо біт встановлений в 1, то швидкість передачі вдвічі більша, ніж при SMOD=0. При скиданні SMOD=0.
	PCON.6	Не використовується
	PCON.5	Не використовується
	PCON.4	Не використовується
GF1 GF0	PCON.3 PCON.2	Прапори, що специфікуються користувачем (прапори загального призначення)
PD	PCON.1	Біт зниженої потужності. При установці біту в 1 мікро-ЕВМ переходить у режим зниженої потужності.
IDL	PCON.0	Біт холостого ходу. Якщо біт встановлений в 1, то мікро-ЕВМ переходить у режим холостого ходу.

Примітка. При одночасному записі 1 в PD і IDL біт PD має перевага. Скидання вмісту PCON виконується шляхом завантаження в нього коду 0XXX0000.

У режимі 2 частота передачі визначається виразом

$$f_2=(2^{SMOD}/64)\cdot f_{рез}$$

Іншими словами, при SMOD = 0 частота передачі дорівнює 1/64 частоти рез f, а пої SMOD =1-1/32 частоти $f_{рез}$.

У режимах 1 і 3 у формуванні частоти передачі, крім керуючого біта SMOD, бере участь таймер 1. При цьому частота передачі f залежить від частоти переповнення OVLT 1 f визначається наступним чином:

$$f_{1,3}=(2^{SMOD}/32)\cdot f_{CVLT1}$$

Переривання від таймера 1 в цьому випадку має бути заблоковано. Сам же таймер може працювати як в режимі таймера, так і в режимі лічильника. Номер режиму (0, 1, 2) ролі не грає. Найбільш типове використання його в режимі таймера з автоперезагрузка (старша тетрада TMOD = 0010B). При цьому частота передачі визначається виразом:

$$f_{1,3}=(2^{SMOD}/32)\cdot(f_{рез}/12)\cdot(256-(TH1))$$

У таблиці 6.3 наводиться опис способів настройки таймера 1 для отримання типових частот передачі даних через UART.

Таблиця 6.3 – Налаштування таймера 1 для керування частотою роботи прийомопередавача

Частота прийому/передачі (BAUD RATE)	Частота резонатора МГц	Таймер/лічильник 1			
		SMOD	C/T	Режим (MODE)	Перезавантажуване число
Режим 0, макс: 1 МГц	12	X	X	X	X
Режим 2, макс: 375 кГц	12	1	X	X	X
Режим 1, 3: 62.2 кГц	12	1	0	2	0FFH
Режим 1, 3: 19.2 кГц	11.059	1	0	2	0FDH
Режим 1, 3: 9.6 кГц	11.059	0	0	2	0FDH
Режим 1, 3: 4.8 кГц	11.059	0	0	2	0FAH
Режим 1, 3: 2.4 кГц	11.059	0	0	2	0F4H
Режим 1, 3: 1.2 кГц	11.059	0	0	2	0F4H
Режим 1, 3: 137.5 кГц	11.059	0	0	2	1DH
Режим 1, 3: 110 кГц	6	0	0	2	72H
Режим 1, 3: 110 кГц	12	0	0	1	0FEEBH

Відзначимо, що швидкості прийому і передачі можуть відрізнятися. Гранично низьких частот прийомопередачі можна досягти при використанні таймера в режимі 1(16-бітний таймер) і дозволі переривань від таймера (старший напівбайт TMOD = 0001B). Перезавантаження 16-бітного таймера повинно здійснюватися програмним шляхом.

6.3. Робота з послідовним портом

Послідовний порт мікроконтролера може використовуватися у вигляді регістра зсуву для розширення введення / виводу або в якості UART з фіксованою або змінною швидкістю послідовного обміну і можливістю дуплексного включення [1]. Тобто через порт можна передавати і приймати дані одночасно. Порт може приймати черговий байт навіть в тому випадку, якщо вже прийнятий до цього байт не прочитаний з регістру приймача. Однак якщо до закінчення прийому байт, що знаходиться в регістрі приймача, НЕ буде прочитаний, прийнятий байт втрачається. Програмний доступ до регістрів

приймача і передавача здійснюється зверненням до регістру спеціальних функцій SBUF.

Нижче наведено приклад фрагмента програми (UART), приймаючої з послідовного порту байт і відправляє його назад в послідовний порт, налаштований на 8- бітний режим зі швидкістю передачі 2400 бод при тактовій частоті мікроконтролера 12 МГц.

```
UART:   MOV SCON, #0x52 ;встановлення режиму 8-бітного UART
        MOV TMOD, #0x20 ;вст. режиму автозавантаження таймера 1
        MOV TH1,  #0xF3 ;автозавантажуване значення для отримання
                        ;швидкості 2400 бод на частоті МК 12 МГц
        SETB    TCON.6  ;пуск таймера
;прийом символу порта
CIN:    JNB RI, CIN     ;очікування завершення прийому
        MOV A, SBUF     ;читання символу
        CLR RI          ;очистка прапору прийому
;видача символу в послідовний порт
COUT:   JNB TI,COUT     ;очікування завершення передачі
        CLR TI          ;очитска прапору передачі
        MOV SBUF, A     ;видача символу
                        ;
        SJMP CIN        ;
```

6.4 Завдання для практичної роботи

Відповісти на наступні питання:

1. Охарактеризуйте режими роботи послідовного порту.
2. Для чого призначений регістр SCON?
3. Для чого використовується дев'ятий біт?
4. Як змінити швидкість передачі даних через послідовний порт?
5. Визначте призначення UART.
6. Назвіть основні регістри, що використовуються портом UART.
7. Дайте опис бітів регістра управління UART0 (SCON0).
8. Призначення генератора швидкості передачі даних в UART
9. Назвіть основні параметри, що визначають швидкість передачі даних?
10. Назвіть режими роботи UART і їх особливості.

Завдання:

Написати підпрограму для ініціалізації послідовного порту, прийому та передачі данних та передати в нього дані свого варіанту.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Мікропроцесорна техніка: Комп’ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / К. П. Вонсевич, М. О. Безуглий ; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 3,53 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 94 с
2. Мікропроцесорна техніка. Однокристальні мікроконтролери: навч. посібник / С.Р.Михайлов. – К.:, 2014. – 123 с.
3. Мікропроцесорна техніка : підручник / В.В. Ткачов, С.М. Проценко, М 59 М.В. Козар, В.І Шевченко; М-во освіти і науки України, НТУ «Дніпровська політехніка». – 2-ге вид., допов. і переробл. – Дніпро : НТУ «ДП». – 2022. – 230 с.
4. Костинюк Л.Д. Мікропроцесорні засоби та системи / Л.Д. Костинюк, Я.С. Парганчук. – Львів.: Львівська політехніка, 2001. – 200 с
5. Електроніка і мікропроцесорна техніка / Сенько В.І., Лисенко В.П., Юрченко О.М., Лукін В.Є., Руденський А.А. — К. : «Агроосвіта», 2015. — 676 с.

Додаток А
Приклад оформлення титульної сторінки

Міністерство освіти та науки України

Національний університет «Запорізька політехніка»

Кафедра ІБтаН

ЗВІТ
З ПРАКТИЧНОЇ РОБОТИ №__
_____
(тема роботи)
з дисципліни „МПТ“

Виконав:
студент гр. __ _____
(Ініціали, Прізвище)

Прийняв:
(посада) _____
(Ініціали, Прізвище)

20__ р.

Додаток Б
Система команд мікроконтролера сімейства 8051

Б.1 Загальна характеристика

Мікро-ЕОМ розглянутого сімейства є типовими мікропроцесорними пристроями з архітектурою CISC - зі стандартним набором команд. Тому їх система команд досить обширна і включає в себе 111 основних команд. Їх довжина - один, два або три байта, причому більшість з них (94%) - одно- або двухбайтне. Всі команди виконуються за один або два машинних циклу (відповідно 1 або 2 мкс при тактовій частоті 12 МГц), виняток - команди множення і ділення, які виконуються за чотири машинних циклу (4 мкс).

Мікро-ЕОМ сімейства 8051 використовують пряму, безпосередню, непряму і неявну, адресацію даних.

Як операнди команд мікро-ЕОМ сімейства 8051 можуть використовувати окремі біти, чотирибітні цифри, байти і двобайтні слова.

Всі ці риси звичайні для набору команд будь-якого CISC-процесора і в порівнянні з RISC набором команд забезпечує більшу компактність програмного коду і збільшення швидкодії при виконанні складних операцій.

У той же час, набір команд сімейства 8051 має кілька особливостей, пов'язаних з типовими функціями виконуваними мікроконтролерами - управлінням, для якого типовим є оперування з одноразрядними двійковими сигналами, велике число операцій введення/виводу і розгалужень програми. Найбільш істотна особливість системи команд розглянутих мікро ЕОМ це можливість адресації окремих біт в резидентній пам'яті даних.

Крім того, як зазначалося, деякі регістри блоку регістрів спеціальних функцій також допускають адресацію окремих біт. Карти адрес окремих біт в резидентній пам'яті даних і в блоці регістрів спеціальних функцій представлені в табл. 1 і 2 відповідно [1, 3, 5].

Типи команд

Всього мікро-ЕОМ виконують 13 типів команд, вони наведені в таблиці. Як випливає з неї, перший байт команди завжди містить код операції (КОП), а другий і третій (якщо вони присутні в команді)

- адреси операндів або їх безпосередні значення. У Табл. 5.1 показані 13 типів команд MCS51.

Таблиця **Б.**1 – Типи команд MCS51

Тип команди	Перший байт D7...D0	Другий байт D7...D0	Третій байт D7...D0
Тип 1	КОП		
Тип 2	КОП	#d	
Тип 3	КОП	ad	
Тип 4	КОП	bit	
Тип 5	КОП	rel	
Тип 6	КОП	a7...a0	
Тип 7	КОП	ad	#d
Тип 8	КОП	ad	rel
Тип 9	КОП	ads	add
Тип 10	КОП	#d	rel
Тип 11	КОП	bit	rel
Тип 12	КОП	ad16h	ad16l
Тип 13	КОП	#d16h	#d16l

Позначення, що використовуються при описі команд:

- Rn (n=0,1,...,7) – регістр загального призначення в обраному банку регістрів;
- @Ri (i=0, 1) – регістр загального призначення в обраному банку регістрів, який використовується в якості регістру непрямої адреси;
- ad – адреса байта, що прямо адресується;
- ads – адреса байта-джерела, що прямо адресується ;
- add – адреса байта-одержувача, що прямо адресується ;
- ad11 – 11-розрядна абсолютна адреса переходу;
- ad16 – 16-розрядна абсолютна адреса переходу;
- Rel – відносна адреса переходу;
- #d – безпосередній операнд;
- #d16 – безпосередній операнд (2 байта);
- bit – адреса біта, що прямо адресується;
- /Bit – інверсія біта, що прямо адресується;
- A – акумулятор;
- PC – лічильник команд;

❖ DPTR – регістр покажчик даних;
❖ () – вміст комірки пам'яті або регістра.

Типи операндів

Склад операндів включає в себе операнди чотирьох типів: біти, 4-бітові цифри, байти і 16-бітові слова.

Мікроконтролер має 128 програмно-керованих прапорів користувача. Є також можливість адресації окремих біт блоку регістрів спеціальних функцій і портів. Для адресації бітів використовується пряма 8-бітову адреса (bit). Непряма адресація біт неможлива. Карти адрес окремих бітів представлені в таблиці 1 і 2. Чотирибітні операнди використовуються тільки при операціях обміну (команди SWAP і XCHD).

Восьмибітним операндом може бути осередок пам'яті програм (PM) або даних (резидентної (RDM) або зовнішньої (EDM)), константа (безпосередній операнд), регістри спеціальних функцій (SFR), а також порти введення / виводу. Порти і регістри спеціальних функцій адресуються тільки прямим способом. Байти пам'яті можуть адресуватися також і непрямим чином через адресні регістри (R0, R1, DPTR і PC).

Двобайтні операнди - це константи і прямі адреси, для подання яких використовуються другий і третій байти команди.

Способи адресації даних

У мікроконтролері застосовуються такі методи адресації даних: пряма, безпосередня, непряма і неявна.

При непрямому способі адресації резидентної пам'яті даних використовуються всі вісім бітів адресних регістрів R0 і R1.

Прапори результату

Слово стану програми PSW включає себе чотири прапори:

❖ C - перенос,
❖ AC - допоміжне перенесення (полуперенос),
❖ 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. Всі спроби змінити прапор

P, привласнюючи йому нове значення, не приносять користі, якщо вміст акумулятора при цьому залишиться незмінним.

Прапор AC встановлюється, якщо при виконанні операції додавання і віднімання між тетрадами байта (напівбайтами) виник перенос або позика.

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

Прапор OV встановлюється, якщо результат операції додавання і 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 ділення на нуль встановлюється. При множенні прапор OV встановлюється, якщо результат більше 255.

У таблиці 7.2 перераховуються команди, при виконанні яких модифікуються прапори результату. У таблиці відсутній прапор паритету, так як його значення змінюється усіма командами, які змінюють вміст акумулятора. Крім команд, наведених у таблиці, прапори модифікуються командами, в яких місцем призначення результату визначені PSW або його окремі біти, а також командами операцій над бітами.

Таблиця Б.2 – Команди, що модифікують прапори

Команди	Прапори	Команди	Прапори
ADD	C, OV, AC	CLR C	C = 0
ADC	C, OV, AC	CPL C	C = NOT(C)
SUBB	C, OV, AC	ANL C, b	C
MUL	C = 0, OV	ANL C, /b	C
DIV	C = 0, OV	ONL C, b	C
DA	C	ONL C, /b	C
RRC	C	MOV C, b	C
RLC	C	CJNE	C
SETB C	C = 1		

Символічна адресація

При використанні асемблера для отримання об'єктних кодів програм допускається застосування в програмах символічних імен

регістрів спеціальних функцій, портів і їх окремих бітів (табл.2). Для адресації окремих бітів і портів (така можливість є не у всіх регістрів спеціальних функцій) можна використовувати символічне ім'я біта наступної структури: <ім'я регістра або порту>. <Номер біта>.

Наприклад, символічне ім'я п'ятого біта акумулятора буде наступним: ACC.5. символічні імена є зарезервованими словами, і їх не треба визначати за допомогою директив асемблера.

Б.2 Групи команд

Система команд сімейства MCS-51 містить 111 базових команд, які за функціональною ознакою можна поділити на п'ять груп:

- ❖ пересилання даних;
- ❖ арифметичних операцій;
- ❖ логічних операцій;
- ❖ операцій над бітами;
- ❖ передачі управління.

Формат команд - одно-, дво- і трьохбайтовий, причому більшість команд (94) мають формат один або два байти. Перший байт будь-яких типу і формату завжди містить код операції, другий і третій байти містять або адреси операндів, або безпосередні операнди.

Склад операндів включає в себе операнди чотирьох типів: біти, тетради (4 біт), байти і 16-бітові слова. Час виконання команд складає 1, 2 або 4 машинних циклу. При тактовій частоті 12 мГц тривалість машинного циклу складає 1 мкс, при цьому команди виконуються за 1 мкс, 45 команд - за 2 мкс і 2 команди (множення і ділення) – за 4 мкс. Набір команд MCS-51 підтримує наступні режими адресації.

Пряма адресація (Direct Addressing). Операнд визначається 8-бітовою адресою в інструкції. Ця адресація використовується тільки для внутрішньої пам'яті даних і регістрів SFR.

Непряма адресація (Indirect Addressing). В цьому випадку інструкція адресує регістр, що містить адресу операнда. Даний вид адресації може застосовуватися при зверненні як до внутрішнього, так і зовнішнього ОЗУ. Для вказівки 8-бітових адрес можуть використовуватися регістри R0 і R1 вибраного реєстрового банку або покажчик стека SP. Для 16-бітної адресації використовується тільки регістр "покажчик даних" (DPTR - Data Pointer).

Реєстрова адресація (Register Instructions). Дана адресація застосовується для доступу до регістрів R0 ... R7 обраного банку.

Команди з реєстрової адресацією містять в байті коду операції трьохбітовое поле, що визначає номер регістра. Вибір одного з чотирьох реєстрових банків здійснюється програмуванням бітів селектора банку (RS1, RS0) в PSW.

Безпосередня адресація (Immediate constants). Операнд міститься безпосередньо в полі команди услід за кодом операції і може займати один або два байти (data8, data16). Індексна адресація (Indexed Addressing). Індексна адресація використовується при зверненні до пам'яті програм і тільки при читанні. В цьому режимі здійснюється перегляд таблиць в пам'яті програм. 16-бітовий регістр (DPTR або PC) вказує базову адресу необхідної таблиці, а акумулятор вказує на точку входу в неї. Адреса елемента таблиці знаходиться складанням бази з індексом (вмістом акумулятора).

Інший тип індексної адресації застосовується в командах "переходу на вибір" (Case Jump). При цьому адреса переходу обчислюється як сума покажчика бази і акумулятора.

Неявна адресація (Register-Specific Instructions). Деякі інструкції використовують індивідуальні регістри (наприклад, операції з акумулятором, DPTR), при цьому дані регістри не мають адреси, що вказує на них; це закладено в код операції.

Б.3 Команди передачі даних

Ця група представлена 28 командами, їх короткий опис наведено в таблиці Б.3, де також вказані тип команди (Т) відповідно до таблиці, її довжина в байтах (Б) і час виконання в машинних циклах (Ц). Більшу частину команд даної групи складають команди передачі і обміну байтів. Команди пересилання бітів представлені в групі команд бітових операцій. Всі команди цієї групи не модифікують прапори результату, за винятком команд завантаження PSW і акумулятора (прапор паритету).

За командою MOV виконується пересилання даних з другого операнда в перший.

Ця команда не має доступу ні до зовнішньої пам'яті даних, ні до пам'яті програм. Для цих цілей призначені команди M0VX і MOVC відповідно. Перша з них забезпечує читання / запис байт із зовнішньої пам'яті даних, друга - читання байт з пам'яті програм.

За командою XCH виконується обмін байтами між акумулятором і осередком РПД, а по команді XCHD - обмін молодшими тетрадами (бітами 0 - 3).

Команди PUSH і POP призначені відповідно для запису даних в стек і їх читання з стека.

Розмір стека обмежений лише розміром резидентної пам'яті даних. В процесі ініціалізації мікро-ЕОМ після сигналу скидання або при включенні напруги живлення в SP заноситься код 07H. Це означає, що перший елемент стека буде розташовуватися в комірці пам'яті з адресою 08H.

Група команд пересилань мікроконтролера має таку особливість - в ній немає спеціальних команд для роботи зі спеціальними регістрами: PSW, таймером, портами введення-виведення. Доступ до них, як і до інших регістрів спеціальних функцій, здійснюється завданням відповідної прямої адреси, тобто це команди звичайних пересилань, в яких замість адреси можна ставити назву відповідного регістра.

Наприклад, читання PSW в акумулятор може бути виконано командою MOV A, PSW яка перетворюється Ассемблером до виду MOV A, 0D0h (E5 D0), де E5 - код операції, а D0 - операнд (адреса PSW).

Таблиця Б.3 – Група команд передачі даних

Назва команди	Мнемокод	T	Операція
Пересилання в акумулятор з регістра (n = 0_7)	MOV A,Rn	1	(A) <= (Rn)
Пересилання в акумулятор прямоадресуемого байта	MOV A,ad	3	(A) <= (ad)
Пересилання в акумулятор байта з РПД (i = 0,1)	MOV A,@Ri	1	(A) <= (Ri))
Завантаження в акумулятор константи	MOV A,#d	2	(A) <= #d
Пересилання в регістр з акумулятора	MOV Rn,A	1	(Rn) <= (A)
Пересилання в регістр прямоадресуемого байта	MOV Rn,ad	3	(Rn) <= (ad)
Пересилання в регістр константи	MOV Rn,#d	2	(Rn) <= #d

Назва команди	Мнемокод	Т	Операція
Пересилання за прямою адресою акумулятора	MOV ad,A	3	(ad) <= (A)
Пересилання за прямою адресою регістра	MOV ad,Rn	3	(ad) <= (Rn)
пересилання прямоадресованого байта за прямою адресою	MOV add,ads	9	(add) <= (ads)
Пересилання байта з РПД за прямою адресою	MOV ad,@Ri	3	(ad) <= ((Ri))
Пересилання за прямою адресою константи	MOV ad,#d	7	(ad) <= #d
Пересилання в РПД з акумулятора	MOV @Ri,A	1	((Ri)) <= (A)
Пересилання в РПД прямоадресного байта	MOV @Ri,ad	3	((Ri)) <= (ad)
Пересилання в РПД константи	MOV @Ri,#d	2	((Ri)) <= #d
Завантаження покажчика даних	MOV DPTR,#d16	13	(DPTR) <= #d16
Пересилання в акумулятор байта з ПП	MOVC A,@A+DPTR	1	(A) <= ((A) + (DPTR))
Пересилання в акумулятор байта з ПП	MOVC A,@A+PC	1	(PC) <= (PC) + 1; (A) <= ((A) + (PC))
Пересилання в акумулятор байта з ВПД	MOVX A,@Ri	1	(A) <= ((Ri))
Пересилання в акумулятор байта з розширеною ЗПД	MOVX A,@DPTR	1	(A) <= ((DPTR))
Пересилання в ЗПД з акумулятора	MOVX @Ri, A	1	((Ri)) <= (A)

Назва команди	Мнемокод	Т	Операція
Пересилання в розширену ЗПД з акумулятора	MOVX @DPTR,A	1	((DPTR)) <= (A)
Завантаження в стек	PUSH ad	3	(SP) <= (SP) + 1 ((SP)) <= (ad)
Витяг з стека	POP ad	3	(ad) <= ((SP)) (SP) <= (SP) - 1
Обмін акумулятора з регістром	XCH A,Rn	1	(A) <=> (Rn)
Обмін акумулятора з прямоадресним байтом	XCH A,ad	3	(A) <=> (ad)
Обмін акумулятора з байтом з РПД	XCH A,@Ri	1	(A) <=> ((Ri))
Обмін молодшої тетради акумулятора з молодшою тетрадой РПД	XCHD A,@Ri	1	(A0_3) <=> ((Ri)0_3)

Структура інформаційних зв'язків

Залежно від способу адресації і місця розташування операнда можна виділити дев'ять типів операндів, між якими можливий інформаційний обмін. Граф можливих операцій передачі даних показаний на рис. Б.1. Передачі даних можуть виконуватися без участі акумулятора [1].

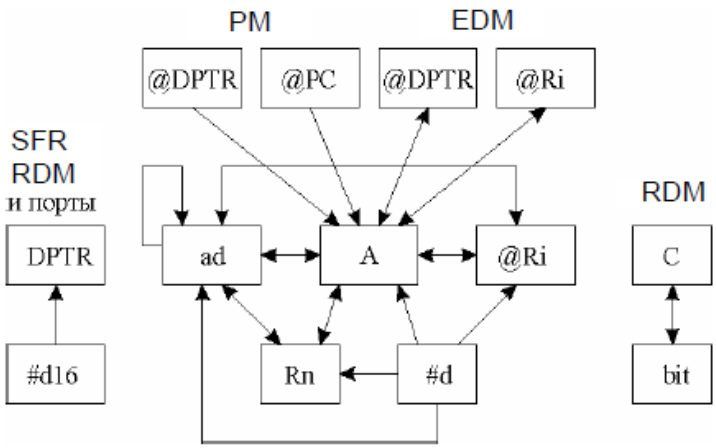


Рисунок Б.1 – Граф шляхів передачі данних

Звернення до акумулятора

Звернення до акумулятора може бути виконано з використанням неявної і прямої адресації. Залежно від способу адресації акумулятора застосовується одне із символічних імен: A або ACC (пряма адресу). При прямій адресації звернення до акумулятора проводиться як до одного з регістрів спеціальних функцій, і його адреса вказується в другому байті команди. Використання неявної адресації акумулятора краще, але не завжди можливо, наприклад, при зверненні до окремих бітів акумулятора.

Звернення до зовнішньої пам'яті даних

При використанні команд MOVX @Ri забезпечується доступ до 256 байтів зовнішньої пам'яті даних. Існує також режим звернення до розширеної зовнішньої пам'яті даних, коли для доступу використовується 16-бітова адреса, що зберігається в регістрі-покажчику даних DPTR. Команди MOVX @DPTR забезпечують доступ до 65536 байтів зовнішньої пам'яті даних. PM EDM SFR RDM RDM Рис. Б.1 Граф шляхів передачі даних

Б.4 Арифметичні операції

Дану групу утворюють 24 команди (Таблиця Б.4), що виконують операції складання, десяткової корекції, інкремента / декремента байтів. Є команди віднімання, множення і ділення байтів.

Таблиця Б.4 – Група команд арифметичних операцій

Назва команди	Мнемокод	Т	Операція
додавання акумулятора регістром (n = 0_7)	ADD A,Rn	1	(A) <= (A) + (Rn)
Додавання акумулятора з прямоадресуємим байтом	ADD A,ad	3	(A) <= (A) + (ad)
Додавання акумулятора з байтом з РПД (i = 0,1)	ADD A,@Ri	1	(A) <= (A) +((Ri))
Додавання акумулятора з константою	ADD A,#d	2	(A) <= (A) + #d
Додавання акумулятора з регістром і перенесенням	ADDC A,Rn	1	(A) <= (A) + (Rn) + (C)
Додавання акумулятора з прямоадресуємим байтом і переносом	ADDC A,ad	3	(A) <= (A) + (ad) + (C)
Додавання акумулятора з байтом з РПД і перенесенням	ADDC A,@Ri	1	(A) <= (A) + ((Ri)) + (C)
Додавання акумулятора з константою і перенесенням	ADDC A,#d	2	(A) <= (A) + #d + (C)
десяткова корекція акумулятора	DA A	1	Если (A0_3) > 9 || ((AC)=1), то (A0_3) <= (A0_3) + 6,

Назва команди	Мнемокод	T	Операція
			затем если (A4_7) > 9 || ((C)=1), то (A4_7) <= (A4_7) + 6
Віднімання з акумулятора регістра і позички	SUBB A,Rn	1	(A) <= (A) - (C) - (Rn)
Віднімання з акумулятора прямоадресуємого байта і позички	SUBB A,ad	3	(A) <= (A) - (C) - (ad)
Віднімання з акумулятора байта з РПД і позички	SUBB A,@Ri	1	(A) <= (A) - (C) - ((Ri))
Віднімання з акумулятора константи і позички	SUBB A,#d	2	(A) <= (A) - (C) - #d
інкремент акумулятора	INC A	1	(A) <= (A) + 1
інкремент регістра	INC Rn	1	(Rn) <= (Rn) + 1
Інкремент прямоадресуємого байта	INC ad	3	(ad) <= (ad) + 1
Інкремент байта з РПД	INC @Ri	1	((Ri)) <= ((Ri)) + 1
Інкремент покажчика даних	INC DPTR	1	(DPTR) <= (DPTR) + 1
декремент акумулятора	DEC A	1	(A) <= (A) - 1
декремент регістра	DEC Rn	1	(Rn) <= (Rn) - 1
Декремент прямоадресуємого байта	DEC ad	3	(ad) <= (ad) - 1
Декремент байта з РПД	DEC @Ri	1	((Ri)) <= ((Ri)) - 1
Множення акумулятора на регістр B	MUL AB	1	(B)(A) <= (A)*(B)
Розподіл акумулятора на регістр B	DIV AB	1	(A)(B) <= (A)/(B)

Команди ADD і ADDC допускають складання акумулятора з великим числом операндів. Аналогічно командам ADDC існують чотири команди SUBB, що дозволяє досить просто проводити віднімання байтів і багатобайтових двійкових чисел.

У мікроконтролері реалізується розширений список команд інкремента / декремента байтів, команда інкремента 16-бітного регістра-покажчика даних.

За результатом виконання команд ADD, ADDC, SUBB, MUL і DIV встановлюються прапори PSW, структура яких наведена в табл. 4. Прапор C встановлюється при перенесенні з розряду D7, т. Е. У разі, якщо результат не поміщається в вісім розрядів; прапор AC встановлюється при перенесенні з розряду D3 в командах додавання і віднімання і служить для реалізації десяткової арифметики. Ця ознака використовується командою DAA. Прапор OV встановлюється при перенесенні з розряду D6, т. Е. У разі, якщо результат не поміщається в сім розрядів і восьмий не може бути інтерпретований як знаковий. Ця ознака служить для організації обробки чисел із знаком. Нарешті, прапор P встановлюється і скидається апаратно. Якщо число одиничних біт в акумуляторі непарній, то P = 1, в іншому випадку P = 0.

Б.5 Логічні операції

Дану групу утворюють 25 команд (Таблиця Б.5), що реалізують функціонально повну систему логічних операції над байтами. У мікроконтролері розширено число типів операндів, що беруть участь в операціях. Є можливість робити операцію "виключає АБО" з вмістом портів. Команда XRL може бути ефективно використана для інверсії окремих бітів портів.

Таблиця Б.5 – Група команд логічних операцій

Назва команди	Мнемокод	T	Операція
Логічне I акумулятора і регістра	ANL A,Rn	1	(A) <= (A) && (Rn)
Логічне I акумулятора і прямоадресуємого байта	ANL A,ad	3	(A) <= (A) && (ad)
Логічне I акумулятора і байта з РПД	ANL A,@Ri	1	(A) <= (A) && ((Ri))
Логічне I акумулятора і константи	ANL A,#d	2	(A) <= (A) &&

			#d
Логічне І прямоадресуємого байта і акумулятора	ANL ad,A	3	(ad) <= (ad) && (A)
Логічне І прямоадресуємого байта і константи	ANL ad,#d	7	(ad) <= (ad) && #d
Логічне АБО акумулятора та регістра	ORL A,Rn	1	(A) <= (A) || (Rn)
Логічне АБО акумулятора та прямоадресуємого байта	ORL A,ad	3	(A) <= (A) || (ad)
Логічне АБО акумулятора та байта з РПД	ORL A,@Ri	1	(A) <= (A) || ((Ri))
Логічне АБО акумулятора та константи	ORL A,#d	2	(A) <= (A) || #d
Логічне АБО прямоадресуємого байта і акумулятора	ORL ad,A	3	(ad) <= (ad) || (A)
Логічне АБО прямоадресуемого байта і константи	ORL ad,#d	7	(ad) <= (ad) || #d
Що виключає Або акумулятора і регістра	XRL A,Rn	1	(A) <= (A) ^ (Rn)
Що виключає Або акумулятора і прямоадресуємого байта	XRL A,ad	3	(A) <= (A) ^ (ad)
Що виключає Або акумулятора і байта з РПД	XRL A,@Ri	1	(A) <= (A) ^ ((Ri))
Що виключає Або акумулятора і константи	XRL A,#d	2	(A) <= (A) ^ #d
Що виключає Або прямоадресуємого байта і акумулятора	XRL A,#d	2	(A) <= (A) ^ #d

Що виключає Або прямоадресуємого байта і константи	XRL ad,#d	7	(ad) <= (ad) ^ #d
Скидання акумулятора	CLR A	1	(A) <= 0
інверсія акумулятора	CPL A	1	(A) <= ~(A)
Циклічний зсув акумулятора вліво	RL A	1	(An+1) <= (An) n=0_6; (A0) <= (A7)
Зрушення акумулятора вліво через перенос	RLC A	1	(An+1) <= (An) n=0_6; (A0) <= (C) ; (C) <= (A7)
Циклічний зсув акумулятора вправо	RR A	1	(An) <= (An+1) n=0_6; (A7) <= (A0)
Зрушення акумулятора вправо через перенос	RRC	1	(An) <= (An+1) n=0_6; (A7) <= (C) ; (C) <= (A0)
Обмін місцями тетрад в акумуляторі	SWAP A	1	(A0_3) <=> (A4_7)

Б.6 Команди передачі управління

До даної групи команд (Таблиця Б.6) відносяться команди, умовного і безумовного розгалуження, виклику підпрограм і повернення з них, а також команда порожньої операції NOP. У більшості команд використовується пряма адресація, тобто адреса переходу цілком (або його частина) міститься в самій команді передачі управління. Можна виділити три різновиди команд розгалуження по розрядності зазначених вище адрес переходу.

Таблиця Б.6 – Група команд передачі управління (управління послідовністю)

Назва команди	Мнемокод	T	Операція
довгий перехід в повному обсязі пам'яті	LJMP ad16	12	(PC) <= ad16

Короткий відносний перехід всередині сторінки в 256 байт	**SJMP rel**	**5**	**(PC) <= (PC) + 2 (PC) <= (PC) + rel**
Непрямий відносний перехід	**JMP @A+DPTR**	**1**	**(PC) <= (A) + (DPTR)**
Перехід, якщо акумулятор дорівнює нулю	**JZ rel**	**5**	**(PC) <= (PC) + 2, если (A)=0, то (PC) <= (PC) + rel**
Перехід, якщо акумулятор не дорівнює нулю	**JNZ rel**	**5**	**(PC) <= (PC) + 2, если (A)<>0, то (PC) <= (PC) + rel**
Перехід, якщо перенесення дорівнює одиниці	**JC rel**	**5**	**(PC) <= (PC) + 2, если (C)=1, то (PC) <= (PC) + rel**
Перехід, якщо перенесення дорівнює нулю	**JNC rel**	**5**	**(PC) <= (PC) + 2, если (C)=0, то (PC) <= (PC) + rel**
Перехід, якщо біт дорівнює одиниці	**JB bit,rel**	**11**	**(PC) <= (PC) + 3, если (bit)=1, то (PC) <= (PC) + rel**
Перехід, якщо біт дорівнює нулю	**JNB bit,rel**	**11**	**(PC) <= (PC) + 3, если (bit)=0, то (PC) <= (PC) + rel**
Перехід, якщо біт встановлений, з подальшим скидом біта	**JBC bit,rel**	**11**	**(PC) <= (PC) + 3, если (bit)=0, то (bit) <= 0 (PC) <= (PC) + rel**
Декремент регістра і перехід, якщо не нуль	**DJNZ Rn,rel**	**5**	**(PC) <= (PC) + 2 (Rn) <= (Rn) - 1, если (Rn)<>0,**

			то (PC) <= (PC) + rel
Декремент прямоадресуємого байта і перехід, якщо не нуль	DJNZ ad,rel	8	(PC) <= (PC) + 2 (ad) <= (ad) - 1, если (ad)<>0, то (PC) <= (PC) + rel
Порівняння акумулятора з прямоадресуємим байтом і перехід, якщо не дорівнює	CJNE A,ad,rel	8	(PC) <= (PC) + 3, если (A)<>(ad), то (PC) <= (PC) + rel, если (A)<(ad), то (C) <= 1, иначе (C) <= 0
Порівняння акумулятора з константою і перехід, якщо не дорівнює	CJNE A,#d,rel	10	(PC) <= (PC) + 3, если (A)<>#d, то (PC) <= (PC) + rel, если (A)<#d, то (C) <= 1, иначе (C) <= 0
Порівняння регістра з константою і перехід, якщо не дорівнює	CJNE Rn,#d,rel	10	(PC) <= (PC) + 3, если (Rn)<>#d, то (PC) <= (PC) + rel, если (Rn)<#d, то (C) <= 1, иначе (C) <= 0
Порівняння байта в РПД з константою і перехід, якщо не дорівнює	CJNE @Ri,#d,rel	10	(PC) <= (PC) + 3, если ((Ri))<>#d, то (PC) <= (PC) + rel, если ((Ri))<#d, то (C) <= 1, иначе(C) <= 0
Довгий виклик підпрограми	LCALL ad16	12	(PC) <= (PC) + 3; (SP) <= (SP) + 1 ((SP)) <= (PC0_7); (SP) <= (SP) + 1 ((SP)) <= (PC8_15); (PC) <= ad16

Абсолютний виклик підпрограми в межах сторінки в 2Кб	ACALL ad11	6	(PC) <= (PC) + 2; (SP) <= (SP) + 1 ((SP)) <= (PC0_7); (SP) <= (SP) + 1 ((SP)) <= (PC8_15); (PC0_10) <= ad11
Повернення з підпрограми	RET	1	(PC8_15) <= ((SP)); (SP) <= (SP) – 1 (PC0_7) <= ((SP)); (SP) <= (SP) - 1
Повернення з підпрограми обробки переривання	RETI	1	(PC8_15) <= ((SP)); (SP) <= (SP) – 1 (PC0_7) <= ((SP)); (SP) <= (SP) - 1
пуста команда	NOP	1	(PC) <= (PC) + 1

Довгий перехід

Перехід по всьому адресному простору пам'яті програм. У команді міститься повна 16-бітову адреса переходу (ad16). Трьохбайтні команди довгого переходу містять в Мнемокоді букву L (Long). Всього існує дві такі команди: LJMP − довгий перехід і LCALL − довгий виклик підпрограми. На практиці рідко виникає необхідність переходу в межах всього адресного простору, і частіше використовуються укорочені команди переходу, що займають менше місця в пам'яті.

Абсолютний перехід

Перехід в межах однієї сторінки пам'яті програм розміром 2048 байтів. Такі команди містять тільки 11 молодших бітів адреси переходу (ad11). Команди абсолютного переходу мають формат 2 байта. Початкова буква мнемокода − A (Absolute). При виконанні команди в розрахунковій адресі наступної по порядку команди ((PC) = (PC) + 2) 11 молодших бітів замінюються на ad11 з тіла команди абсолютного переходу.

Відносний перехід

Короткий відносний перехід дозволяє передати управління в межах від - 128 до +127 байт щодо адреси наступної команди (команди, наступної по порядку за командою відносного переходу). Існує одна команда короткого безумовного переходу SJMP (Short). Всі

команди умовного переходу використовують даний метод адресації. Відносна адреса переходу (rel) міститься в другому байті команди.

Непрямий перехід

Команда JMP @ A + DPTR дозволяє передавати управління за непрямою адресою.

Ця команда зручна тим, що надає можливість організації переходу за адресою, обчислюваній самою програмою і невідому при написанні вихідного тексту програми.

Умовні переходи

Система умовних переходів надає можливість здійснювати розгалуження за наступними умовами: акумулятор містить нуль (JZ), вміст акумулятора не дорівнює нулю (JNZ), перенесення дорівнює одиниці (JC), перенесення дорівнює нулю (JNC), адресований біт дорівнює одиниці (JB), адресований біт дорівнює нулю (JNB).

Для організації програмних циклів зручно користуватися командою DJNZ. Як лічильник циклів може використовуватися не тільки регістр, але і байт, що прямо адресується (наприклад, осередок резидентної пам'яті даних).

Команда CJNE ефективно використовується в процедурах очікування якої-небудь події. Наприклад, команда WAIT: CJNE A, P0, WAIT буде виконуватися до тих пір, поки на лініях порту 0 встановиться інформація, що збігається з вмістом акумулятора.

Всі команди даної групи, за винятком CJNE і JBC, не впливають на прапори. Команда CJNE встановлює прапор C, якщо перший операнд виявляється менше другого. Команда JBC скидає прапор C в разі переходу.

Підпрограми

Для звернення до підпрограм необхідно використовувати команди виклику підпрограм LCALL і ACALL. Ці команди на відміну від команд переходу LJMP і AJMP зберігають в стеку адреси повернення в основну програму. Для повернення з підпрограми необхідно виконати команду RET. Команда RETI відрізняється від команди RET тим, що дозволяє переривання обслуговуваного рівня.

Б.7 Операції з бітами

Відмінною особливістю даної групи команд (Таблиця Б.7) є те, що вони оперують з однобітний операндами. В якості таких операндів можуть виступати окремі біти деяких регістрів спеціальних функцій і

портів, а також 128 програмних прапорів користувача. Існують команди скидання (CLR), установки (SETB) і інверсії (CPL) бітів, а також кон'юнкції і диз'юнкції біта і прапора перенесення. Для адресації бітов використовується пряма восьмирозрядна адреса (bit). Непряма адресація бітів неможлива.

Таблиця Б.7 – Група команд операцій з бітами

Назва команди	Мнемокод	T	Операція
Скидання біта	CLR bit	4	(bit) <= 0
установка перенесення	SETB C	1	(C) <= 1
установка біта	SETB bit	4	(bit) <= 1
інверсія перенесення	CPL C	1	(C) <= !(C)
інверсія біта	CPL bit	4	(bit) <= !(bit)
Логічне І біта і перенесення	ANL C,bit	4	(C) <= (C) && (bit)
Логічне І інверсії біта і перенесення	ANL C,/bit	4	(C) <= (C) && !(bit)
Логічне АБО біта і перенесення	ORL C,bit	4	(C) <= (C) || (bit)
Логічне АБО інверсії біта і перенесення	ORL C,/bit	4	(C) <= (C) || !(bit)
Пересилання біта в перенос	MOV C,bit	4	(C) <= (bit)
Пересилання перенесення в біт	MOV bit,C	4	(bit) <= (C)