МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторного практикуму

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 лабораторного практикуму 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. – 114 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
інформаційної безпеки
та наноелектроніки
Протокол № 5
від “ 22 “ січня 2025 р.

Рекомендовано до видання
НМК ФІБЕК
Протокол №6
від “ 29 “ січня 2025 р.

ЗМІСТ

ВСТУП

Мікропроцесори і мікропроцесорні системи є в даний час найбільш масовими засобами обчислювальної техніки. Після появи перших мікропроцесорів у 1971 р., число нових розробок мікропроцесорних інтегральних схем (чіпів) продовжує безупинно збільшуватися. Досвід розробки різних систем показав, що очікувана від мікропроцесорів універсальність є в значній мірі обмеженою. Розмаїтість мікропроцесорів, що виражається в їх різній базовій технології, архітектурі, технічних характеристиках поставило розроблювачів цифрових систем перед складними проблемами, а саме: які особливості цього класу пристроїв і тенденції його розвитку, чим відрізняється використання потужних мікро-ЕОМ від застосування міні-ЕОМ, як проектувати системи на основі мікропроцесорів, 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, включаючи технічні й алгоритмічні питання, представлення особливостей мікропроцесорних інтегральних схем, а головне досвіду розробки систем на основі мікропроцесорів.

Курс "Мікропроцесорна техніка в автоматизованих системах" 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 на основі мікропроцесорних систем;

-сучасний стан мікропроцесорної техніки і перспективи розвитку.

ПІДГОТОВКА ДО ПРОВЕДЕННЯ ЛАБОРАТОРНОЇ РОБОТИ ТА ОФОРМЛЕННЯ ЗВІТУ

Лабораторна робота включає самостійне опрацювання теоретичного матеріалу, вивчення документації з апаратних засобів, методик проведення розрахунків, програмування і моделювання, обробку й інтерпретацію результатів. При проведенні лабораторного практикуму необхідно:

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

1) титульний аркуш (див. Додаток А);
2) мету роботи;
3) опис словами;
4) алгоритм роботи у вигляді блок схеми;
5) необхідні математичні розрахунки роботи пристрою;
6) текст розробленої програми;
7) основні результати моделювання пристрою;
8) 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 лабораторної роботи та відповісти на запитання викладача (отримати залік за виконану роботу).

1 ЛАБОРАТОРНА РОБОТА №1 “МІКРОКОНТРОЛЕРИ З АРХІТЕКТУРОЮ INTEL 8051. НАЙПРОСТІША ПРОГРАМА ДЛЯ МІКРОКОНТРОЛЕРА”

Мета роботи – ознайомитися з сімейством 8-розрядних мікроконтролерів 8051 і отримати навички розробки програм для них.

1.1 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 висновків. Цокольовка корпусу MCS51 і найменування виводів показані на рис.1.1. 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 виводу 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.

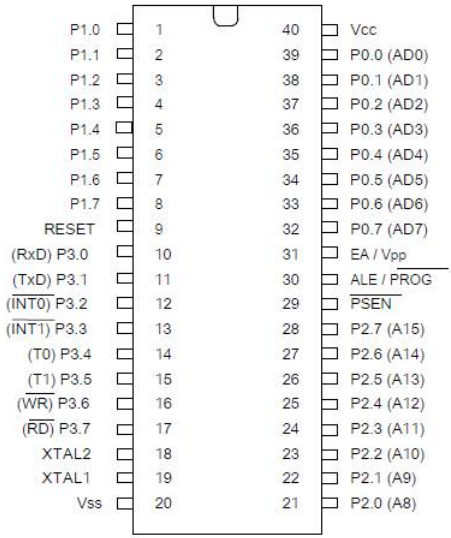


Рисунок 1.1 − Призначення виводів 8051

Позначення на Рис. 1.1:

- 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 – восьми бітний квазі двонаправлений порт введення / виводу: кожен розряд порту може бути запрограмований як на введення, так і на вивід інформації, незалежно від стану інших розрядів;
- P2 – восьми бітний 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 – восьми бітний квазі двонаправлений порт, аналогічний. P1; крім того, висновки 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.

1.2 Структура мікроконтролера

Основу структурної схеми MCS51 (рис. 1.2) 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.

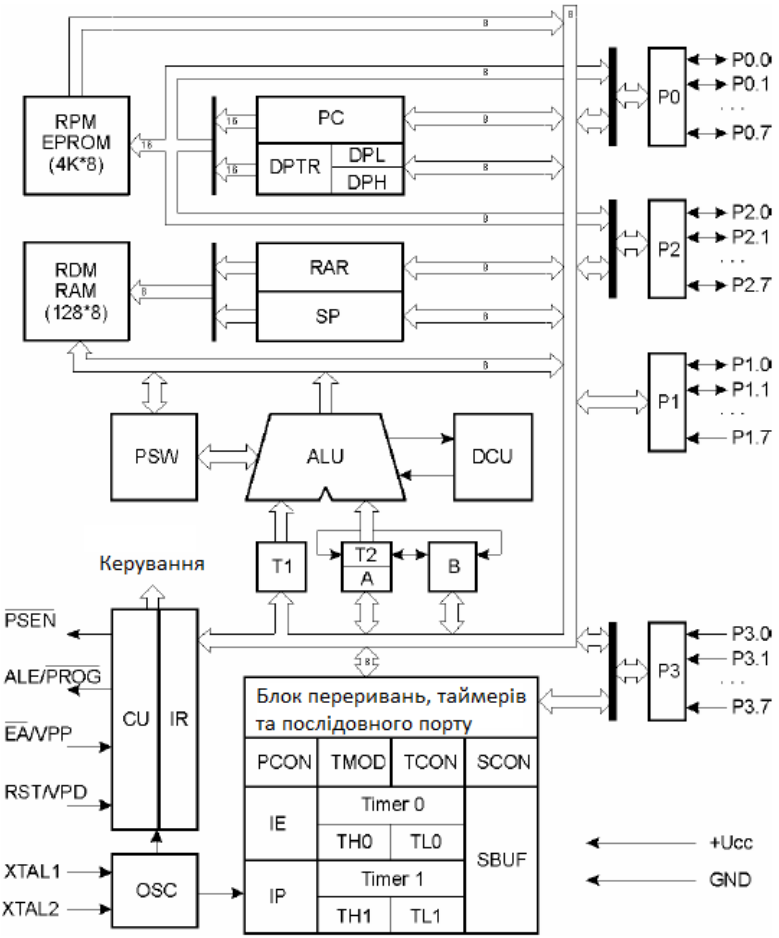


Рисунок 1.2 – Структурна схема MCS51

1.2.1 Арифметико-логічний пристрій

8-бітовий 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 десяткової 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 лічильник 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.

1.2.2 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.

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 двобайтового 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).

Вільний 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 відповідної команди в один з них повинен бути занесений 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 1.1 – 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 записана або у вигляді виразу (адреса Байта). (Розряд), наприклад вираз 21.3 означає адресу третього розряду комірки пам'яті з адресою 21H, або у вигляді абсолютної бітової адреси, яка для даного біта дорівнює (див. Таблицю 1.1) 0B.

1.2.3 Регістри спеціальних функцій

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

Таблиця 1.2 – 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 1.3.

Таблиця 1.3 – Карта адресних 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 записаний або у вигляді виразу (Назва Регістру). (Розряд), наприклад вираз SCON.3 означає адресу третього розряду регістра SCON, або у вигляді абсолютного бітового адреси, який 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 регістрові 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 інкрементується, після чого проводиться запис інформації в стек. Відповідно при добуванні інформації з стека регістр SP декрементується 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-бітного 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 B 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 1.4 наводиться перелік прапорів PSW, даються їх символічні імена і описуються умови їх формування.

Таблиця 1.4 – Формат слова стану програми PSW.

<table>
<tr><td>Символ</td><td>Розряд</td><td>Ім’я та призначення</td></tr>
<tr><td>C</td><td>PSW.7</td><td>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>Прапор переповнення. Встановлюється и скидається аппаратурно 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><td>RS0</td><td>RS1</td><td>Банк</td><td>Межі адрес ОЗУ</td></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>Прапор користувача. Може бути встановлений, скинутий або перевірений програмою користувача</td></tr>
<tr><td>-</td><td>PSW.1</td><td>Прапор 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>Прапор 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 "булевого 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 логічних операцій 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.

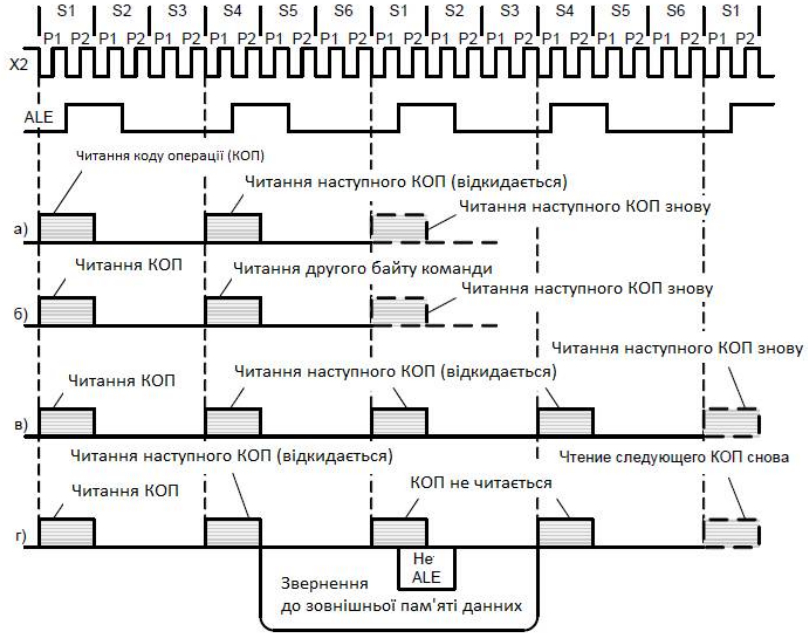
1.2.4 Пристрій управління і синхронізації

Кварцевий 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, 23 починаючи з фази S1P1 і закінчуючи фазою S6P2, як показано на рис. 1.3. 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 стрибає адреса 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 [1, 4,5].

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 циклу. 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 1.3 – Послідовності вибірки і виконання команд в MCS51

1.3 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 який 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 виводу 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 1.5. 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.

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 виводить при зверненні 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 перекладі в режим введення інформації вони стають джерелами струму для мікросхеми або транзистора, навантажених на даний вивід.

Таблиця 1.5 – 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.

1.3.1 Особливості роботи портів

Звернення до портів введення/виводу можливо з використанням команд, що оперують з байтом, окремим бітом, довільної 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, наприклад, 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 виводу 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 вивід порту

1.4 Завдання до лабораторної роботи

1 Ознайомтеся із особливостями архітектури і можливостями мікроконтролерів сімейства 8051.

2 Використовуючи тексти програм, наведені нижче і паспортні дані (datasheet) на мікроконтролер, напишіть програму на мові асемблера, яка забезпечує мерехтіння світлодіоду, підключеному за нижченаведеною схемою із періодом:

a. $T = 100 \cdot G + N_{\text{var}} \cdot 3$, мс

де: $G = \begin{cases} 3, \textit{для груп } PT-3** \\ 4, \textit{для груп } PT-4** \end{cases}$, N_{var} – номер варіанту.

b. $T = 100 \cdot G + N_{\text{var}} \cdot 3$, із скважністю $S = \left(20 + N_{\text{var}} \cdot 2\right)\%$

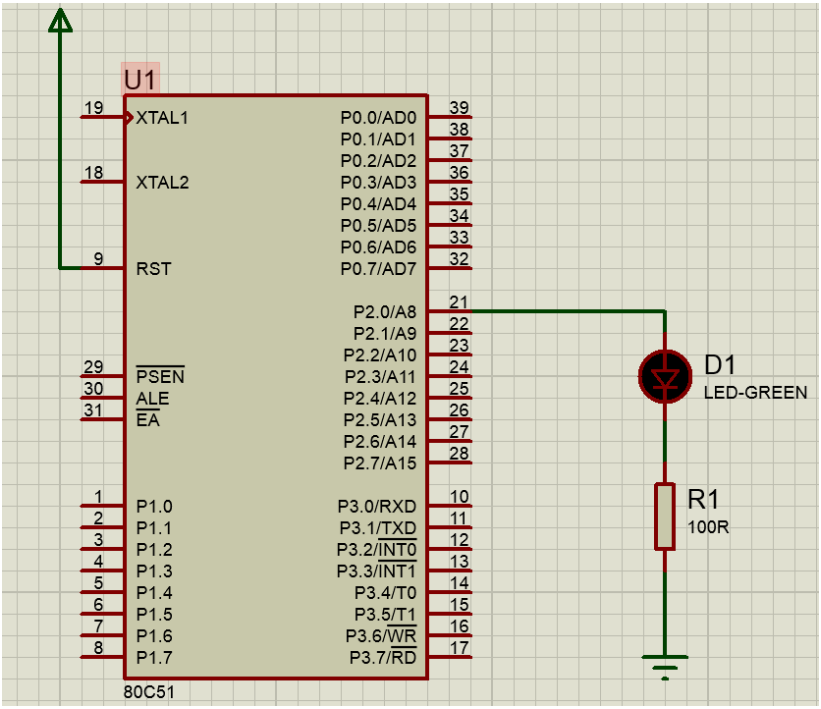


Рисунок 1.4 – Схема включення мікроконтролера

3 Оформити звіт, який має містити:

- ❖ архітектура мікроконтролера 8051;
- ❖ основні характеристики мікроконтролера 8051;
- ❖ блок-схему алгоритму роботи програми;
- ❖ текст програми на асемблері з коментарями кожного рядка програми;
- ❖ результати моделювання у вигляді принтскринів (*print screens*) осцилографу;
- ❖ відповіді на контрольні запитання;
- ❖ висновки по роботі.

1.5 Контрольні запитання

1 Яку архітектуру мають МК сімейства 8051?

2 Які основні характеристики 8051?

3 Які види пам’яті є у 8051, чим вони відрізняються?

4 Що таке стек, яке його призначення?

5 Скільки регістрів універсального призначення доступно програмісту одночасно?

6 Як здійснюється вибір між банками регістрів?

7 Що таке таблиця переривань?

8 Які типи переривань підтримує 8051?

9 Як здійснюється затримка між виконанням двох команд?

1.6 Приклад виконання завдання

1 Відкрити Proteus 8 Professional та створити новий проект (New Project). Обрати робочу папку та ввести ім’я проекту. Next.

2 Обрати Create a schematic from the selected template. Натиснути Next.

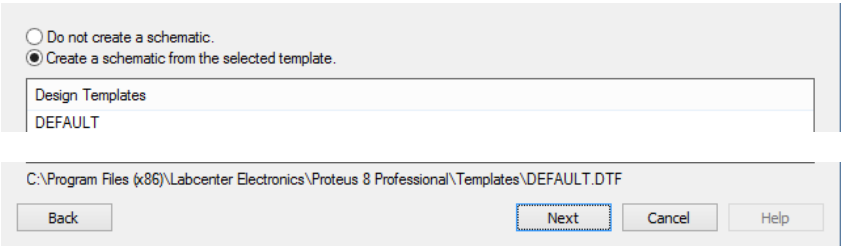


Рисунок 1.5 – Print screensпроцесу створення проекту

3 Обрати Do not create a PCB layout. Натиснути Next.

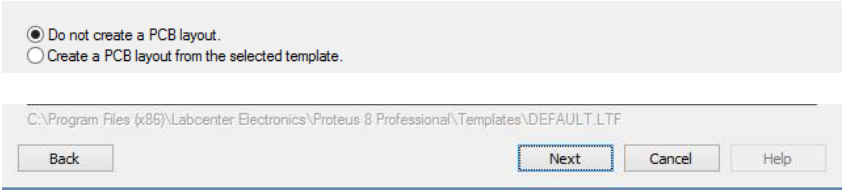


Рисунок 1.6 – Print screensпроцесу створення проекту

4 Обрати Create Firmware Project.8051. 80C51. IAR for 8051 (Assembler) Натиснути Next.

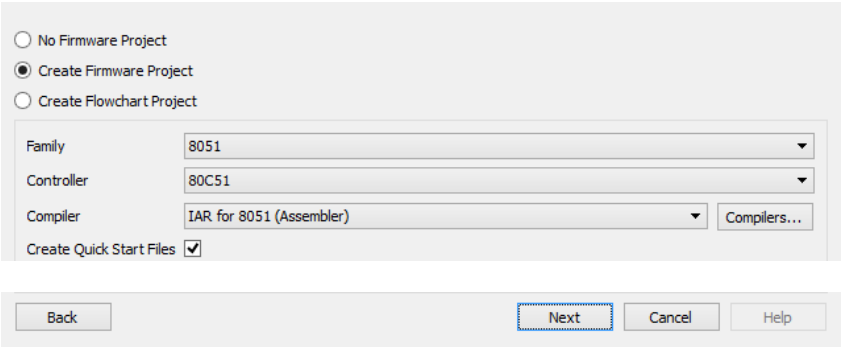


Рисунок 1.7 – Print screensпроцесу створення проекту

5 Перевірити налаштування та натиснути Finish.

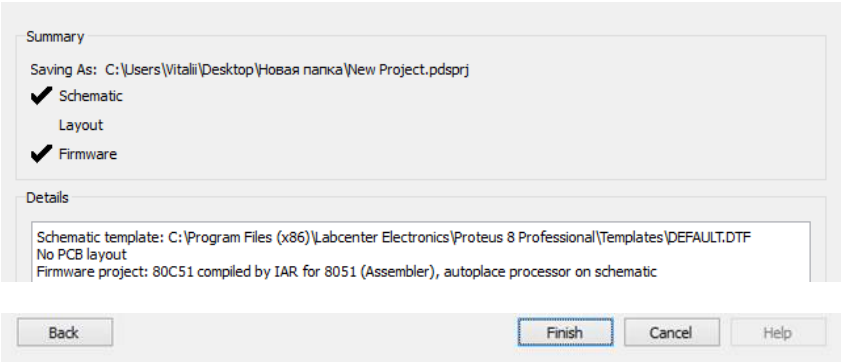


Рисунок 1.8 – Print screensпроцесу створення проекту

6 Проєкт створено, тепер використовуючи інструменти програми у вкладці **Shematic Capture** збираємо схему як на рис 1.4.

7 Далі переходимо у вкладку **Source Code**. І створюємо програму.

Лістинг прикладу програми для 1 завдання:

```
#include "io8051.h"                ; Include register definition file
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start
;=====================================================================
==
; CODE SEGMENT
```

```
;=====================================================================
==
      RSEG CODE_SEG:CODE            ; Switch to this code segment.
                                    ; Register bank 0 by default

Start:                          ;мітка, що вказує початок програми

MEANDR:
        cpl P2.0              ;встановлення піну 0 порту 2 у стан
                              ;логічного 0
        call DELAY1              ;виклик підпрограми затримки
        nop                      ;пустий оператор (пропуск 1 такту)
        nop                      ;пустий оператор (пропуск 1 такту)
        setb P2.0              ;встановлення піну 0 порту 2 у стан
                              ;логічної 1
        call DELAY1              ;виклик підпрограми затримки
        jmp MEANDR              ;повернення до мітки MEANDR
;=====================================================================
==
DELAY1:
        MOV R2, #2              ;завантаження числа X1
COUNT1: MOV R3, #254          ;завантаження числа X2
COUNT2: MOV R4, #244          ;завантаження числа X3
COUNT3: DJNZ R4, COUNT3          ;декремент R4 та цикл, якщо не рівний
0
        DJNZ R3, COUNT2          ;декремент R3 та цикл, якщо не рівний
0
        DJNZ R2, COUNT1          ;декремент R2 та цикл, якщо не рівний
0
        MOV R1, #186          ;завантаження числа X4
COUNT:  nop                      ;пустий оператор (пропуск 1 такту)
        DJNZ R1, COUNT          ;декремент R2 та цикл, якщо не рівний 0
        ret                      ;повернення з підпрограмми затримки
;=====================================================================
==
        END
```

Змінюючи числа X1-X4 ми змінюємо тривалість затримки.

Запуск програми, покроковий запуск, пауза та стоп проводяться клавішами  (лівий нижній кут).

Навігація по програмі здійснюється клавішами  (правий верхній кут):

-  – перехід на крок вперед (функція);
-  – перед, до входу в підпрогаму;
-  – перед, до виходу з підпрограми;

-  – перехід на вказаний рядок;
-  – встановлення мітки.

Отримати точний час виконання програми встановлення/скидання біту можна двома способами:

- Встановивши мітки, як показано на рис 1.2, та переходити між мітками (Ctrl+F11), у рядку стану буде вказаний час переходу;
- Використовуючи осцилограф, виміряти тривалість імпульсу.

Рисунок 1.9 – Приклад відладки програми

Лістинг прикладу програми для 2 завдання:

```
#include "io8051.h"                ; Include register definition file
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start
;=========================================================================
; CODE SEGMENT
;=========================================================================
      RSEG CODE_SEG:CODE            ; Switch to this code segment.
                                    ; Register bank 0 by default

Start:                              ;мітка, що вказує початок програми
MEANDR:     cpl P2.0              ;встановлення піну 0 порту 2 у стан
                                  ;логічного 0
            call DELAY1             ;виклик підпрограми затримки
            nop                     ;пустий оператор (пропуск 1 такту)
            nop                     ;пустий оператор (пропуск 1 такту)
            setb P2.0              ;встановлення піну 0 порту 2 у стан
                                  ;логічної 1
            call DELAY2             ;виклик підпрограми затримки
            jmp MEANDR             ;повернення до мітки MEANDR
;=========================================================================
DELAY1:     MOV R2, #2            ;завантаження числа X1
COUNT11:    MOV R3, #254            ;завантаження числа X2
COUNT12:    MOV R4, #244            ;завантаження числа X3
COUNT13:    DJNZ R4, COUNT13      ;декремент R4 та цикл, якщо не рівний
0
            DJNZ R3, COUNT12      ;декремент R3 та цикл, якщо не рівний
0
            DJNZ R2, COUNT11      ;декремент R2 та цикл, якщо не рівний
0
            MOV R1, #186            ;загрузка числа X1
COUNT1:     nop                 ;пустий оператор (пропуск 1 такту)
            DJNZ R1, COUNT1           ;декремент R2 та цикл, якщо не
рівний 0
            ret                 ;повернення з підпрограмми затримки
;=============================================================
DELAY2:     MOV R2, #4            ;завантаження числа X1
COUNT21:    MOV R3, #255            ;завантаження числа X2
COUNT22:    MOV R4, #243            ;завантаження числа X3
COUNT23:    DJNZ R4, COUNT23      ;декремент R4 та цикл, якщо не рівний
0
            DJNZ R3, COUNT22      ;декремент R3 та цикл, якщо не рівний
0
            DJNZ R2, COUNT21      ;декремент R2 та цикл, якщо не рівний
0
            MOV R1, #200            ;загрузка числа X1
COUNT2:     nop
            nop
            nop
            nop                 ;пустий оператор (пропуск 1 такту)
            DJNZ R1, COUNT2           ;декремент R2 та цикл, якщо не
рівний 0
```

```
            ret             ;повернення з підпрограмми затримки
;===============================================================================
            END
```

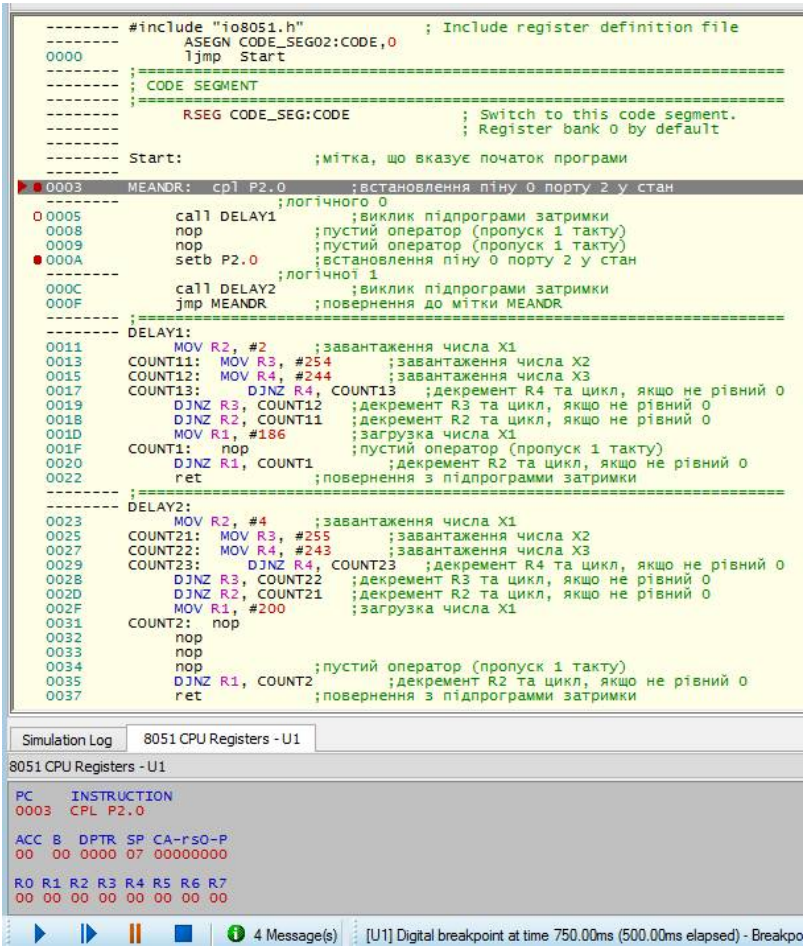


Рисунок 1.10 – Приклад відладки програми

2 ЛАБОРАТОРНА РОБОТА №2 “СИСТЕМА ПЕРЕРИВАНЬ. ТАЙМЕРИ МК 8051”

Мета роботи – ознайомитися із системою переривань мікроконтролерів сімейства 8051; вивчити режими роботи таймерів і навчитися використовувати таймери для організації затримок.

2.1 Система переривань мікроконтролера 8051

Спрощена схема переривань мікро-ЕОМ 8051 показана на рис. 2.1.

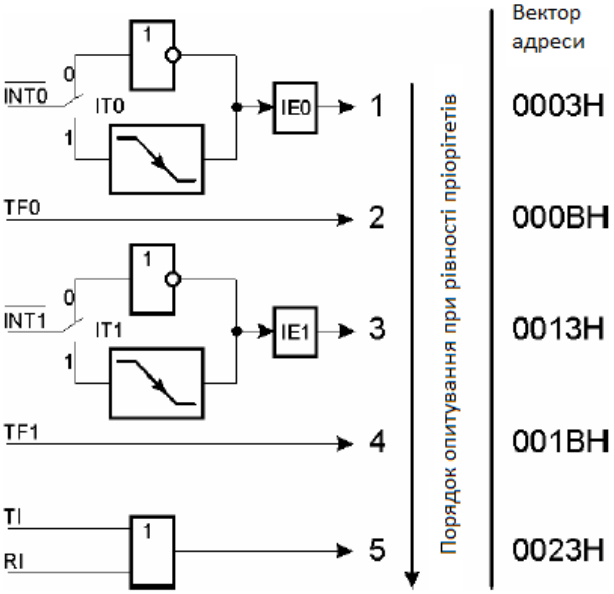


Рисунок 2.1 – Схема переривань

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. 2.1. Схема переривань встановлені / скинуті програмою з тим же результатом, як якщо б 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 2.1 і 2.2 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 2.1 – 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 2.2 – 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

2.1.1 Виконання підпрограми переривання

Система переривань формує апаратний виклик (LCALL) відповідної підпрограми обслуговування, якщо вона не заблокована одною 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 адресу 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 перерваній основній програмі, але при цьому не зніме блокування переривань, що приводить до необхідності мати програмний механізм аналізу закінчення процедури обслуговування даного переривання.

2.2 Таймери/лічильники мікроконтролерів сімейства 8051

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) інкрементується в кожному машинному циклі, тобто через кожні 12 періодів коливань кварцевого резонатора, у другому воно інкрементується під впливом переходу з 1 в 0 зовнішнього вхідного сигналу, що подається на відповідний (T0, T1) вивід мікро-ЕОМ 8051. Опитування значення зовнішнього вхідного сигналу виконується в момент часу S5P2 кожного машинного циклу . 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 (інкрементоване) 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 T / C і для організації їх взаємодії з системою переривань використовуються два регістри спеціальних функцій (TMOD і TCON), опис яких наведено в табл. 7 і 8 відповідно.

Таблиця 2.3 – Регістр режиму работи таймера/лічильника TMOD

Символ	Позиція	ім’я та призначення		
GATE	TMOD.7 для T/C1 и TMOD.3 для T/CO	Управління блокуванням. Якщо біт встановлено, то таймер/лічильник "x" дозволений доти , поки на вході "INTx" високий рівень і біт управління "TRx" встановлено. Якщо біт скинуто, то T/C дозволяється, як тілько біт управління "TRx" встановлюється		
C/T	TMOD.6 для T/C1 и TMOD.2 для T/CO	Біт вибору режиму таймеру або лічильника подій. Якщо біт скинуто, працює таймер від внутрішнього джерела сигналів синхронізації. Якщо встановлений, працює лічильник від зовнішніх сигналів на вході "Tx"		
M1	TMOD.5 для T/C1 и TMOD.1 для T/CO	Режим работи		
		M1	M0	
		0	0	Таймер BE48. "TLx" працює как 5-бітний переддільник
		0	1	16 бітный таймер/лічильник. "THx" и "TLx" включені послідовно
M0	TMOD.4 для T/C1 и TMOD.0 для T/CO	1	0	8-бітний авто перезавантажуваний таймер/лічильник. "THx" зберігає значення, яке має завантажуватись в "TLx" щоразу при переповненні
		1	1	Таймер/лічильник 1 зупиняється. Таймер/лічильник 0: TLO працює як 8-бітный таймер/лічильник, і його режим визначається управляючими бітами таймера 0. TH0 працює тільки як 8 бітний таймер, і його режим визначається управляючими бітами таймера 1

Таблиця 2.4– Регістр управління/статуса таймера TCON.

Символ	Позиция	Имя и назначение
TF1	TCON.7	Прапор переповнення таймера 1. Встанавлюється апаратно при переповненні таймеру/лічильника. Скидається при обслуговуванні переривання апаратно
TR1	TCON.6	Біт управління таймера 1. Встановлюється, / скидається програмою для пуска/зупинки
TF0	TCON.5	Прапор переповнення таймера 0. Встановлюється апаратно. Скидається при обслуговуванні переривання
TR0	TCON.4	Біт управління таймера 0. Встановлюється, / скидається програмою для пуска/зупинки таймера/лічильника
IE1	TCON.3	Прапор фронту переривання 1. Встановлюється апаратно, коли детектується зріз зовнішнього сигналу INT1 Скидається при обслуговуванні переривання

Продовження таблиці 2.4

IT1	TCON.2	Біт управління типом переривання 1. Встановлюється / скидається програмно для специфікації запиту INT1 (зріз/низький рівень)
IE0	TCON.1	Прапор фронту переривання 0. Встановлюється по зрізу сигналу INT0 Скидається при обслуговуванні переривання
IT1	TCON .0	Біт управлінія типом переривання 0. Встановлюється / скидається програмно для специфікації запиту INT0 (зріз/низький рівень)

2.2.1 Режими роботи таймерів-лічильників

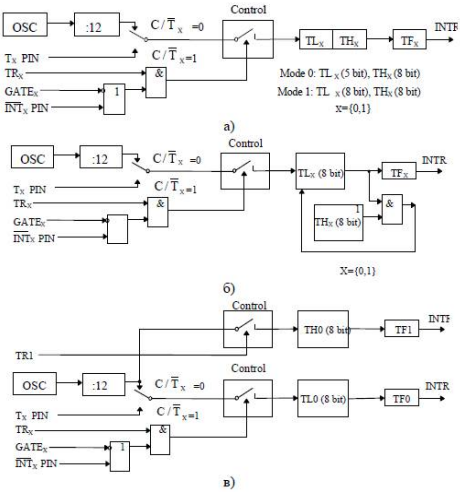
Як випливає з опису керуючих біт TMOD, для обох Т / З режими роботи 0, 1 і 2 однакові. Режими 3 для Т/З і Т/С1 різні. Розглянемо коротко роботу Т / С в кожному з режимів [1].

- 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 вивід 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. 12, 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 T/C0 і T/C1 також працюють абсолютно однаково.

- Режим 3. У режимі 3 T/C0 і T/C1 працюють по-різному. T/C1 зберігає незмінним свій поточний зміст. Іншими словами, ефект такий же як і при скиданні керуючого біта TR1 в 0. Роботу T/C0 ілюструє рис. 2.2, ст.39. У режимі 3 TL0 і TH0 функціонують як два незалежних 8-бітових лічильника. Роботу TL0 визначають керуючі біти T / C0 (C / T, 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 T / C0 використовується в режимі 3, T / C1 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 T/C2 і (або) блок програмних лічильників PCA, які теж можуть бути використані для відліку часових інтервалів.



а – T/C0 та T/C1 у режимах 0 та 1; б – T/C0 та T/C1 у режимі 2; в – T/C0 у режимі 3.

Рисунок 2.2 – таймери-лічильники T/C0 та T/C1 у режимах 0, 1, 2 та 3

2.3 Завдання до роботи

1 Вивчити документацію до мікроконтролера 80C51, визначити наявні в цій моделі таймери/лічильнки та їх можливості.

2 Скласти програму на асемблері, яка дозволяє сформувати послідовність прямокутних імпульсів на виході будь-якого порту мікроконтролера із параметрами:

a. $T = 100 \cdot G + N_{\text{var}} \cdot 3$, мкс

b. де: $G = \begin{cases} 3, для\ груп\ PT-3** \\ 4, для\ груп\ PT-4** \end{cases}$, N_{var} – номер варіанту.

c. $T = 100 \cdot G + N_{\text{var}} \cdot 3$, із скважністю $S = \left(20 + N_{\text{var}} \cdot 2\right)\%$

d. Розробити програму для виконання данної лабораторної роботи, що буде використовувати лише 1 таймер для генерації відповідно до завдання b.

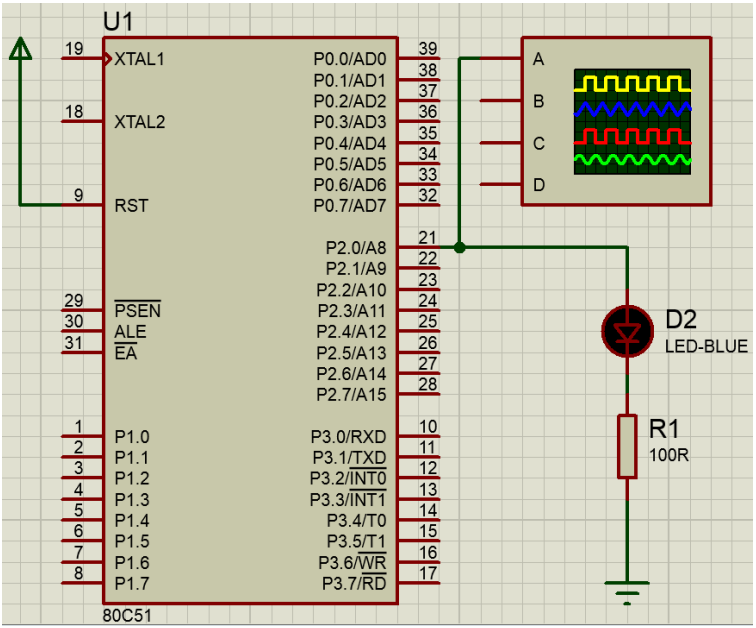


Рисунок 2.3 – Схема підключення мікро контролера

3 Скласти звіт з роботи, у якому слід розмістити:
- алгоритм роботи програми у вигляді блок схеми;
- схему електричну принципову;
- текст програми на асемблері;
- результати моделювання програми;
- висновки.

2.4 Контрольні запитання

1 Характеристика наявних у 8051 таймерів.
2 Як здійснити конфігурацію таймера?
3 Робота таймера у асинхронному режимі.
4 Як здійснити конфігурацію переддільника?
5 Які перериванні доступні у 8051 при роботі з таймерами?

2.5 Приклад виконання завдання

1 Створити проект як в прикладі 1.6 та використовуючи інструменти програми у вкладці **Shematic Capture** зібрати схему як на рис 2.3.

Лістинг прикладу програми для 1 завдання:

```
CLOCK = 1000
;=====================================================================
#include "io8051.h"                ; Include register definition file
      ASEGN CODE_SEG02:CODE,0
      jmp  Start
      ASEGN CODE_SEG02:CODE,0x0B
      jmp  timer0              ;Вектор обробки переррування таймера 1
;=====================================================================
      RSEG CODE_SEG:CODE         ; Switch to this code segment.
                                 ; Register bank 0 by default
Start:                 ;мітка, що вказує початок програми
        call init_timer
loop:
        jmp loop                              ;нескінченний цикл
init_timer:
        MOV  TL0, #LOW(65535-CLOCK);молодший байт 16-ти бітного числа
        MOV  TH0, #HIGH(65535-CLOCK);старший байт 16-ти бітного числа
        clr  TMOD.0                  ;16-бітний таймер/лічильник,
        setb TMOD.1                  ;TL0 та TH0 ввімкнені послідовно
        clr  TMOD.2                  ;таймер працює від внутрішнього
джерела
                                                          ;сигналів
синхронізації
        clr  TMOD.3                  ;робота таймера дозволяється
якщо

       ;встановлений керуючий біт TCON_TR0
        setb TCON_TR0                ;запуск таймера 0
        setb IE_ET0                  ;дозвіл переривання по таймеру 0
        setb IE_EA                   ;глобальний дозвіл препивань
        ret
;=====================================================================
timer0:
        MOV  TL0, #LOW(65535-CLOCK);молодший байт 16-ти бітного числа
        MOV  TH0, #HIGH(65535-CLOCK);старший байт 16-ти бітного числа
        cpl P2.0              ;інверсія піну 0 порту 2
        reti
;=====================================================================
          END
```

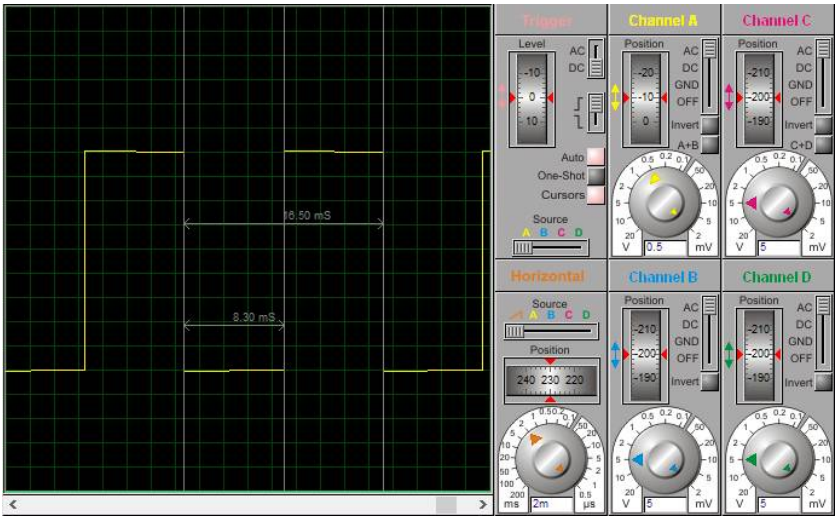


Рисунок 2.4 – зображення з екрану осцилографа

Лістинг прикладу програми для 2 завдання:

```
CLOCK = 65535  ;тривалість періоду
CLOCKS = 32000 ;тривалість імпульсу

#include "io8051.h"                ; Include register definition file
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start
      ASEGN CODE_SEG02:CODE,0x0B
      ljmp  timer0              ;Вектор обробки переривання таймера 0
      ASEGN CODE_SEG02:CODE,0x1B
      ljmp  timer1              ;Вектор обробки переривання таймера 1
;=======================================================================
      RSEG CODE_SEG:CODE          ; Switch to this code segment.
                                  ; Register bank 0 by default
Start:                           ;мітка, що вказує початок програми
 lcall init_timer
 setb TCON_TR0          ;запуск таймера 0
loop:
 call loop              ;нескінченний цикл
init_timer:
 clr  TMOD.0            ;16-бітний таймер/лічильник,
 setb TMOD.1            ;TL0 та TH0 ввімкнені послідовно
 clr  TMOD.2            ;таймер працює від внутрішнього джерела
                        ;сигналів синхронізації
 clr  TMOD.3            ;робота таймера дозволяється якщо
                        ;встановлений керуючий біт TCON_TR0
 clr  TMOD.4            ;16-бітний таймер/лічильник,
```

```
 setb TMOD.5           ;TL1 та TH1 ввімкнені послідовно
 clr  TMOD.6           ;таймер працює від внутрішнього джерела
                       ;сигналів синхронізації
 clr  TMOD.7           ;робота таймера дозволяється якщо
                       ;встановлений керуючий біт TCON_TR1
 setb IE_ET0           ;дозвіл переривання по таймеру 0
 setb IE_ET1           ;дозвіл переривання по таймеру 1
 setb IE_EA            ;глобальний дозвіл препивань
 ret
;=====================================================================
timer0:
 setb P2.0             ;інверсія піну 0 порту 2
 MOV  TL1, #LOW(65535 - CLOCKS);молодший байт 16-ти бітного числа
 MOV  TH1, #HIGH(65535 - CLOCKS);старший байт 16-ти бітного числа
 clr  TCON_TR0         ;зупинка таймера 0
 setb TCON_TR1         ;запуск таймера 1
 reti
;=====================================================================
timer1:
 clr  P2.0             ;інверсія піну 0 порту 2
 MOV  TL0, #LOW(65535 - CLOCK);молодший байт 16-ти бітного числа
 MOV  TH0, #HIGH(65535 - CLOCK);старший байт 16-ти бітного числа
 clr  TCON_TR1         ;зупинка таймера 1
 setb TCON_TR0         ;запуск таймера 0
 reti
;=====================================================================
         END
```

3 ЛАБОРАТОРНА РОБОТА №3 „ЗАСОБИ ВВОДУ-ВИВОДУ ІНФОРМАЦІЇ В МПС. LED-ІНДИКАТОРИ”

Мета роботи – ознайомитися із загальними принципами організації системи вводу і виводу інформації в мікропроцесорних системах.

3.1 Принципи побудови системи вводу-виводу

3.1.1 Загальна інформація про індикатори у МПС

Відображення інформації є важливою частиною будь-якого інформаційного процесу вимірювання, управління і контролю. Інформація, яку сприймає людина, має подаватись в ефективній формі з мінімальними вимогами до перекодування і перерахунку інформації. Тому цифрове відтворення інформації у багатьох випадках краще аналогового.

У цифровій техніці застосовують індикатори, які базуються на різних принципах дії. *Індикатором* називають вихідний пристрій інформаційного приладу або системи, що забезпечує візуальне відображення інформації.

Індикатори класифікують за принципом формування зображення на знакомодулюючі (ЗМІ) і знакосинтезуючі (ЗСІ).

Знакомодулюючі індикатори містять певний набір знаків і відображають лише один з них, наприклад, газорозрядні індикатори.

Знакосинтезуючі індикатори містять набір сегментів, за допомогою якого проводиться синтез великої кількості різних знаків, наприклад, вакуумні люмінесцентні, напівпровідникові, рідкокристалічні. Серед знакосинтезуючих індикаторів розрізняють сегментні і матрич-ні індикатори.

Сегментні індикатори – це індикатори, елементи відображення яких є сегменти і згруповані в одне (однорозрядний) або декілька (багаторозрядний) знакомісць.

Матричні індикатори (або інакше - графічні) складаються з набору окремих точок. Призначені для відображення будь-якої інформації, включаючи символи і графіку. Їх використовують для побудови високоергономічних відображаючих пристроїв.

У сучасній електронній техніці найбільш поширені такі типи індикаторів: світлодіодні (LED – light emission diode) і рідкокристалічні індикатори (LCD – liquid crystal display). Поширеність LCD обумовлена в порівнянні з іншими типами індикаторів майже ідеальними електричними характеристиками. Світлодіодні індикатори мають низьку напругу живлення (1,5…3,5 В), що зручно, проте їх споживаний струм досить великий (2 - 20 мА), і це практично "ставить хрест" на їх використанні в сучасній мікропотужній радіоелектронній апаратурі. LCD при робочій напрузі 3-5 В споживають малі струми - долі мікроампера. Керування LCD здійснюється змінною напругою, але для сучасної техніки це - не проблема. На відміну від решти індикаторів вони практично нечутливі до електричних перевантажень. І ще одна особливість: LED-індикатори випромінюють світло, а LCD – поглинають.

Використання того чи іншого індикатора залежить від багатьох факторів: умов роботи, вартості, вимог споживачів готової продукції.

LED-індикатори набули поширення завдяки зручності, наочності і невисокій вартості. Цифрові і знакові LED-індикатори є впорядкованим набором окремих світлодіодів (сегментів індикатора), розміщених в одному корпусі і, зазвичай, сполучених між собою в певну електричну схему (рис. 3.1а).

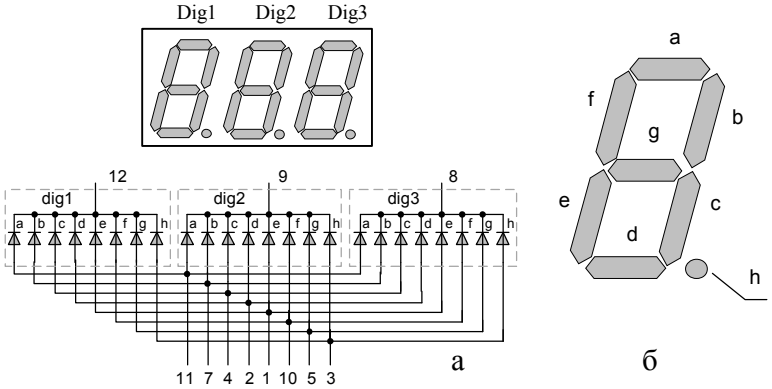


Рисунок 3.1 - Будова семисегментних LED-індикаторів

Цифрові і знакові індикатори бувають однорозрядними і багаторозрядними, їх знаки складаються з семи і більш сегментів.

Окрім сегментів індикатори мають «крапку» або «кому». На рис. 3.1,б наведено стандартне позначення сегментів індикатора. Висвітлення обраного сегменту або групи сегментів при отриманні зображення знаку забезпечується включенням їх в ланцюг проходження струму.

Залежно від схеми включення світлодіодів в індикаторі розрізняють індикатори із загальним катодом (рис. 3.1,а) і загальним анодом. За способом включення індикаторів в електричну схему можна виділити два режими: статична і динамічна індикація.

LCD-індикатори конструктивно виконуються як сегментні, наприклад (ИЖКЦ1-4/14), або матричні. Останні внаслідок необхідності роботи на змінному струмі, зазвичай виконуються у вигляді готового модуля на друкованій платі із спеціальним контролером (наприклад HD44780 для знакосинтезуючих і S6B0108 для матричних).

3.1.2 Види індикації LED-індикаторів

Спосіб ***статичної індикації*** полягає в постійному підсвічуванні індикатора від одного джерела інформації (рис. 3.2, 3.3), тобто кожен з цифрових індикаторів блоку індикації постійно підключений через власний перетворювач коду (наприклад, дешифратор) до власного джерела інформації (лічильник або МК).

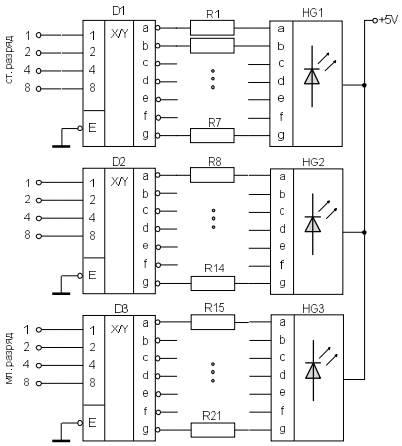


Рисунок 3.2 – Схема пристрою з блоком статичної індикації на індикаторах із загальним анодом

Коли мікроконтролер є джерелом інформації і перетворювачем інформації, індикатори приєднують безпосередньо до нього через струмообмежувальні резистори (рис. 3.3).

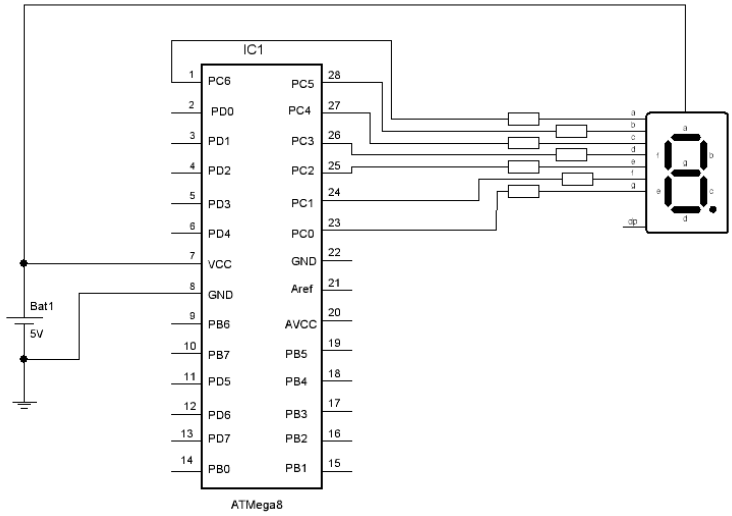


Рисунок 3.3 – Схема МПС із статичною індикацією: один індикатор із загальним анодом

При кількості індикаторів більше 4 технічна реалізація режиму статичної індикації стає недоцільною. Це пов'язано із збільшенням необхідної кількості перетворювачів коду, резисторів і дротів для сполучення, а також значним збільшенням споживаного струму (переважно, індикаторами).

Крім того, деякі види індикаторів, наприклад матричні світлодіодні або рідкокристалічні індикатори, зображення графічних символів

в яких отримують при подачі імпульсної напруги на впорядкованих в стовпці і рядки певні точки матриці, не розраховані на застосування в системах статичної індикації.

Динамічна індикація полягає в наступному**.** Для зменшення загальної кількості провідників, які зв'язують між собою мікросхеми індикатора, цифрові індикатори частіше почали виготовляти багаторозрядними блоками, біля яких в єдиному корпусі може бути

від 2 до 16 цифрових знаків. Для скорочення кількості виводів у багаторозрядному індикаторі всі однойменні сегменти з'єднані разом і мають один загальний вивід (рис. 3.1, а).

Для забезпечення роботи багаторозрядних індикаторів або декількох одноразрядних індикаторів використовують динамічну індикацію, суть якої полягає в тому, що розряди індикатора працюють не одночасно, а по черзі. Якщо перемикання індикаторів здійснюється з великою швидкістю, то людина не помічає мерехтіння, і картинка виглядає як статична. Індикатори підключають з частотою 50÷70 Гц, що достатньо для того, щоб не помічати мерехтіння індикаторів. Загальну структурну схему динамічної індикації наведено на рис. 3.4.

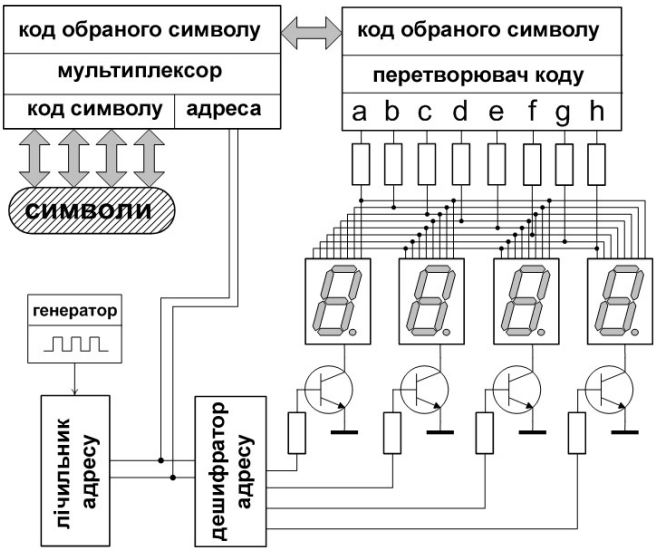


Рисунок 3.4 - Структурна схема блоку динамічної індикації на 4 знаки

Інформація поступає не на декілька перетворювачів, а на один, загальний для всіх, але порціями. Цей перетворювач коду своїми виходами підключений до всіх елементів відразу. У системі динамічної індикації мов би працює швидкодіючий безконтактний перемикач на багато положень. У одному з його положень всі виводи (сегменти) одного з розрядів підключені до відповідних виходів дешифратора. І в цей же момент через транзисторні ключі поступає

сигнал управління засвіченням елементів того знакомісця, яке відповідає цьому розряду.

Коли МПС виконує всі завдання з перетворювання коду і вибору адреси знакомісця, схема підключення значно спрощується (рис.3.5).

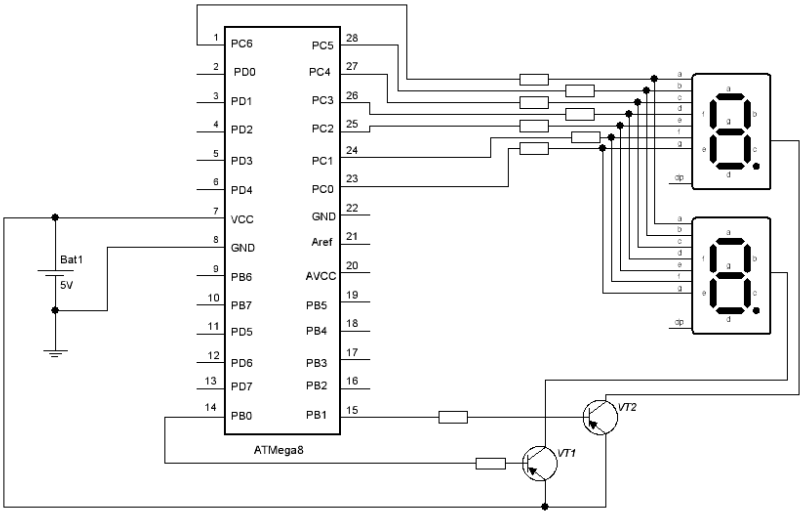


Рисунок 3.5 – Організація динамічної індикації дворозрядного індикатора

Перевага способу динамічної індикації відчутна при кількості розрядів більше 4. Економія перетворювачів коду і дротів сполучення є значною, якщо схема динамічної цифрової індикації віддалена від джерела інформації. Схема з динамічною індикацією споживає менший струм, має менші габарити і вартість. Однак при організації великої кількості розрядів виникає проблема яскравості індикаторів у зв'язку з малим часом світіння кожного розряду індикатора.

3.2 Завдання до лабораторної роботи

1 Розробити принципову схему і алгоритм, та написати програму для пристрою на мікроконтролері 80С51, яка б дозволяла:

- генерувати задану послідовність імпульсів заданої сквапності $S = \left(20 + N_{\text{var}} \cdot 2\right)\%$

- змінювати у заданих межах скважність при натисненні на клавіші $S = \pm(10 + N_{\text{var}})\%$

- відображати змінюваний параметр на семисегментних дисплеях;

Розробити принципову схему і алгоритм, та написати програму для пристрою на мікроконтролері 80С51, яка б дозволяла за допомогою динамічної індикації відображати дату. Студент бере дату свого народження.

2 Провести моделювання роботи приладу.

3 Оформити звіт за вимогами. Звіт повинен містити:

- титульну сторінку, тему і мету роботи;
- необхідний теоретичний матеріал;
- розрахунки і пояснення;
- електричні принципові схеми включення МПС;.
- блок-схему алгоритму роботи програми;
- лістинг програми на асемблері із поясненнями;
- результати моделювання: *print screen* екрану у потрібні моменти часу;
- висновки по роботі;
- відповіді на контрольні запитання.

3.3 Контрольні запитання

1 Структура статичної індикації, особливості, переваги і недоліки.

2 Структура динамічної індикації, особливості, переваги і недоліки.

3 Якою має бути частота оновлення для динамічної індикації?

4 Ефект брязкоту контактів, способи його усунення.

5 Організація індикації восьмирозрядного індикатора.

3.4 Приклад виконання завдання

1 Створити проект як в прикладі 1.6 та використовуючи інструменти програми у вкладці **Shematic Capture** зібрати схему як на рис.:

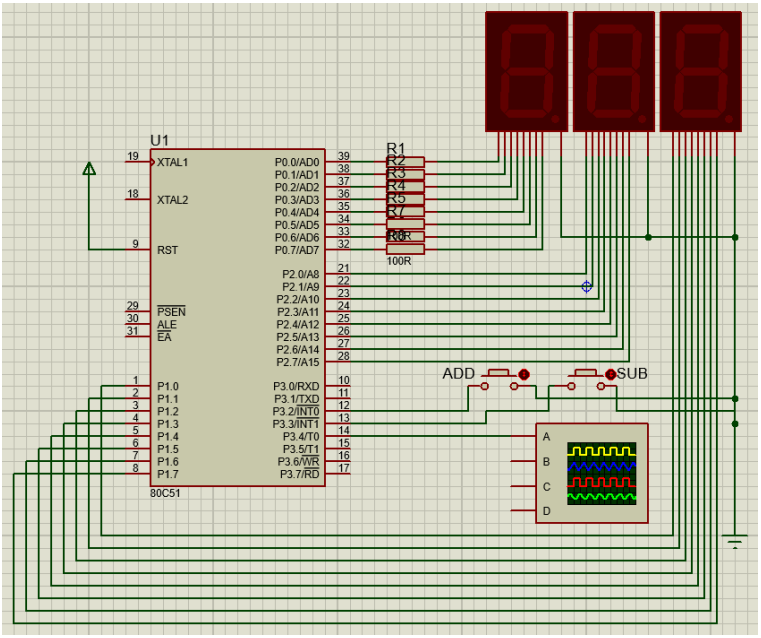


Рисунок 3.6 – Схема підключення мікро контролера

Лістинг прикладу програми для 1 завдання:

```
#include "io8051.h"                ; Include register definition file
       ASEGN CODE_SEG02:CODE,0
       jmp   Start
       ASEGN CODE_SEG02:CODE,0x0B
       jmp   timer0      ;Вектор обробки переррування таймера 0
       ASEGN CODE_SEG02:CODE,0x03
       jmp   Interrupt0  ;Вектор обробки зовнышнього переррування 0
       ASEGN CODE_SEG02:CODE,0x1B
       jmp   timer1      ;Вектор обробки переррування таймера 1
       ASEGN CODE_SEG02:CODE,0x13
       jmp   Interrupt1  ;Вектор обробки зовнышнього переррування 0
;=====================================================================
       RSEG CODE_SEG:CODE          ; Switch to this code segment.
                                   ; Register bank 0 by default
Start:                            ;мітка, що вказує початок програми
        call init
loop:
        jmp loop                  ;нескінченний цикл
init:
        MOV  TL0, 0x00                  ;задання періоду
        MOV  TH0, 0x00                  ;імпульсі
        MOV ACC, #255             ;зміна
```

```
        SUBB A, R7              ;скважності
        MOV  B, 0xA0            ;імпульсів за
        MOV  TL1, B             ;допомогою
        MOV  TH1, ACC           ;коеффіцієнта
        clr  TMOD.0             ;16-бітний таймер/лічильник,
        setb TMOD.1             ;TL0 та TH0 ввімкнені послідовно
        clr  TMOD.2             ;таймер працює від внутрішнього джерела
                                ;сигналів синхронізації
        clr  TMOD.3             ;робота таймера дозволяється якщо
                                ;встановлений керуючий біт TCON_TR0
        clr  TMOD.4             ;16-бітний таймер/лічильник,
        setb TMOD.5             ;TL1 та TH1 ввімкнені послідовно
        clr  TMOD.6             ;таймер працює від внутрішнього джерела
                                ;сигналів синхронізації
        clr  TMOD.7             ;робота таймера дозволяється якщо
                                ;встановлений керуючий біт TCON_TR1
        setb TCON_TR0           ;запуск таймера 0
        setb TCON_TR1           ;запуск таймера 1
        setb IE_ET0             ;дозвіл переривання по таймеру 0
        setb IE_ET1             ;дозвіл переривання по таймеру 1
;=====================================================================
        setb TCON.0             ;перепад сигналу 1-0
        setb IE_EX0             ;дозвіл переривання по зовнігшньому
                                ;виводу 0
        setb TCON.2             ;перепад сигналу 1-0
        setb IE_EX1             ;дозвіл переривання по зовнігшньому
                                ;виводу 1
;=====================================================================
        setb IE_EA              ;глобальний дозвіл препивань
;=====================================================================
        mov R7, #95
        call print
        ret
;=====================================================================
timer0:
        MOV  TL0, 0x00                  ;задання періоду
        MOV  TH0, 0x00                  ;імпульсі
        MOV ACC, #255           ;зміна
        SUBB A, R7              ;скважності
        MOV  B, 0xA0            ;імпульсів за
        MOV  TL1, B             ;допомогою
        MOV  TH1, ACC           ;коеффіцієнта
        setb P3.4               ;початок імпульсу
        reti
timer1:
        clr  P3.4               ;скидання імпульсу
        reti
Interrupt0:                     ;збільшення значення в R7 на одиницю
        mov a, R7               ;перевірка
        SUBB a, #199            ;на
        JZ IN0rel00             ;вихід
        inc R7                  ;з інтервалу
IN0rel00:
```

```
        call print          ;оновлення
        reti
Interrupt1:                 ;зменшення значення в R7 на одиницю
        mov a, R7           ;перевірка
        SUBB a, #0          ;на
        JZ IN1rel00         ;вихід
        dec R7              ;з інтервалу
IN1rel00:
        call print          ;оновлення
        reti
print:
        MOV a, R7           ;виймання значення з R7
        call DA_corr        ;двійково-десяткова корекція
        mov dptr, #code     ;показник на кодуючу таблицю
        MOV a, R0           ;кодування і
        movc a, @a+dptr     ;відображення 3-го
        MOV  P0, a          ;розряду
        MOV a, R1           ;кодування і
        movc a, @a+dptr     ;відображення 2-го
        MOV  P2, a          ;розряду
        MOV a, R2           ;кодування і
        movc a, @a+dptr     ;відображення 1-го
        MOV  P1, a          ;розряду
        ret
DA_corr:
        MOV R0, #0          ;очищення регітсрів
        MOV R1, #0          ;для збереження
        MOV R2, #0          ;результату
DA_rel00:
        clr C               ;очищення біту переносу
        MOV R5, a
        SUBB a, #100
        JC DA_rel01
        inc R0
        jmp DA_rel00
DA_rel01:
        MOV a, R5
DA_rel02:
        clr C               ;очищення біту переносу
        MOV R5, a
        SUBB a, #10
        JC DA_rel03
        inc R1
        jmp DA_rel02
DA_rel03:
        MOV a, R5
        MOV R2, a
        ret
;=====================================================================
code: db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F,
0x00
;=====================================================================
          END
```

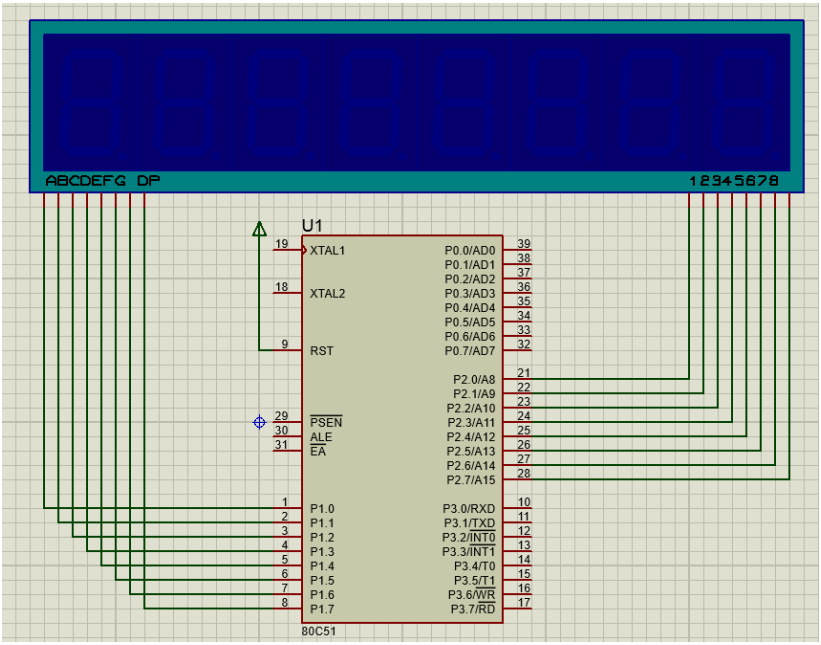


Рисунок 3.7 – Схема підключення мікро контролера

Лістинг прикладу програми для 2 завдання:

```
#include "io8051.h"                ; Include register definition file

      ASEGN CODE_SEG02:CODE,0
      jmp   Start
      ASEGN CODE_SEG02:CODE,0x0B
      jmp   timer0                 ;Вектор обробки переррівання таймера 1

;=======================================================================
      RSEG CODE_SEG:CODE             ; Switch to this code segment.
                                     ; Register bank 0 by default
Start:                               ;мітка, що вказує початок програми
        call init
loop:   call print
        jmp loop                     ;нескінченний цикл
init:
        mov P1, #0xFF
        mov P2, #0xFF
        MOV  TL0, 0x00
        MOV  TH0, 0x00
        clr  TMOD.0                  ;16-бітний таймер/лічильник,
        setb TMOD.1                  ;TL0 та TH0 ввімкнені послідовно
        clr  TMOD.2                  ;таймер працює від внутрішнього джерела
```

```
                                ;сигналів синхронізації
        clr  TMOD.3             ;робота таймера дозволяється якщо
                                ;встановлений керуючий біт TCON_TR0
        setb TCON_TR0           ;запуск таймера 0
        setb IE_ET0             ;дозвіл переривання по таймеру 0
;=======================================================================
        setb IE_EA              ;глобальний дозвіл препивань
        ret
;=======================================================================
timer0:

        MOV  TL0, 0x00                ;встановлення значення регістрів
таймеру
        MOV  TH0, 0x8F                ;встановлення значення регістрів
таймеру
        call print
        reti

print:
        mov R0, #0              ;номер символу в строці, що виводиться
        mov R1, #0x01           ;пін відповідної позиції
rel0:
        mov  dptr, #send        ;указник у початок строки
        mov  a, R0              ;виймання цифри
        movc a, @a+dptr         ;з пам'яті
        mov  dptr, #code        ;кодування для відображення
        movc a, @a+dptr         ;на дисплеї
        mov  P1, a              ;відправка у порт
        mov  a, R0              ;перевірка
        JZ   rel2               ;на
        subb a, #0              ;необхідність
        JNZ  rel1               ;крапки
        setb P1.7               ;для
        jmp  rel2               ;розділення
rel1:
        mov  a, R0              ;дня,
        subb a, #3              ;місяці
        JNZ  rel2               ;і
        setb P1.7               ;року
rel2:
        mov  a, R1              ;спалах
        cpl  A                  ;цифрової позиції
        mov  P2, a              ;на дисплеї
        call DELAY              ;затримка
        mov  P2, #0xFF                ;гасіння
        inc  R0                 ;наступна цифра
        mov  a, R1              ;зсув
        RL   a                  ;для показника позиції
        mov  R1,a               ;на дисплеї
        mov  a, R0              ;перевірка
        subb a, #8              ;на кінець
        JNZ rel0                ;строки
        ret                     ;і вихід
```

```
;=====================================================================
DELAY:
        MOV R2, #1              ;завантаження числа X1
COUNT1:  MOV R3, #1             ;завантаження числа X2
COUNT2:  MOV R4, #80            ;завантаження числа X3
COUNT3: DJNZ R4, COUNT3         ;декремент R4 та цикл, якщо не рівний 0
        DJNZ R3, COUNT2         ;декремент R3 та цикл, якщо не рівний 0
        DJNZ R2, COUNT1         ;декремент R2 та цикл, якщо не рівний 0
        ret                     ;повернення з підпрограмми затримки

;=====================================================================
code: db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F,
0x00
send: db 0x02, 0x01, 0x01, 0x02, 0x02, 0x00, 0x01, 0x02, 0x00
;=====================================================================
         END
```

4 ЛАБОРАТОРНА РОБОТА №4 “РОБОТА З LCD-ІНДИКАТОРАМИ НА БАЗІ КОНТРОЛЕРА HD44780”

Мета роботи – ознайомитися із загальними принципами організації системи виводу інформації в мікропроцесорних системах за допомогою сучасних знакосинтезуючих дисплеїв на базі контролеру HD44780.

4.1 Організація індикації у МПС на базі HD44780

Контролер HD44780 виробництва Hitachi є найбільш поширеним контролером управління алфавітно-цифровим модулем. Майже всі провідні виробники - Epson, Sanyo, Toshiba, Samsung, Philips випускають аналоги цього контролера або сумісні з ним по інтерфейсу і командній мові мікросхеми або РКІ на базі цих контролерів. Практично HD44780 є промисловим стандартом. Модулі з цим контролером застосовують у вимірювальних приладах, промисловому, технологічному і медичному устаткуванні, офісній техніці.

Контролер підтримує розмір символу 5x7 точок і 5x10 точок. HD44780 може управляти двома рядками по 40 символів.

Підключення до МК здійснюється так. Контролер з'єднується з РКІ через паралельну синхронну шину (8 або 4 ліній даних - вибирається програмно), через лінію вибору операцій (R/W), лінію вибору регістра (RS), лінію стробування і синхронізації (E) (рис. 4.1).

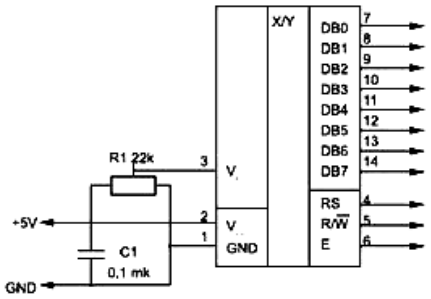


Рисунок 4.1 – Інтерфейс підключення контролера HD44780

Підстроювальний резистор R1 плавно змінює напругу живлення РКІ, що дозволяє виставляти необхідну контрастність індикатора при необхідному куті огляду. Модуль може під'єднуватися за допомогою 4- або 8-мирозрядної шини (рис. 4.2). При цьому обмін можна організувати або з системною шиною, або через порти вводу-виводу програмними засобами.

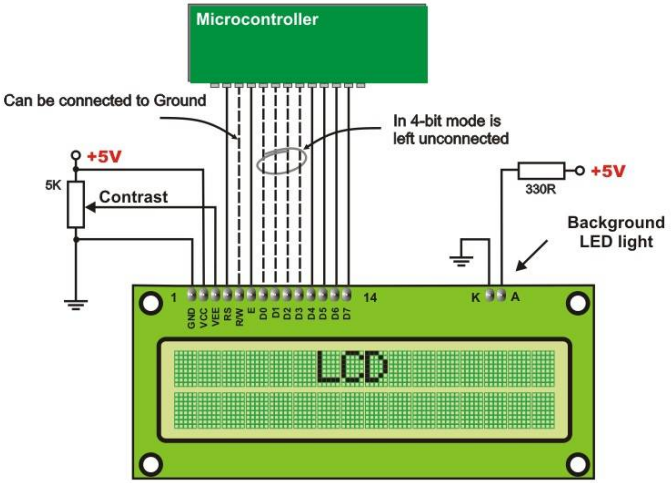


Рисунок 4.2 – Схема підключення РКІ до МК

На часовій діаграмі на рис. 4.3 показані стани керуючих сигналів і шини даних під час читання і запису.

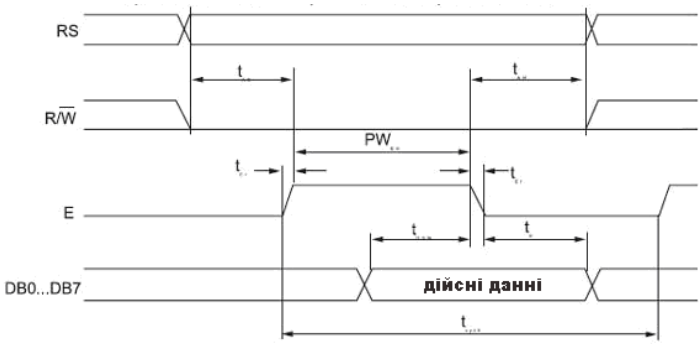


Рисунок 4.3 – Часова діаграма запису даних у РКІ

У вихідному стані E=0, R/W=0 значення сигналу RS довільне, шина даних DB0...DB7 перебуває в стані високого імпедансу. У проміжках між операціями обміну E і R/W також мають дорівнювати 0, у цей момент шина даних вільна і може використовуватися в мультиплексному режимі для інших цілей. У табл.4.1 наведено послідовність дій при виконанні операцій читання-запису. Час виконання кожного кроку не менше 250 нс.

Таблиця 4.1 – Послідовність дій при читання та запису

Операції запису для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Виведення значення байта даних на шину DB0...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Установка шини DB0...DB7 у стан HI
Операції читання для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування байта даних з шини DB0...DB7
5	Установка лінії E=0
6	Установка лінії R/W=0
Операції запису для 4-х розрядної шини	
1	Установка значення лінії RS
2	Виведення значення старшої тетради байта даних DB4...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Виведення значення молодшої тетради байта даних DB4...DB7
6	Установка лінії E=1
7	Установка лінії E =0
8	Установка шини DB4...DB7 в стан HI
Операції читання для 4-х розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування значення старшої тетради байта даних DB4...DB7
5	Установка лінії E=0

Продовження таблиці 4.1

6	Установка лінії E=1
7	Зчитування значення молодшої тетради байта даних DB4...DB7
8	Установка лінії E=0
9	Установка лінії R/W=0

Описані операції читання-запису байта є базовими для здійснення обміну даними з РКІ. Процес обміну по 4-х і 8-мі розрядною шиною розрізняється лише реалізацією цих операцій. Ці дві операції можуть бути реалізовані апаратно, коли модуль підключений до системної шини, або програмно, коли він взаємодіє з портами МК.

У таблиці 4.2 наведені значення часових інтервалів сигналів.

Таблиця 4.2 – Часові характеристики

Параметр	Позн.	Мін., нс	Макс., нс
Операція читання			
Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	80	-
Утримання даних	t_{DSH}	10	-
Операція запису			
Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	-	160
Утримання даних	t_{DHW}	5	-

4.1.1 Програмування і управління LCD на базі HD4470

На рис. 4.4 наведено основні елементи контролера HD4470, які безпосередньо взаємодіють з керуючою програмою - регістр даних

(DR), регістр команд (IR), відеопам'ять (DDRAM), ОЗП знакогенератора (CGRAM), лічильник адреси пам'яті (AC), прапор зайнятості контролера.

Управляється контролер через керуючий інтерфейс системи. Основні об'єкти взаємодії - регістри DR і IR. Вибір регістра, що адресується, здійснюється лінією RS, якщо RS=0 - адресується регістр команд (IR), якщо RS=1 - регістр даних (DR). Дані через регістр DR можуть розміщатися або зчитуватися у відеопам'ять (DDRAM), чи в ОЗП знакогенератора (CGRAM) за поточною адресою, на яку вказує лічильник адреси (AC). Інформація в регістрі IR інтерпретується пристроєм виконання команд як керуюча послідовність. Зчитування регістра IR повертає в 7-мі молодших розрядах поточне значення лічильника AC, а в старшому розряді прапор зайнятості (BF). Відеопам'ять має загальний об'єм 80 байтів і призначена для зберігання коду символів.

У таблиці 4.3 показаний набір різних прапорів для HD44780. Прапори визначають режими роботи різних елементів контролера. У таблиці 4.4 наведені значення управляючих прапорів, безпосередньо після подачі на РКІ-модуль напруги живлення. Перевизначення значень прапорів виробляється спеціальними командами, записуваними в регістр IR, при цьому комбінації старших бітів визначають групу прапорів або команду, а молодші містять власне прапори.

Таблиця 4.3 – Управляючі прапори контролера HD44780

I/D	Режим зсуву лічильника адреси AC, 0 - зменшення, 1 - збільшення
S	Прапор режиму зсуву вмісту екрану. 0 – зсуву немає, 1 - після запису в DDRAM чергового коду екран зсувається в напрямі, який визначається прапором I/D: 0 - вправо, 1 - вліво. При зсуві не змінюється вміст DDRAM. змінюються лише внутрішні покажчики розташування видимого початку рядка в DDRAM
S/C	Прапор-команда, яка виробляє разом з прапором R/L операцію зсуву вмісту екрану (так само, як і у попередньому випадку, без змін в DDRAM) або курсору. Визначає об'єкт зсуву: 0 - зсувається курсор, 1 - зсувається екран
R/L	Прапор-команда, яка виробляє разом з прапором S/C операцію зсуву екрану або курсору. Уточнює напрям зсуву: 0 - вліво, 1 - вправо
D/L	Прапор, що визначає ширину шини даних: 0 - 4 розряди, 1 - 8 розрядів

Продовження таблиці 4.3

N	Режим розгортки зображення на ЖКИ: 0 - один рядок, 1 - два рядки
F	Розмір матриці символів: 0 - 5 x 8 точок, 1 - 5 x 10 точок
D	Наявність зображення: 0 - вимкнено, 1 - включено
C	Курсор у вигляді підкреслення: 0 - вимкнений, 1 - включений
B	Курсор у вигляді мерехтливого знакомісця: 0 - вимкнений, 1 - включений

Таблиця 4.4 – Значення прапорів після подачі живлення

I/D=1	Режим збільшення лічильника на 1
S=0	Без зсуву зображення
D/L=1	8-мирозрядна шина даних
N=0	Режим розгортки одного рядка
F=0	Символи з матрицею 5 x 8 точок
D=0	Відображення вимкнене
C = 0	Курсор у вигляді підкреслення вимкнений
B = 0	Курсор у вигляді мерехтливого знакомісця вимкнений

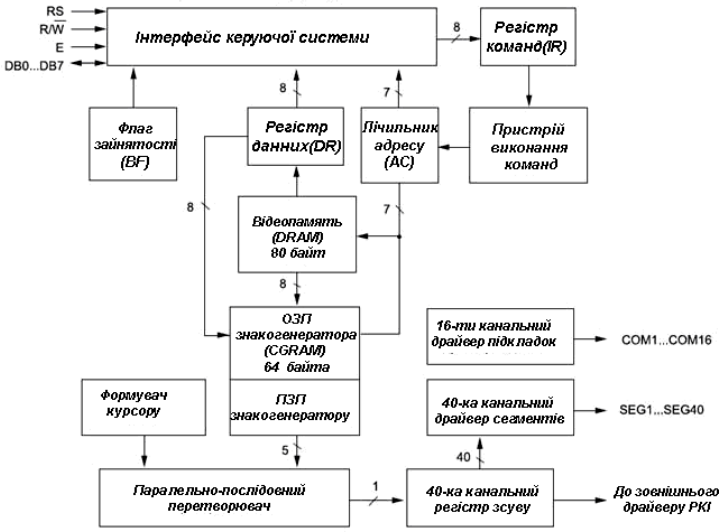


Рисунок 4.4 –Спрощена структурна схема контролера HD4470

Список комбінацій бітів регістра IR і виконувані ними управляючі команди наведені в табл.4.5. Оскільки на момент включення РКІ-модуль нічого не відображує (прапор D=0), то для того, щоб вивести будь-який текст, необхідно, як мінімум, включити відображення, встановивши прапор D=1. Наприклад, широко поширена послідовність для ініціалізації модуля: $ОС, 6 (знак $ перед числом вказує на шістнадцяткове представлення) встановлює режим відображення 2-х рядків з матрицею 5 х 8 точок і робота з 8-мирозрядною шиною даних; $ОС включає відображення на екрані модуля без відображення курсорів; 6 встановлює режим автоматичного переміщення курсору зліва-направо після виведення кожного символу.

Таблиця 4.5 - Комбінації бітів регістра IR

Призначення	D7	D6	D5	D4	D3	D2	D1	D0
Очищення екрану, АС = 0, адресація АС на DDRAM	0	0	0	0	0	0	0	1
АС=0, адресація на DDRAM, скинуті зсуви, початок рядка адресується на початку DDRAM	0	0	0	0	0	0	1	-
Вибирається напрям зсуву курсору чи екрану	0	0	0	0	0	1	I/D	S
Вибирається режим відображення	0	0	0	0	1	D	C	B
Команда зсуву курсору/екрану	0	0	0	1	S/C	R/L	-	-
Визначення параметрів розгортки і ширини шини даних	0	0	1	DL	N	F	-	-
Присвоєння лічильнику АС адреси в області CGRAM	0	1	AG	AG	AG	AG	AG	AG
Присвоєння лічильнику АС адреси в області DDRAM	1	AD	AD	AD	AD	AD	AD	AD

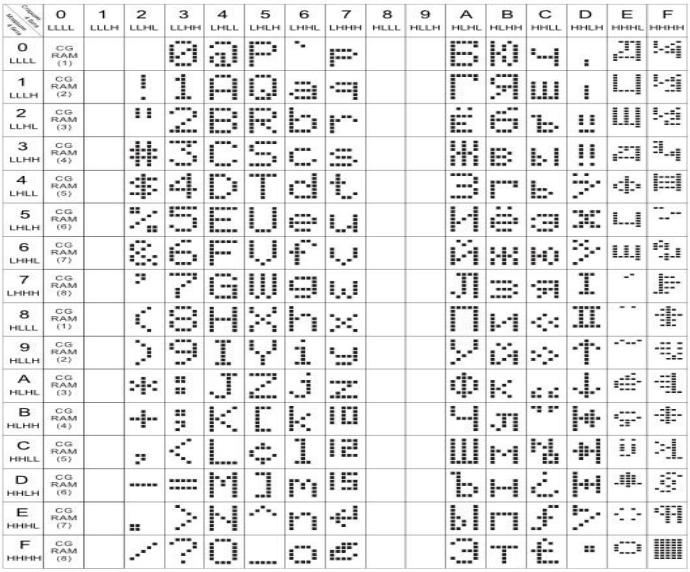
Контролер HD44780 підтримує як операції запису, так і операції читання. Читання регістра DR призводить до завантаження вмісту DDRAM або CGRAM залежно від поточного режиму, при цьому курсор зміщується на одну позицію, як і при записі. Читання регістра IR повертає 8 значущих розрядів, причому в 7-ми молодших міститься поточне значення лічильника АС (7 розрядів, якщо адресується DDRAM, і 6 - якщо CGRAM), а в старшому - прапор зайнятості BF. Цей прапор має значення 1 коли контролер зайнятий і 0 - коли

вільний. Необхідно враховувати, що більшість операцій, які виконуються контролером, займають значний час, близько 40 мкс, а час виконання деяких доходить до одиниць мілісекунд, тому цикл чекання зняття прапора BF має бути обов'язково присутнім в програмах драйвера РКІ-модуля і передувати здійсненню будь-якої операції (природно, окрім операції перевірки прапора BF).

Після здійснення операції запису або читання DDRAM і появи після неї ознаки готовності (BF=0), прочитане в цьому ж циклі (разом з прапором BF) значення AC швидше за все не буде достовірним, оскільки між появою ознаки готовності і обчисленням контролером нового значення АСА існує деякий часовий інтервал близько 4 мкс при тактовій частоті контролера 270 кГц. Тому якщо необхідно набути дійсного значення АСА, потрібно зробити повторну операцію прочитання IR після не менше ніж 4 мкс (якщо контролер працює на частоті 270 кГц час чекання необхідно пропорційно збільшити).

Виведення на екран символу здійснюється шляхом запису його коду (табл. 4.6) в регістр DR. Наприклад, код символу *0* за таблицею є *$30*. При цьому символ розміщується в DDRAM за поточною адресою, що вказується в AC, а значення AC збільшується чи зменшується на 1.

Таблиця 4.6 – Коди символів для відображення на екрані



Виробник контролера рекомендує виконувати наступну послідовність дій для ініціалізації. Витримати паузу не менше 15 мс між встановленням робочої напруги живлення (більше 4,5 В) і виконанням будь-яких операцій з контролером. Першою операцією виконати команду, яка вибирає розрядність шини (це має бути команда незалежно від того, якої розрядності інтерфейс ви збираєтеся використовувати надалі), причому перед виконанням цієї операції не перевіряти значення прапора BF. Далі знову витримати паузу не менше 4,1 мс і повторити команду вибору розрядності шини, причому перед поданням команди знову не перевіряти прапор BF. Наступний крок - знов витримати паузу, цього разу 100 мкс, і втретє повторити команду встановлення розрядності шини, знов без перевірки BF. Ці три операції призначені для ініціалізації і покликані вивести контролер у вихідний режим роботи (тобто перевести в режим роботи з 8-мирозрядною шиною) з будь-якого стану. Слідом за ними нормальним порядком (без витримки пауз, але з перевіркою прапора BF) виконується ініціалізація режимів роботи з видачею послідовності для ініціалізації, аналогічній вказаній раніше (що містить у тому числі команду вибору необхідної розрядності шини).

Слід пам'ятати, що при виборі режиму роботи з 4-х розрядною шиною, тобто видачі команд, це звичайно відбувається з 8-ми розрядного режиму, який встановлюється автоматично після подачі напруги живлення, а отже ви не зможете адекватно оголосити необхідне значення прапорів N і F, розташованих в молодшій тетраді команди установки розрядності шини. Тому команду необхідно повторити у вже сталому 4-х розрядному режимі послідовною передачею двох тетрад.

4.2 Завдання до лабораторної роботи

1 Ознайомитися з технічною документацією на запропоновану викладачем модель LCD – дисплею.

2 Нарисувати блок-схему алгоритму ініціалізації LCD - дисплею по 8-бітному інтерфейсу.

3 Нарисувати блок-схему алгоритму ініціалізації LCD - дисплею по 4-бітному інтерфейсу.

4 Нарисувати блок схему і написати програму для виводу на LCD-дисплей (HD44780) свого прізвища при підключенні його до МК по 8-бітному інтерфейсу.

5 Нарисувати блок схему і написати програму для виводу на LCD-дисплей (HD44780) свого прізвища при підключенні його до МК по 4-бітному інтерфейсу.

6 Здійснити симуляцію роботи МПС. За наявності помилок знайти і виправити помилку алгоритму чи самої програми.

7 Оформити звіт, занести до нього результати моделювання. Зробити висновки.

4.3 Контрольні запитання

1 Принцип побудови РКІ-модуля з контролером HD44780.
2 Алгоритм роботи по 4-х провідному інтерфейсу.
3 Алгоритм роботи по 8-ми провідному інтерфейсу.
4 Переваги і недоліки модулів з контролером HD44780.
5 Схема підключення РКІ-модуля до МК.

4.4 Приклад виконання завдання

Створити проект як в прикладі 1.6 та використовуючи інструменти програми у вкладці **Shematic Capture** зібрати схему як на рис.:

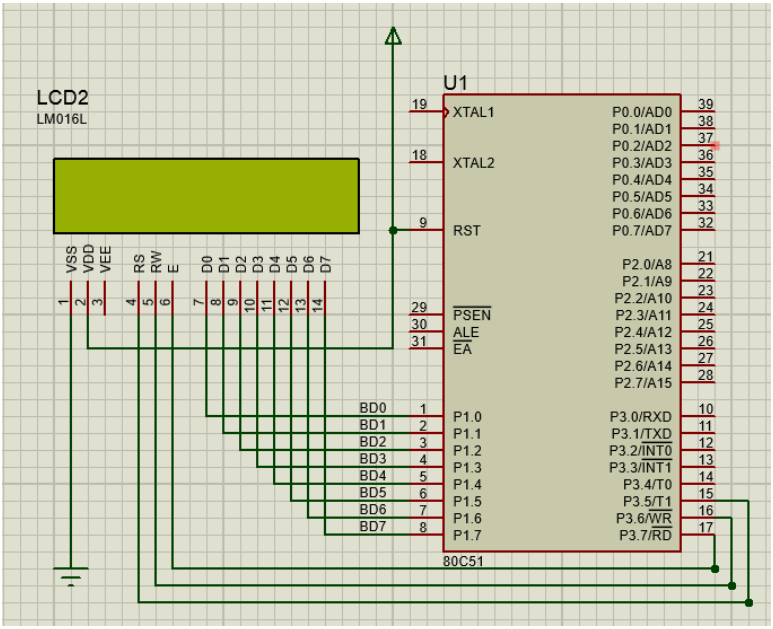


Рисунок 4.5 –Схема підключення контролера HD4470 по 8-му бітному інтерфейсу

Лістинг прикладу програми для 1 завдання:

```
;===================================================================
; DEFINITIONS
;===================================================================

#include "io8051.h"                ; Include register definition file

;===================================================================
; VARIABLES
;===================================================================

DBP = P1
DB0 = P1.0
DB1 = P1.1
DB2 = P1.2
DB3 = P1.3
DB4 = P1.4
DB5 = P1.5
DB6 = P1.6
DB7 = P1.7
RS = P3.5
RW = P3.6
```

```
EN = P3.7

;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================
      ; Reset Vector
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start
;=====================================================================
; CODE SEGMENT
;=====================================================================
      RSEG CODE_SEG:CODE         ; Switch to this code segment.
                                 ; Register bank 0 by default
Start:
      lcall delay_20ms
      lcall init_lcd               ;виклик підпрограми ініціалізації
LCD

      MOV ACC,#0x80                   ;курсор в початок першої строки
      lcall send_com               ;відправлення команди

      MOV ACC,#0x48                ;H
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x65                ;e
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x6C                ;l
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x6C                ;l
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x6F                ;o
      lcall send_dat               ;відправлення данних

      MOV ACC,#0xC0                   ;курсор в початок другої строки
      lcall send_com               ;відправлення команди

      MOV ACC,#0x77                ;w
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x6F                ;o
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x72                ;r
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x6C                ;l
      lcall send_dat               ;відправлення данних
      MOV ACC,#0x64                ;d
      lcall send_dat               ;відправлення данних

Loop:
      ljmp Loop                        ;нескінченний цикл

init_lcd:
      MOV ACC,#0x38                ;8 розрядів;2 строки;символи 5*8
точок
      lcall send_com               ;відправлення команди
```

```
      MOV ACC,#0x08                ;дисплей вимкнений, курсор
вимкнений
      lcall send_com               ;відправлення команди
      MOV ACC,#0x01                ;очистка дисплея, курсор в початок
екрану
      lcall send_com               ;відправлення команди
      MOV ACC,#0x06                ;Зсув показника пам'яті вправо,
зсув екрана
                                  ;заборонений
      lcall send_com               ;відправлення команди
      MOV ACC,#0x0C                ;ввімкнення дисплею
      lcall send_com               ;відправлення команди
      ret

send_dat:
      setb RS                       ;RS=1 (данні)
      lcall send                    ;підпрограмма відправки
      ret

send_com:
      clr  RS                       ;RS=0 (команда)
      lcall send                    ;підпрограмма відправки
      ret

send:
      MOV  DBP, ACC                    ;відпровка данних у порт, з
акумулятору
      clr  RW                       ;RW=0 (запуск)
      setb EN                        ;початок тробуючого імпульсу
      lcall delay_1600mks           ;затримка
      clr  EN                        ;кінець тробуючого імпульсу
      clr  RS                        ;очистка RS
      ret

delay:MOV R2,100
count:DJNZ R2, count
      ret

;=====================================================================
delay_20ms:
     MOV R3, #50                   ;завантаження числа X2
COUNT11: MOV R4, #200               ;завантаження числа X3
COUNT12: DJNZ R4, COUNT12           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT11               ;декремент R3 та цикл, якщо не
рівний 0
     ret                           ;повернення з підпрограмми затримки
;=====================================================================
delay_1600mks:
     MOV R3, #4                      ;завантаження числа X2
COUNT21: MOV R4, #200               ;завантаження числа X3
COUNT22: DJNZ R4, COUNT22           ;декремент R4 та цикл, якщо не
рівний 0
```

```
     DJNZ R3, COUNT21              ;декремент R3 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
delay_40mks:
     MOV R4, #17                  ;завантаження числа X3
COUNT32: DJNZ R4, COUNT32          ;декремент R4 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
mesg1:  db  "ZNTU kaf. MiNE ",0
mesg2:  db  "Laba MPT ",0
;=====================================================================
          END
```

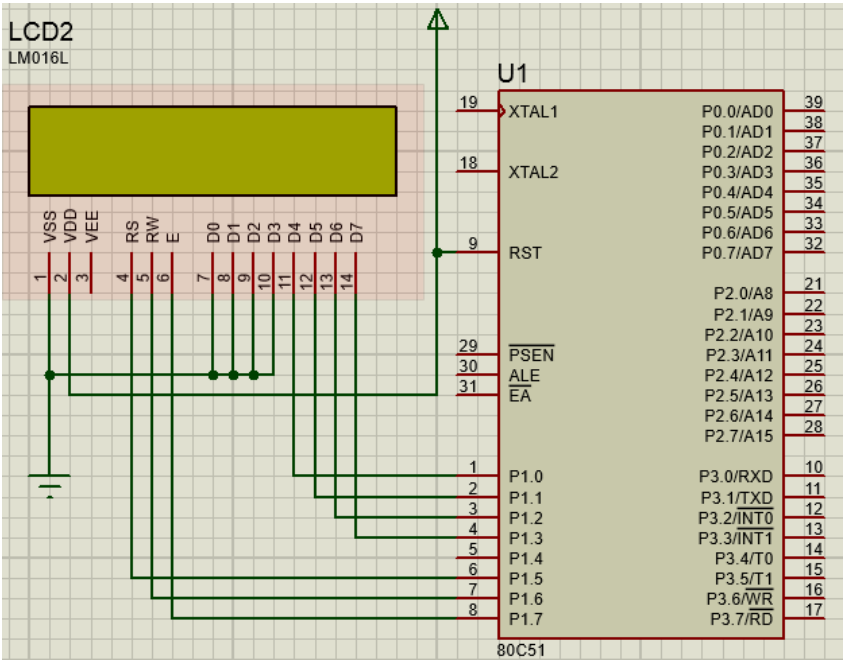


Рисунок 4.6 –Схема підключення контролера HD4470 по 4-му бітному інтерфейсу

Лістинг прикладу програми для 2 завдання:

```
;=====================================================================
; DEFINITIONS
;=====================================================================
#include "io8051.h"                ; Include register definition file
;=====================================================================
```

```
; VARIABLES
;=====================================================================
DB0 = P1.0
DB1 = P1.1
DB2 = P1.2
DB3 = P1.3
RS  = P1.5
RW  = P1.6
EN  = P1.7
;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================
      ; Reset Vector
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start
      RSEG CODE_SEG:CODE        ; Switch to this code segment.
Start:
      lcall delay_20ms
      lcall init_lcd              ;виклик підпрограми ініціалізації
LCD
      MOV ACC,#0x80                 ;курсор в початок першої строки
      lcall send_com              ;відправлення команди
      mov dptr, #mesg1              ;запис у регістр-показник адреси
строки
      lcall wr_string              ;виклик підпрограми виводу строки


      MOV ACC,#0xC0                 ;курсор в початок другої строки
      lcall send_com              ;відправлення команди
      mov dptr, #mesg2              ;запис у регістр-показник адреси
строки
      lcall wr_string              ;виклик підпрограми виводу строки
Loop:
      ljmp Loop                        ;нескінченний цикл

wr_string:
      clr  a                        ;очистка акумулятора
      movc a, @a+dptr             ;запис в акумулятор байту по
адресі в DPTR
      inc dptr                      ;збільшення DPTR на одиницю
      jz  str_end                 ;перевірка на кінецб стороки, якщо
так - вихід
      lcall send_dat              ;якщо НЕ кінецб строки - відправка
данних у LCD
      jnz wr_string                 ;якщо НЕ кінецб строки - перехід
до мітки wr_string
str_end:
      ret

init_lcd:
      MOV  P1,#0x03               ;8 розрядів
      clr  RS
      clr  RW                     ;RW=0 (запуск)
```

```
      lcall strob                   ;стробуючий імпульс
      MOV P1,#0x02                 ;4 розряди;2 строки;символи 5*8
точок
      clr  RS
      clr  RW                      ;RW=0 (запуск)
      lcall strob                   ;стробуючий імпульс
      MOV ACC,#0x28                ;4 розряди;2 строки;символи 5*8
точок
      lcall send_com               ;відправлення команди
      MOV ACC,#0x08                ;дисплей вимкнений, курсор
вимкнений
      lcall send_com               ;відправлення команди
      MOV ACC,#0x01                ;очистка дисплея, курсор в початок
екрану
      lcall send_com               ;відправлення команди
      MOV ACC,#0x06                ;Зсув показника пам'яті вправо,
зсув екрана
                                  ;заборонений
      lcall send_com               ;відправлення команди
      MOV ACC,#0x0C                ;ввімкнення дисплею
      lcall send_com               ;відправлення команди
      ret
send_dat:
      setb RS                       ;RS=1 (данні)
      lcall send                    ;підпрограмма відправки
      ret
send_com:
      clr  RS                       ;RS=0 (команда)
      lcall send                    ;підпрограмма відправки
      ret
send:
      JB  ACC.4, lab11
      clr  DB0
lab11:JNB ACC.4, lab12
      setb DB0
lab12:JB  ACC.5, lab21
      clr  DB1
lab21:JNB ACC.5, lab22
      setb DB1
lab22:JB  ACC.6, lab31
      clr  DB2
lab31:JNB ACC.6, lab32
      setb DB2
lab32:JB  ACC.7, lab41
      clr  DB3
lab41:JNB ACC.7, lab42
      setb DB3
lab42:
      clr  RW                       ;RW=0 (запуск)
      lcall strob                      ;стробуючий імпульс
      JB  ACC.0, lab51
      clr  DB0
lab51:JNB ACC.0, lab52
```

```
      setb DB0
lab52:JB  ACC.1, lab61
      clr  DB1
lab61:JNB ACC.1, lab62
      setb DB1
lab62:JB  ACC.2, lab71
      clr  DB2
lab71:JNB ACC.2, lab72
      setb DB2
lab72:JB  ACC.3, lab81
      clr  DB3
lab81:JNB ACC.3, lab82
      setb DB3
lab82:clr  RW                        ;RW=0 (запуск)
      lcall strob                       ;стробуючий імпульс
      clr  RS                          ;очистка RS
      ret
strob:
      setb EN                           ;початок стробуючого імпульсу
      lcall delay_1600mks              ;затримка
      clr  EN                           ;кінець стробуючого імпульсу
      ret
delay:MOV R2,100
count:DJNZ R2, count
      ret
delay_20ms:
     MOV R3, #50                  ;завантаження числа X2
COUNT11: MOV R4, #200              ;завантаження числа X3
COUNT12: DJNZ R4, COUNT12          ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT11              ;декремент R3 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
delay_1600mks:
     MOV R3, #4                      ;завантаження числа X2
COUNT21: MOV R4, #200              ;завантаження числа X3
COUNT22: DJNZ R4, COUNT22          ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT21              ;декремент R3 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
delay_40mks:
     MOV R4, #17                  ;завантаження числа X3
COUNT32: DJNZ R4, COUNT32          ;декремент R4 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
mesg1:  db  "ZNTU kaf. MiNE ",0
mesg2:  db  "Laba MPT ",0
;=====================================================================
          END
```

5 ЛАБОРАТОРНА РОБОТА №5 “СИСТЕМИ ВВОДУ ІНФОРМАЦІЇ У МПС”

Мета роботи – ознайомитися із загальними принципами організації системи клавіатурного вводу інформації в мікропроцесорних системах за допомогою готових модулів.

5.1 Принципи побудови системи вводу інформації у МПС

Найпростішим засобом вводу інформації в мікропроцесорну систему є ручне введення інформації з клавіатури. У найпростішому випадку клавіатура може складатися лише з однієї кнопки (рис. 5.1).

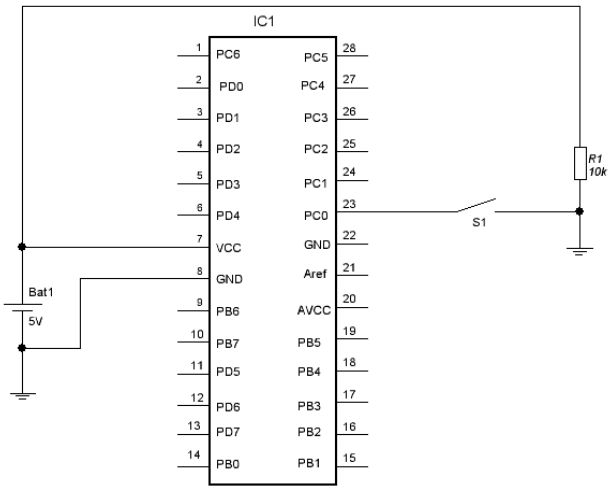


Рисунок 5.1 – Схема підключення кнопки до МПС

У випадку застосування більшої кількості кнопок, які утворюють клавіатуру, розробнику може просто не вистачити вільних портів. З урахуванням того, що контролер зазвичай ще виконує функції виводу якоїсь інформації чи керування певною кількістю виконуючих пристроїв, це питання стає проблемою. Рішенням цієї проблеми є складання кнопок у матриці, рис. 5.2.

Роботу мікроконтролера при такому підключенні умовно можна представити так. Порт PB (виводи 0-1-2) налаштовані на вивід, у вихідному стані на виході «1». Виводи порту PD налаштовані на ввід. Через струмообмежуючі резистори на входах портів присутні логічні одиниці. Читання порту PD дає значення **$ff**. При роботі МК сканує клавіатуру: по черзі подає на виводи порту PB0, PB1, PB2 логічні нулі і одночасно здійснює читання стану порту PD. У випадку натискування однієї з клавіш результат читання порту PD буде відрізнятися від значення **$ff**.

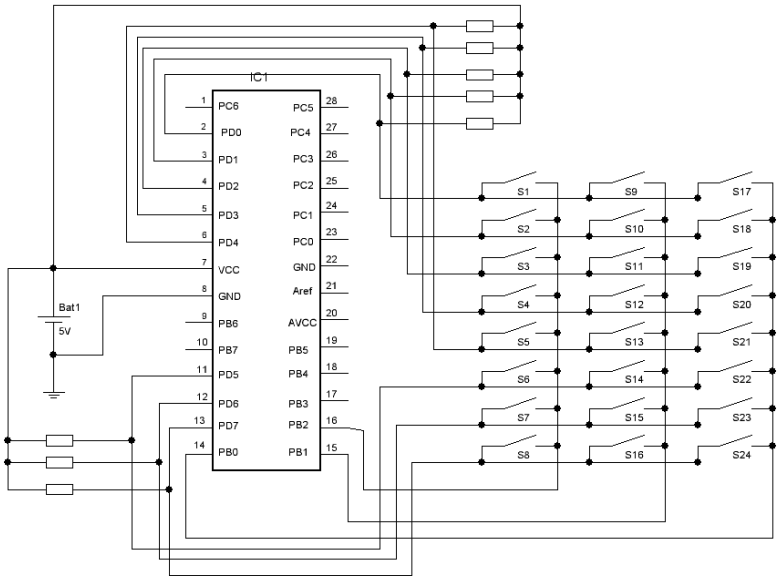


Рисунок 5.2 – Схема підключення матричної клавіатури 3x8 (24 кнопки)

5.2 Завдання до лабораторної роботи

1 Ознайомитися з принципом дії та способами опитування матричної клавіатури.

2 Нарисувати блок-схему алгоритму опитування матричної клавіатури.

3 Нарисувати блок схему і написати програму для виводу на LCD-дисплей свого прізвища при підключенні його до МК по 4-

бітному інтерфейсу у першу строку та виводу символів з клавіатури у другу строку.

4 Здійснити симуляцію роботи МПС. За наявності помилок знайти і виправити помилку алгоритму чи самої програми.

5 Оформити звіт, занести до нього результати моделювання. Зробити висновки.

5.3 Контрольні запитання

1. Принцип побудови матричної клавіатури та алгоритм її опитування;
2. Брязкіт контакту та спосіб його усунення;
3. Небезпека одночасного натискання двох і більше клавіш одного рядка, спосіб його усунення.

5.4 Приклад виконання завдання

1 Створити проект як в прикладі 1.6 та використовуючи інструменти програми у вкладці **Shematic Capture** зібрати схему як на рис.:

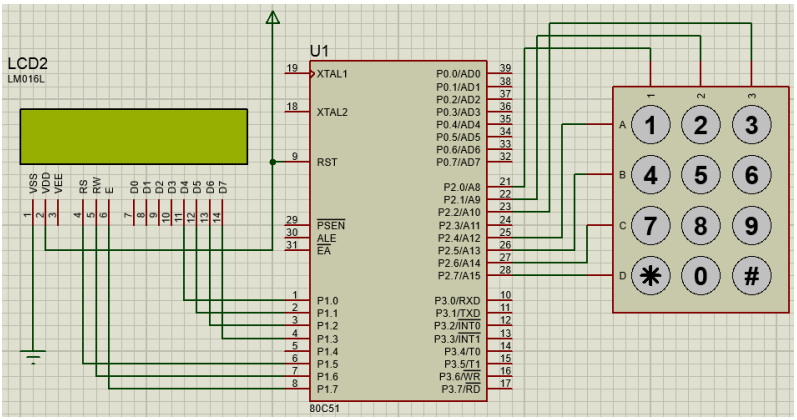


Рисунок 5.3 –Схема підключення матричної клавіатури

Лістинг прикладу програми для завдання:

```
;===================================================================
; DEFINITIONS
;===================================================================
```

```
#include "io8051.h"                ; Include register definition file

;=====================================================================
; VARIABLES
;=====================================================================

DB0 = P1.0
DB1 = P1.1
DB2 = P1.2
DB3 = P1.3

RS  = P1.5
RW  = P1.6
EN  = P1.7

K1 = P2.0
K2 = P2.1
K3 = P2.2
KA = P2.4
KB = P2.5
KC = P2.6
KD = P2.7
;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

      ; Reset Vector
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start

;=====================================================================
; CODE SEGMENT
;=====================================================================

      RSEG CODE_SEG:CODE           ; Switch to this code segment.
                                 ; Register bank 0 by default

Start:
      lcall delay_20ms
      lcall init_lcd              ;виклик підпрограми ініціалізації
LCD

      MOV ACC,#0x80                 ;курсор в початок першої строки
      lcall send_com              ;відправлення команди
      mov dptr, #mesg1              ;запис у регістр-показник адреси
строки
      lcall wr_string              ;виклик підпрограми виводу строки

      MOV ACC,#0xC0                 ;курсор в початок другої строки
      lcall send_com              ;відправлення команди

      MOV P2,#0xFF
```

```
Loop:
;опитування ряду A
;опитування ряду A кнопка 1
      clr   KA
      JB    K1, Klb11
      MOV   ACC, #0x31
      lcall send_dat
      lcall delay_100ms
Klb12:JNB   K1, Klb12
;опитування ряду A кнопка 2
Klb11:JB    K2, Klb21
      MOV   ACC, #0x32
      lcall send_dat
      lcall delay_100ms
Klb22:JNB   K2, Klb22
;опитування ряду A кнопка 3
Klb21:JB    K3, Klb31
      MOV   ACC, #0x33
      lcall send_dat
      lcall delay_100ms
Klb32:JNB   K3, Klb32
;опитування ряду B
;опитування ряду B кнопка 4
Klb31:setb  KA
      clr   KB
      JB    K1,  Klb41
      MOV   ACC, #0x34
      lcall send_dat
      lcall delay_100ms
Klb42:JNB   K1, Klb42
;опитування ряду B кнопка 5
Klb41:JB    K2,  Klb51
      MOV   ACC, #0x35
      lcall send_dat
      lcall delay_100ms
Klb52:JNB   K2, Klb52
;опитування ряду B кнопка 6
Klb51:JB    K3, Klb61
      MOV   ACC, #0x36
      lcall send_dat
      lcall delay_100ms
Klb62:JNB   K3, Klb62
;опитування ряду C
;опитування ряду C кнопка 7
Klb61:setb  KB
      clr   KC
      JB    K1, Klb71
      MOV   ACC, #0x37
      lcall send_dat
      lcall delay_100ms
Klb72:JNB   K1, Klb72
;опитування ряду C кнопка 8
Klb71:JB    K2, Klb81
```

```
      MOV   ACC, #0x38
      lcall send_dat
      lcall delay_100ms
Klb82:JNB   K2, Klb82
;опитування ряду C кнопка 9
Klb81:JB    K3, Klb91
      MOV   ACC, #0x39
      lcall send_dat
      lcall delay_100ms
Klb92:JNB   K3, Klb92
;опитування ряду C
;опитування ряду C кнопка *
Klb91:setb  KC
      clr   KD
      JB    K1, KlbA1
      MOV   ACC, #0x2A
      lcall send_dat
      lcall delay_100ms
KlbA2:JNB   K1, KlbA2
;опитування ряду C кнопка 0
KlbA1:JB    K2, KlbB1
      MOV   ACC, #0x30
      lcall send_dat
      lcall delay_100ms
KlbB2:JNB   K2, KlbB2
;опитування ряду C кнопка #
KlbB1:JB    K3, KlbC1
      MOV   ACC, #0x23
      lcall send_dat
      lcall delay_100ms
KlbC2:JNB   K3, KlbC2
KlbC1:setb  KD
      ljmp  Loop                    ;нескінченний цикл

wr_string:
      clr  a                         ;очистка акумулятора
      movc a, @a+dptr               ;запис в акумулятор байту по
адресі в DPTR
      inc dptr                       ;збільшення DPTR на одиницю
      jz  str_end                   ;перевірка на кінецб стороки, якщо
так - вихід
      lcall send_dat                ;якщо НЕ кінецб строки - відправка
данних у LCD
      jnz wr_string                    ;якщо НЕ кінецб строки - перехід
до мітки wr_string
str_end:
      ret

init_lcd:
      MOV P1,#0x03                  ;8 розрядів
      clr  RS
      clr  RW                       ;RW=0 (запуск)
      lcall strob                      ;стробуючий імпульс
```

```
      MOV P1,#0x02                ;4 розряди;2 строки;символи 5*8
точок
      clr  RS
      clr  RW                     ;RW=0 (запуск)
      lcall strob                    ;стробуючий імпульс
      MOV ACC,#0x28               ;4 розряди;2 строки;символи 5*8
точок
      lcall send_com              ;відправлення команди
      MOV ACC,#0x08               ;дисплей вимкнений, курсор
вимкнений
      lcall send_com              ;відправлення команди
      MOV ACC,#0x01               ;очистка дисплея, курсор в початок
екрану
      lcall send_com              ;відправлення команди
      MOV ACC,#0x06               ;Зсув показника пам'яті вправо,
зсув екрана
                                 ;заборонений
      lcall send_com              ;відправлення команди
      MOV ACC,#0x0C               ;ввімкнення дисплею
      lcall send_com              ;відправлення команди
      ret

send_dat:
      setb RS                     ;RS=1 (данні)
      lcall send                  ;підпрограмма відправки
      ret

send_com:
      clr  RS                      ;RS=0 (команда)
      lcall send                   ;підпрограмма відправки
      ret
send:
      JB  ACC.4, lab11
      clr  DB0
lab11:JNB ACC.4, lab12
      setb DB0
lab12:JB  ACC.5, lab21
      clr  DB1
lab21:JNB ACC.5, lab22
      setb DB1
lab22:JB  ACC.6, lab31
      clr  DB2
lab31:JNB ACC.6, lab32
      setb DB2
lab32:JB  ACC.7, lab41
      clr  DB3
lab41:JNB ACC.7, lab42
      setb DB3
lab42:
      clr  RW                      ;RW=0 (запуск)
      lcall strob                     ;стробуючий імпульс
      JB  ACC.0, lab51
      clr  DB0
```

```
lab51:JNB ACC.0, lab52
      setb DB0
lab52:JB  ACC.1, lab61
      clr  DB1
lab61:JNB ACC.1, lab62
      setb DB1
lab62:JB  ACC.2, lab71
      clr  DB2
lab71:JNB ACC.2, lab72
      setb DB2
lab72:JB  ACC.3, lab81
      clr  DB3
lab81:JNB ACC.3, lab82
      setb DB3
lab82:clr  RW                        ;RW=0 (запуск)
      lcall strob                       ;стробуючий імпульс
      clr  RS                         ;очистка RS
      ret

strob:
      setb EN                         ;початок стробуючого імпульсу
      lcall delay_1600mks            ;затримка
      clr  EN                         ;кінець стробуючого імпульсу
      ret
delay:MOV R2,100
count:DJNZ R2, count
      ret

;=====================================================================
delay_20ms:
     MOV R3, #50                   ;завантаження числа X2
COUNT11: MOV R4, #200               ;завантаження числа X3
COUNT12: DJNZ R4, COUNT12           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT11               ;декремент R3 та цикл, якщо не
рівний 0
     ret                           ;повернення з підпрограмми затримки
;=====================================================================
delay_100ms:
     MOV R3, #255                   ;завантаження числа X2
COUNT31: MOV R4, #255               ;завантаження числа X3
COUNT32: DJNZ R4, COUNT32           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT31               ;декремент R3 та цикл, якщо не
рівний 0
     ret                           ;повернення з підпрограмми затримки
;=====================================================================
delay_1600mks:
     MOV R3, #4                      ;завантаження числа X2
COUNT21: MOV R4, #200               ;завантаження числа X3
COUNT22: DJNZ R4, COUNT22           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT21               ;декремент R3 та цикл, якщо не
```

```
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
delay_40mks:
     MOV R4, #17                  ;завантаження числа X3
COUNT42: DJNZ R4, COUNT42          ;декремент R4 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
mesg1:  db  "Opros klaviatury",0
mesg2:  db  "Laba MPT ",0
;=====================================================================
          END
```

6 ЛАБОРАТОРНА РОБОТА №6 “ОБРОБКА ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ МПС”

Мета роботи – ознайомитися із загальними принципами організації системи клавіатурного вводу інформації в мікропроцесорних системах за допомогою готових модулів.

6.1 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 двобайтне. 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 однорозрядного двійковими сигналами, велике число операцій введення/виводу і розгалужень програми. Найбільш істотна особливість системи команд розглянутих мікро ЕОМ це можливість адресації окремих біт в резидентну пам'ять 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. 1.1 і 1.2 відповідно [1, 3, 5].

6.1.1 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. 6.1 показані 13 типів команд MCS51.

Таблиця 6.1 – Типи команд MCS51

Тип команди	Первый байт D7...D0	Второй байт D7...D0	Третий 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

6.1.2 Позначення, використовувані 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 прямоадресного байта;
- ads – адреса прямоадресного байта-джерела;
- add – адреса прямоадресного байта-одержувача;

- 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 прямоадресного біта;
- /Bit – інверсія прямо адресного біта;
- A – акумулятор;
- PC – лічильник команд;
- DPTR – регістр покажчик даних;
- () – вміст комірки пам'яті або регістра.

6.1.3 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-бітова адреса (bit). Непряма адресація біт неможлива. Карти адрес окремих бітів представлені в таблиці 1.1 і 1.2. 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.

6.1.4 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.

6.1.5 Прапори результату

Слово стану програми PSW включає в себе чотири прапори:

- C - перенос,
- AC - допоміжне перенесення (напівперенос),
- 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 поділу на нуль встановлюється. При множенні прапор OV встановлюється, якщо результат більше 255.

У таблиці 6.2 перераховуються команди, при виконанні яких модифікуються прапори результату. У таблиці відсутня 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 6.2 – 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		

6.1.6 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.6.2). 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.

6.2 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 трибайтовий, 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 трибітовое поле, що визначає номер регістра. Вибір одного з чотирьох регістрових 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.

6.3 Команди передачі даних

Ця група представлена 28 командами, їх короткий опис наведено в додатку 1, 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 відповідного прямого 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).

6.3.1 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. 20. Передачі даних можуть виконуватися без участі акумулятора [1].

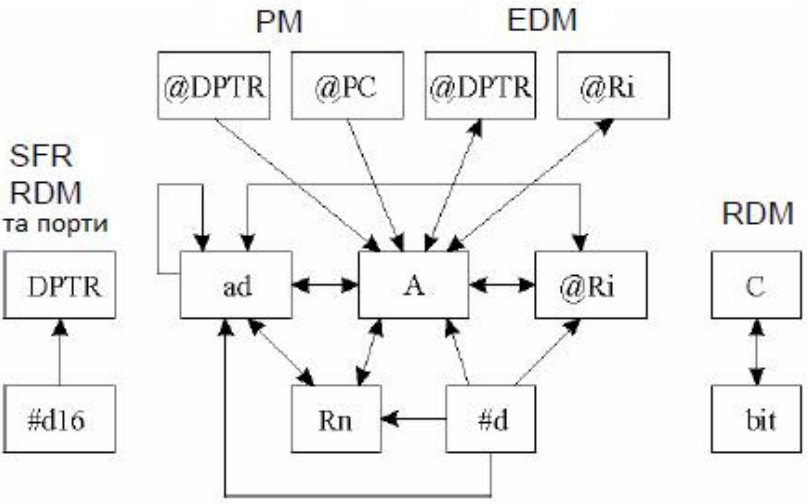


Рисунок 6.1 – Граф шляхів передачі данних

6.3.2 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 (прямий адресу). При прямої адресації звернення до акумулятора проводиться як до одного з регістрів спеціальних функцій, і його адреса вказується в другому байті команди. Використання неявній адресації акумулятора краще, але не завжди можливо, наприклад, при зверненні до окремих біт акумулятора.

6.3.3 Звернення до зовнішньої пам'яті даних

При використанні команд MOVX @Ri забезпечується доступ до 256 байт 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. 6.1. Граф шляхів передачі даних

6.4 Арифметичні операції

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

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.1.3. 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.

6.5 Логічні операції

Дану групу утворюють 25 команд (див. Додаток Б), 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.

6.6 Команди передачі управління

До даної групи команд (див. Додаток Б) 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.

6.6.1 Довгий перехід

Перехід по всьому адресному простору пам'яті програм. У команді міститься повна 16-бітова адреса переходу (ad16). Трьохбайтні команди довгого переходу містять в Мнемокод 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.

6.6.2 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.

6.6.3 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. Відносний адреса переходу (rel) міститься в другому байті команди.

6.6.4 Непрямий перехід

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

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

6.6.5 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 прямо-адресований байт (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 роблять впливу на прапори. Команда CJNE встановлює прапор C, якщо перший операнд виявляється менше другого. Команда JBC скидає прапор C в разі переходу.

6.6.6 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 обслугованого рівня.

6.7 Операції з битами

Відмінною особливістю даної групи команд (див. Додаток 5) є те, що вони оперують з однобітними 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 бі-тов використовується прямий восьмизарядний адреса (bit). Непряма адресація бітів неможлива.

6.8 Завдання до лабораторної роботи

1 Ознайомитися з системою команд та можливостями умовних переходів у мікроконтролерах на базі 8051.

2 Нарисувати блок-схему алгоритму роботи кодового замка.

3 Нарисувати блок схему і написати програму для виводу на LCD-дисплей статусу замка при підключенні його до МК по 4-бітному інтерфейсу у першу строку та прогресу вводу коду з клавіатури у другу строку.

4 Здійснити симуляцію роботи МПС. За наявності помилок знайти і виправити помилку алгоритму чи самої програми.

5 Оформити звіт, занести до нього результати моделювання. Зробити висновки.

6.9 Контрольні запитання

1 Принцип побудови та алгоритм роботи кодового замка;

2. Брязкіт контакту та спосіб його усунення;

3. Небезпека одночасного натискання двох і більше клавіш одного рядка, спосіб його усунення.

6.10 Приклад виконання завдання

1 Створити проект як в прикладі 1.6 та використовуючи інструменти програми у вкладці **Shematic Capture** зібрати схему як на рис.:

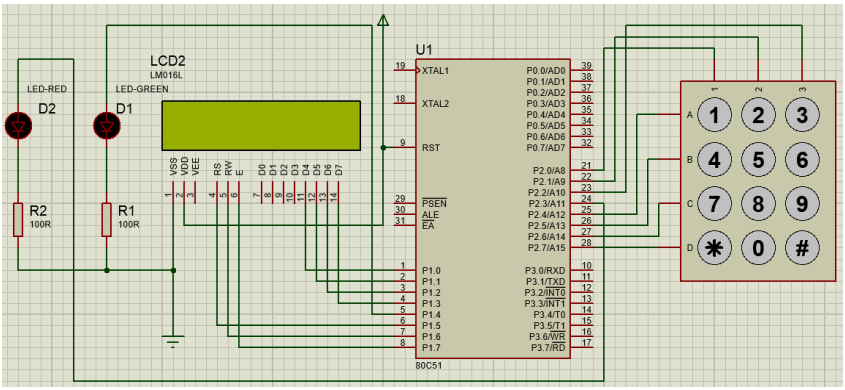


Рисунок 6.3 –Схема кодового замка

Лістинг прикладу програми для завдання:

```
;===================================================================
; DEFINITIONS
;===================================================================

#include "io8051.h"                ; Include register definition file

;===================================================================
; VARIABLES
;===================================================================
DB0 = P1.0
DB1 = P1.1
DB2 = P1.2
DB3 = P1.3

RS  = P1.5
RW  = P1.6
EN  = P1.7

K1 = P2.0
K2 = P2.1
K3 = P2.2
KA = P2.4
KB = P2.5
KC = P2.6
KD = P2.7
```

```
;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

      ; Reset Vector
      ASEGN CODE_SEG02:CODE,0
      ljmp  Start

;=====================================================================
; CODE SEGMENT
;=====================================================================

      RSEG CODE_SEG:CODE            ; Switch to this code segment.
                                    ; Register bank 0 by default

Start:
      lcall delay_20ms
      lcall init_lcd              ;виклик підпрограми ініціалізації
LCD
      MOV P2,#0xFF
      lcall start_c

      MOV ACC,#0x80                  ;курсор в початок першої строки
      lcall send_com              ;відправлення команди
      mov dptr, #mesg1              ;запис у регістр-показник адреси
строки
      lcall wr_string              ;виклик підпрограми виводу строки


      MOV ACC,#0xC0                  ;курсор в початок другої строки
      lcall send_com              ;відправлення команди

Loop:
;опитування ряду A
;опитування ряду A кнопка 1
      clr   KA
      JB    K1, Klb11
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x31
      lcall compare
      lcall delay_100ms
Klb12:JNB   K1, Klb12
;опитування ряду A кнопка 2
Klb11:JB    K2, Klb21
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x32
      lcall compare
      lcall delay_100ms
Klb22:JNB   K2, Klb22
;опитування ряду A кнопка 3
Klb21:JB    K3, Klb31
```

```
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x33
      lcall compare
      lcall delay_100ms
Klb32:JNB   K3, Klb32
;опитування ряду B
;опитування ряду B кнопка 4
Klb31:setb  KA
      clr   KB
      JB    K1,  Klb41
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x34
      lcall compare
      lcall delay_100ms
Klb42:JNB   K1, Klb42
;опитування ряду B кнопка 5
Klb41:JB    K2,  Klb51
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x35
      lcall compare
      lcall delay_100ms
Klb52:JNB   K2, Klb52
;опитування ряду B кнопка 6
Klb51:JB    K3, Klb61
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x36
      lcall compare
      lcall delay_100ms
Klb62:JNB   K3, Klb62
;опитування ряду C
;опитування ряду C кнопка 7
Klb61:setb  KB
      clr   KC
      JB    K1, Klb71
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x37
      lcall compare
      lcall delay_100ms
Klb72:JNB   K1, Klb72
;опитування ряду C кнопка 8
Klb71:JB    K2, Klb81
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x38
      lcall compare
      lcall delay_100ms
Klb82:JNB   K2, Klb82
;опитування ряду C кнопка 9
```

```
Klb81:JB    K3, Klb91
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x39
      lcall compare
      lcall delay_100ms
Klb92:JNB   K3, Klb92
;опитування ряду C
;опитування ряду C кнопка *
Klb91:setb  KC
      clr   KD
      JB    K1, KlbA1
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x2A
      lcall compare
      lcall delay_100ms
KlbA2:JNB   K1, KlbA2
;опитування ряду C кнопка 0
KlbA1:JB    K2, KlbB1
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x30
      lcall compare
      lcall delay_100ms
KlbB2:JNB   K2, KlbB2
;опитування ряду C кнопка #
KlbB1:JB    K3, KlbC1
      MOV   ACC, #0x2A
      lcall send_dat
      MOV   ACC, #0x23
      lcall compare
      lcall delay_100ms
KlbC2:JNB   K3, KlbC2
KlbC1:setb  KD
      ljmp  Loop                    ;нескінченний цикл

start_c:
      setb P2.3
      clr  P1.4
      mov dptr, #code
      MOV R6, DPH
      MOV R7, DPL
      MOV ACC,#0x01                 ;очистка дисплея, курсор в початок
екрану
      lcall send_com                ;відправлення команди
      MOV ACC,#0x80                   ;курсор в початок першої строки
      lcall send_com                ;відправлення команди
      mov dptr, #mesg1                ;запис у регістр-показник адреси
строки
      lcall wr_string                ;виклик підпрограми виводу строки
      MOV ACC,#0xC0                   ;курсор в початок першої строки
      lcall send_com                ;відправлення команди
```

```
      ret
compare:
      mov R0, ACC                ;збереження символу
      MOV DPH, R6                ;встановлення dptr
      MOV DPL, R7                ;на порівнюваний символ
      clr  a                      ;очистка акумулятора
      movc a, @a+dptr            ;запис в акумулятор байту по
адресі в DPTR
      SUBB A, R0
      jnz start_c
      inc dptr
      MOV R6, DPH
      MOV R7, DPL
      clr ACC                      ;очистка акумулятора
      movc a, @a+dptr            ;запис в акумулятор байту по
адресі в DPTR
      jz   unlock
      ret

unlock:
      MOV ACC,#0x01              ;очистка дисплея, курсор в початок
екрану
      lcall send_com             ;відправлення команди
      MOV ACC,#0x80                ;курсор в початок першої строки
      lcall send_com             ;відправлення команди
      mov dptr, #mesg2             ;запис у регістр-показник адреси
строки
      lcall wr_string            ;виклик підпрограми виводу строки
      mov dptr, #code
      MOV R6, DPH
      MOV R7, DPL
      clr  P2.3
      setb P1.4
      ret
wr_string:
      clr  a                       ;очистка акумулятора
      movc a, @a+dptr            ;запис в акумулятор байту по
адресі в DPTR
      inc dptr                     ;збільшення DPTR на одиницю
      jz  str_end                ;перевірка на кінецб стороки, якщо
так - вихід
      lcall send_dat             ;якщо НЕ кінецб строки - відправка
данних у LCD
      jnz wr_string                ;якщо НЕ кінецб строки - перехід
до мітки wr_string
str_end:
      ret

init_lcd:
      MOV P1,#0x03               ;8 розрядів
      clr  RS
      clr  RW                    ;RW=0 (запуск)
      lcall strob                  ;стробуючий імпульс
```

```
      MOV P1,#0x02                ;4 розряди;2 строки;символи 5*8
точок
      clr  RS
      clr  RW                     ;RW=0 (запуск)
      lcall strob                    ;стробуючий імпульс
      MOV ACC,#0x28               ;4 розряди;2 строки;символи 5*8
точок
      lcall send_com              ;відправлення команди
      MOV ACC,#0x08               ;дисплей вимкнений, курсор
вимкнений
      lcall send_com              ;відправлення команди
      MOV ACC,#0x01               ;очистка дисплея, курсор в початок
екрану
      lcall send_com              ;відправлення команди
      MOV ACC,#0x06               ;Зсув показника пам'яті вправо,
зсув екрана
                                 ;заборонений
      lcall send_com              ;відправлення команди
      MOV ACC,#0x0C               ;ввімкнення дисплею
      lcall send_com              ;відправлення команди
      ret

send_dat:
      setb RS                     ;RS=1 (данні)
      lcall send                  ;підпрограмма відправки
      ret

send_com:
      clr  RS                      ;RS=0 (команда)
      lcall send                   ;підпрограмма відправки
      ret
send:
      JB  ACC.4, lab11
      clr  DB0
lab11:JNB ACC.4, lab12
      setb DB0
lab12:JB  ACC.5, lab21
      clr  DB1
lab21:JNB ACC.5, lab22
      setb DB1
lab22:JB  ACC.6, lab31
      clr  DB2
lab31:JNB ACC.6, lab32
      setb DB2
lab32:JB  ACC.7, lab41
      clr  DB3
lab41:JNB ACC.7, lab42
      setb DB3
lab42:
      clr  RW                      ;RW=0 (запуск)
      lcall strob                     ;стробуючий імпульс
      JB  ACC.0, lab51
      clr  DB0
```

```
lab51:JNB ACC.0, lab52
      setb DB0
lab52:JB  ACC.1, lab61
      clr  DB1
lab61:JNB ACC.1, lab62
      setb DB1
lab62:JB  ACC.2, lab71
      clr  DB2
lab71:JNB ACC.2, lab72
      setb DB2
lab72:JB  ACC.3, lab81
      clr  DB3
lab81:JNB ACC.3, lab82
      setb DB3
lab82:clr  RW                        ;RW=0 (запуск)
      lcall strob                       ;стробуючий імпульс
      clr  RS                         ;очистка RS
      ret

strob:
      setb EN                         ;початок стробуючого імпульсу
      lcall delay_1600mks            ;затримка
      clr  EN                         ;кінець стробуючого імпульсу
      ret
delay:MOV R2,100
count:DJNZ R2, count
      ret

;=====================================================================
delay_20ms:
     MOV R3, #50                   ;завантаження числа X2
COUNT11: MOV R4, #200               ;завантаження числа X3
COUNT12: DJNZ R4, COUNT12           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT11               ;декремент R3 та цикл, якщо не
рівний 0
     ret                           ;повернення з підпрограмми затримки
;=====================================================================
delay_100ms:
     MOV R3, #255                   ;завантаження числа X2
COUNT31: MOV R4, #255               ;завантаження числа X3
COUNT32: DJNZ R4, COUNT32           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT31               ;декремент R3 та цикл, якщо не
рівний 0
     ret                           ;повернення з підпрограмми затримки
;=====================================================================
delay_1600mks:
     MOV R3, #4                      ;завантаження числа X2
COUNT21: MOV R4, #200               ;завантаження числа X3
COUNT22: DJNZ R4, COUNT22           ;декремент R4 та цикл, якщо не
рівний 0
     DJNZ R3, COUNT21               ;декремент R3 та цикл, якщо не
```

```
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
delay_40mks:
     MOV R4, #17                  ;завантаження числа X3
COUNT42: DJNZ R4, COUNT42          ;декремент R4 та цикл, якщо не
рівний 0
     ret                          ;повернення з підпрограмми затримки
;=====================================================================
mesg1:   db  "CLOSED",0
mesg2:   db  "UNLOCKED",0
code:    db  "12345",0
;=====================================================================
          END
```

Додаток А
Приклад оформлення титульної сторінки

Міністерство освіти та науки України

Національний університет «Запорізька політехніка»

Кафедра ІБ та Н

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

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

Прийняв:
(посада) _____
(Ініціали, Прізвище)

20___ р.

Додаток Б
Набір команд мікроконтролерів з архітектурою 8051

Таблиця Б.1 – Група команд передачі даних

Назва команди	Мнемокод	Т	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
Пересилання за прямим адресою акумулятора	MOV ad,A	3	(ad) <= (A)
Пересилання за прямим адресою регістра	MOV ad,Rn	3	(ad) <= (Rn)
пересилання прямоадресованого байта за прямим адресою	MOV add,ads	9	(add) <= (ads)
Пересилання байта з РПД по прямому 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)

Продовження таблиці Б.1

Назва команди	Мнемокод	Т	Операція
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)
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 прямоадресованим 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)

Таблиця Б.2 – 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 прямоадресованим 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 прямоадресованим 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, затем если (A4_7) > 9 || ((C)=1), то (A4_7) <= (A4_7) + 6

Продовження таблиці Б.2

Назва команди	Мнемокод	Т	Операція
Віднімання з акумулятора регістра і позички	SUBB A,Rn	1	(A) <= (A) - (C) - (Rn)
Віднімання з акумулятора прямоадрсованого 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 прямоадресуемого 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 прямоадресованого байта	DEC ad	3	(ad) <= (ad) - 1
Декремент байта з РПД	DEC @Ri	1	((Ri)) <= ((Ri)) - 1
Множення акумулятора на регістр В	MUL AB	1	(B)(A) <= (A)*(B)
Розділення акумулятора на регістр В	DIV AB	1	(A)(B) <= (A)/(B)

Таблиця Б.3 – Група команд логічних операцій

Назва команди	Мнемокод	T	Операція
Логічне І акумулятора і регістра	ANL A,Rn	1	(A) <= (A) && (Rn)
Логічне І акумулятора і прямоадресованого байта	ANL A,ad	3	(A) <= (A) && (ad)
Логічне І акумулятора і байта з РПД	ANL A,@Ri	1	(A) <= (A) && ((Ri))
Логічне І акумулятора і константи	ANL A,#d	2	(A) <= (A) && #d
Логічне І прямоадресованого байта і акумулятора	ANL ad,A	3	(ad) <= (ad) && (A)
Логічне І прямоадресованого 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 прямоадресованого 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 прямоадресованого байта і акумулятора	ORL ad,A	3	(ad) <= (ad) || (A)
Логічне АБО прямоадресованого байта і константи	ORL ad,#d	7	(ad) <= (ad) || #d

Продовження таблиці Б.3

Назва команди	Мнемокод	Т	Операція
Що виключає Або акумулятора і регістра	XRL A,Rn	1	(A) <= (A) ^ (Rn)
Що виключає Або акумулятора і прямоадресованого 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 прямоадресованого байта і акумулятора	XRL A,#d	2	(A) <= (A) ^ #d
Що виключає Або прямоадресованого 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)

Таблиця Б.4 – Група команд операцій з бітами

Назва команди	Мнемокод	Т	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)

Таблиця Б.5 – Група команд передачі управління (управління послідовністю)

Назва команди	Мнемокод	Т	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

Продовження таблиці Б.5

Назва команди	Мнемокод	T	Операція
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
	DJNZ Rn,rel	5	(PC) <=

Декремент регістра і перехід, якщо не нуль			(PC) + 2 (Rn) <= (Rn) - 1, если (Rn)<>0, то (PC) <= (PC) + rel

Продовження таблиці Б.5

Назва команди	Мнемокод	T	Операція
Декремент прямоадресовaваного 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 прямоадресованим 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

Продовження таблиці Б.5

Назва команди	Мнемокод	T	Операція
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**

Продовження таблиці Б.5

Назва команди	Мнемокод	Т	Операція
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**