МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

з вивчення дисципліни

“Мікропроцесорні пристрої”

та виконання контрольної роботи
для студентів спеціальності

141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА
заочної форми навчання

2018

Методичні вказівки з вивчення дисципліни «Мікропроцесорні пристрої» та виконання контрольної роботи для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА заочної форми навчання. /Укл: В.В. Осадчий, О.С. Назарова. - Запоріжжя: ЗНТУ, 2018. – 42 с.

Укладачі:

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

Рецензент: В.І. Бондаренко, к.т.н., доцент

Відповідальний за випуск: В.І. Бондаренко, к.т.н., доцент

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 10 від 02.07.2018 р.

Затверджено
на засіданні НМК ЕТФ
протокол № 01 від 23.08.2018 р.

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки містять основні питання, вихідні дані та завдання контрольної роботи (КР), а також рекомендації щодо вивчення дисципліни «Мікропроцесорні пристрої» у відповідності до навчальних планів ОКР бакалаврів.

КР сприяє розширенню та закріпленню теоретичних знань з дисципліни при вирішенні конкретних практичних завдань, розвиває навички ведення самостійної творчої роботи та професійні якості.

У додатку подано перелік команд мікроконтролера Intel 8051, який включає назву, мнемокод та опис дій, що виконуються.

Для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА заочної форми навчання.

Основні питання дисципліни «Мікропроцесорні пристрої»

1. Процедура виведення інформації на семисегментний індикатор.
2. Принцип роботи портів МК родини x51.
3. Опитування двійкового давача. Очікування статичного сигналу.
4. Принцип роботи семисегментних індикаторів.
5. Формування часових затримок малої і великої тривалості програмним методом.
6. Робота з перериваннями. Поняття вектора переривання.
7. Регістр масок переривання.
8. Регістр пріоритетів переривань.
9. Програмна реалізація обробки переривання (порівняння з викликом підпрограми).
10. Зміна пріоритетів переривань.
11. Режими роботи таймера-лічильника.
12. Регістр керування таймера.
13. Регістр статуса таймера.
14. Частота змінювання вмісту таймера/лічильника при роботі в режимі таймера.
15. Формування часової затримки за допомогою таймера
16. Максимальна тривалість часової затримки, що реалізується за допомогою таймера (для режимів 0, 1, 2).
17. Видача сигналів керування на зовнішні пристрої на прикладі ADuC 841.

Загальні вимоги до виконання контрольної роботи

Кожний студент заочної форми навчання в процесі вивчення дисципліни виконує контрольну роботу (КР), яка охоплює основні розділи дисципліни.

В процесі виконання КР студент повинен:

- розкрити одне теоретичне питання згідно з варіантом;
- виконати завдання згідно з варіантом, у КР навести умови, вихідні дані, текст програмы з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми та копію екрану з результатом виконання програми у Proview.

Варіанти індивідуальних завдань для кожного студента визначаються викладачем.

КР виконується на аркушах формату А4. Текст може бути виконаний рукописним способом або набраний на комп'ютері у редакторі Word, шрифт 14пт, міжрядковий інтервал 1,5; поля верхнє, нижнє, зліва та зправа по 20 мм. Роздрукована КР має бути зброшурована злівої сторони.

Послідовність розділів:

- титульна сторінка;
- завдання 1 , яке містить розгорнуту відповідь на теоретичне питання згідно варіанту;
- завдання 2, яке містить вихідні дані згідно варіанту, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми та копію екрану з результатом виконання програми у Proview;
- перелік використаної літератури.

При виконанні КР студент може користуватися не тільки літературою, що рекомендована, але і будь-якою доступною навчальною та технічною.

Рекомендації до виконання завдання 2

Засоби ProView для налагодження взаємодії з об'єктами керування.

Для прискорення процесу розробки керуючих програм існують спеціальні програмні засоби, що називаються інструментальним програмним забезпеченням. Одним з таких засобів є Proview від Franklin Software. Вказане інструментальне програмне забезпечення (ПЗ) дозволяє набирати, редагувати та зберігати вихідний текст програми, перетворювати його в машинний код, здійснювати перевірку працездатності програми, як окремих її частин, так і вцілому.

Інтегровані програмні емулятори апаратних засобів дозволяють імітувати роботу портів і таймерів мікроконтролера, оброблювати зовнішні переривання й переривання від таймерів.

Інструментальне ПЗ Proview фірми Franklin Software Inc. містить декілька вікон, що призначені для налагодження взаємодії мікроконтролера з об'єктами керування.

У вікні Main Registers (рисунок 1) відображається вміст лічильника команд PC, акумулятора ACC, слова стану процесу PSW, показчика стеку SP, показчика даних DPTR, допоміжного акумулятора B, біту позики/перенесення C, регістру масок переривань IE; регістру блокувань преривань EA, показчика банку регістрів RB, регістрів загального призначення R0-R7; портів P0-P3, регістру керування/статусу таймера TCON, регистрів таймерів THL0-THL2; регістру керування потужністю PCON.

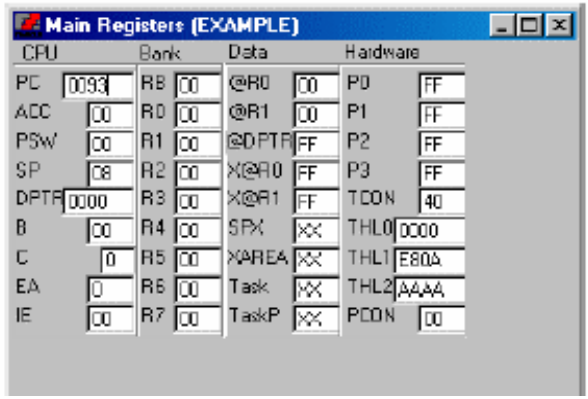


Рисунок 1 – Вікно регістрів

Крім того, через пункт Hardware меню View (рисунок 2) можна відкрити вікна паралельних портів контролера переривань, таймерів і послідовного порту.

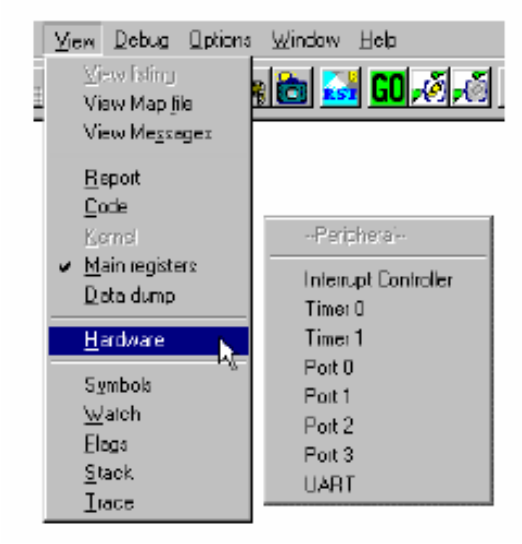


Рисунок 2 – Меню View

У інформаційному вікні контролера переривань «Interrupt Controller» (рисунок 3) вказується статус і пріоритет всіх джерел переривань: «Enable» - наявність переривання, «Not Enable» - відсутність переривання.

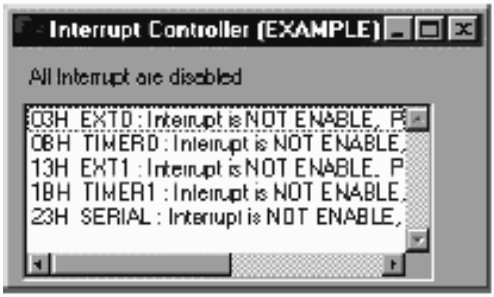
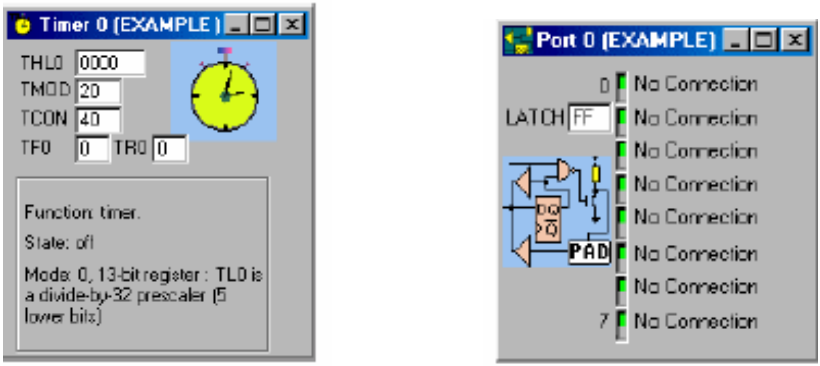


Рисунок 3 – Вікно контролера переривань

В рядку стану (рисунок 4) при виконанні програми у покроковому режимі або в режимі анімації показується поточний реальний час.

Рисунок 4 – Рядок стану



а) б)

а - вікно стану таймеру; б - вікно стану паралельного порту

Рисунок 5 – Вікна Proview таймера та паралельного порту

У вікнах таймерів (рисунок 5) відображається вміст 16-бітного таймера/лічильника THLx, регістру режиму роботи таймера/лічильника TMOD, регістра керування/статусу таймера TCON, флаг переповнення таймера TFx, біт керування таймера TRx. В цьому вікні також вказується стан і режим роботи таймера.

У вікнах паралельних портів (рисунок 5, б) відображається вміст регістру-фіксатора LATCH і окремих ліній порту. Стан ліній порту може бути змінено за допомогою миші.

У вікні послідовного порту UART (рисунок 6) відображаються дані буфера передавача Buffer, режим роботи і швидкість. У буфер приймача Input може бути введена послідовність байтів. Інформація може бути представлена у символьному вигляді ASCII або у шістнадцятковій формі HEX. Буфер очищується за допомогою кнопки Reset Buffer.

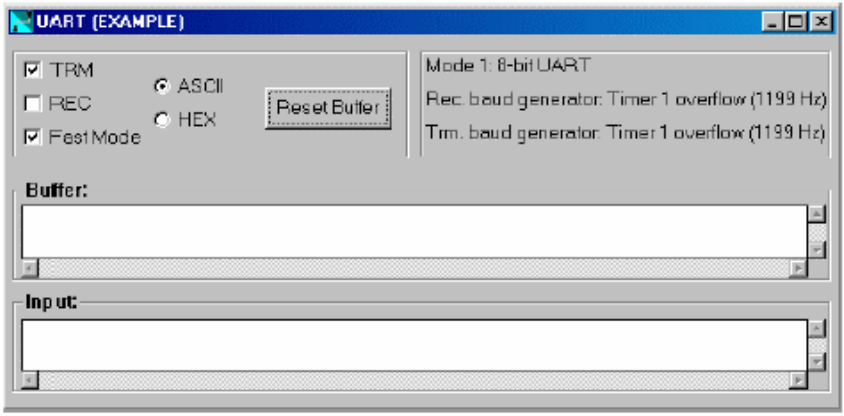


Рисунок 6 – Вікно послідовного порту

При відлагодженні програми (рисунок 7) часто виникає необхідність призупинити її виконання на певній команді.

Breakpoints – це маркери у вашій програмі, які примушують відлагоджувальник зупинити її виконання у «реальному масштабі часу». Ви можете встановлювати маркери на будь-яких командах програми (рисунок 8).

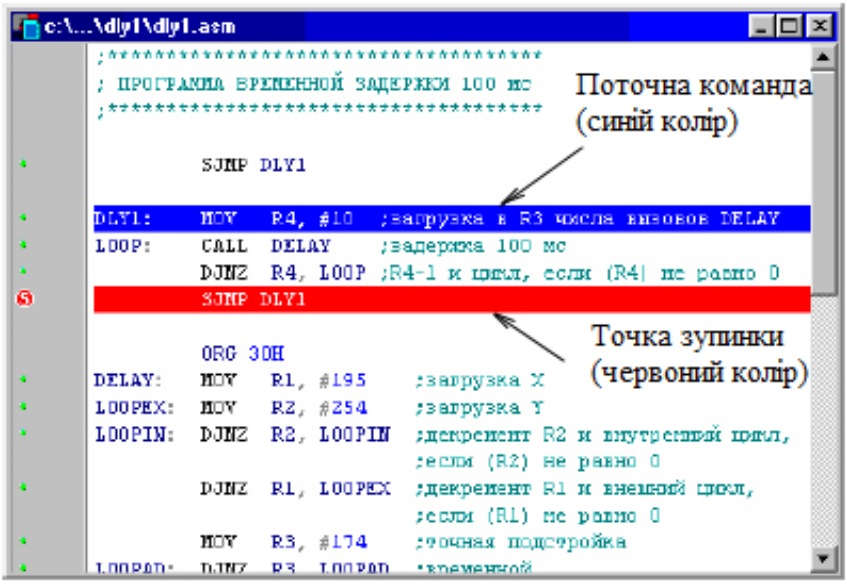
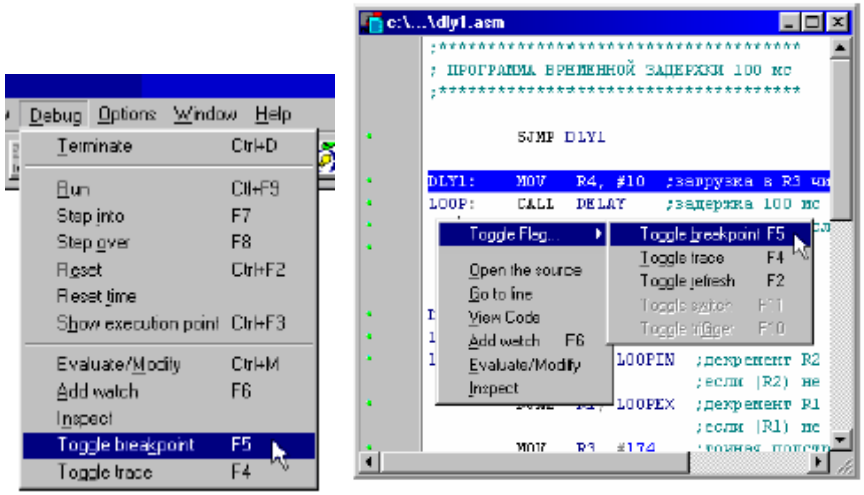


Рисунок 7 – Вікно відлагодження



а) б)

а) з головного меню; б) з контекстного меню

Рисунок 8 – Встановлення Breakpoint

Послідовність написання та відлагодження програми.

Ознайомитися з теоретичними відомостями та прикладами з тематики запропонованого завдання.

Скласти блок-схему виконання завдання:

- виділити із умов завдання вихідні дані;
- визначити послідовність виконання дій для вирішення задачі;
- визначити регістри, в яких будуть зберігатися необхідні дані, вигадати для них коротке та інформативне ім'я, а також чіткий описуючий коментар;
- за необхідності визначити перелік дій, які будуть (якщо буде необхідність) повторюватися в ході виконання завдання і вирішити, що саме для цього використати: цикл (декілька циклів) або підпрограму.

Згідно складеної блок-схеми написати текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин усієї програми. Бажано уникати коментарів, що дублюють дію окремої команди, тобто дані, які відомі або можуть бути взяті з опису ситеми команд. Коментар повинен роз'яснювати з якою метою використовується команда (група команд) у конкретній програмі.

Наприклад:

```
не бажано (погано) INC R1      ;R1+1→R1
бажано (добре )    INC R1      ;збільшення вказівника
                               ;елементів масиву
```

Процес створення програми бажано розбити на умовні частини:

- складання основного переліку дій для виконання завдання;
- введення вихідних даних;
- опис дій, які виконуються циклічно;
- підпрограми, які будуть викликатися під час виконання основної програми.

Перевірити наявність міток у тексті програми і команд переходу за мітками, а також команд виклику підпрограм і виходу з підпрограм.

Наприклад:

```
org 0h
     mov r0, #30h ; адреса початкового елемента першого
                    масиву
     mov r7, #8h  ; кількість елементів першого масиву
     call count_null; виклик п/п підрахунку
                    кількості елементів масиву, що
                    дорівнюють нулю
     mov b, a     ; збереження результату виконання
                    підпрограми
     mov r0, #40h ; адреса початкового елемента другого
                    масиву
     mov r7, #8h  ; кількість елементів другого масиву
     call count_null ; виклик п/п підрахунку кількості
                    елементів масиву, що дорівнюють нулю
     cjne a, b, m_ne; порівняння результатів підрахунків
     mov 20h, #0h ; запис у комірку 20h числа 0h, якщо
                    кількість елементів, що дорівнюють
                    нулю, в обох масивах однакова
      jmp m_end ;
m_ne:
     jc m1 ;
     mov 20h, #2h ; запис у комірку 20h числа 2h, якщо
                    у другому масиві кількість елементів,
                    що дорівнюють нулю, більша, ніж у
                    першому
     jmp m_end

m1:
     mov 20h, #1h ; запис у комірку 20h числа 1h, якщо
                    у першому масиві кількість елементів,
                    що дорівнюють нулю, більша ніж у
                    другому

m_end:
     jmp m_ne;
     count_null:  ; п/п підрахунку кількості елементів
                    масиву, що дорівнюють нулю
     mov r6, #0h  ;встановлення початкового стану
                    лічильника
cnt_m1:           ; початок циклу
     mov a, @r0   ;
```

```
     jnz cnt_m2    ; перехід, якщо елемент масиву не
                   дорівнює нулю
     inc r6        ; збільшення вмісту лічильника на 1
cnt_m2:            ;
     inc r0        ; збільшення "вказівника" елементів
                   масиву на 1
     djnz r7, cnt_m1     ; перевірка завершення циклу
     mov a, r6     ; запис результату підрахунку у
                   акумулятор
     ret
END
```

Перевірити правильність програми, виконавши такі дії:

Набрати у вікні вводу програми одну з умовних частин всієї програми;

Зберегти файл з цим текстом програми
(File / Save as / [им'я файлу латинськими літерами]. asm);

Запустити проект для відлагодження (Project / Build all).

За наявності у тексті програми синтаксичних помилок, у вікні «Message» з'явиться виділене червоним кольором повідомлення «ERROR», яке вказує на рядок, що містить помилку.

Запустити проект для перевірки працездатності цієї частини програми (Debug / Start). При запуску програми доступні такі функції емулятора (таблиця 1).

Вікно «Main Registers» дозволяє перевіряти стан портів та регістрів та порівнювати його з бажаним.

У вікні «Code» відображуються адреси, коди, мнемоніки програми.

Перевірити стан і режим роботи таймерів можна додатково відкривши відповідні вікна (View / Hardware / Timer 0 та/або Timer 1, Timer 2).

За необхідності не програмного керування окремими лініями паралельних портів їх стан може бути змінений за допомогою миші у вікнах паралельних портів (View / Hardware / Port 0 та/або Port 1, Potr 2, Port 3).

Таблиця 1 - Функції емулятора

Граф. зображення	Назва	Функції
	«Animate»	Анімоване виконання програми – при запуску програми у відповідності до порядку виконання команд вони будуть виділені рядком синього кольору
	«Step into»	Покрокове виконання програми – при натисканні на цю кнопку буде виконана тільки одна команда
 	«Run» / «Stop»	Запуск/зупинка програми – після запуску програми ця ж кнопка використовується для зупинки виконання програми, при повторному натисканні програма буде запущена з того місця, на якому була зупинена.
	«Reset»	Повернення у початкове положення – при наступному запуску виконанння програми почнеться з першої команди.

Після того як відлагоджена одна частина програми, слід поступово доповнювати її іншими частинами до повного тексту програми і повної працездатності.

16

Завдання 1

Дати повну розгорнуту відповідь на одне із запитань, наведених нижче, згідно індивідуального варіанту, який визначає викладач.

1. Призначення семисегментного індикатора.
2. Підключення семисегментного індикатора.
3. Процедура виведення інформації на семисегментний індикатор.
4. Принцип роботи портів МК родини x51.
5. Запис у порт. Альтернативні функції порта.
6. Особливості роботи портів.
7. Опитування двійкового давача. Очікування статичного сигналу.
8. Принцип роботи семисегментних індикаторів.
9. Формування часових затримок малої тривалості (до 100 мкс) програмним методом.
10. Формування часової затримки великої тривалості (до декількох секунд) програмним методом.
11. Робота з перериваннями. Поняття вектора переривання.
12. Джерела переривання (чим воно може бути викликане).
13. Регістр масок переривання.
14. Регістр пріоритетів переривань.
15. Програмна реалізація обробки переривання (порівняння викликів підпрограм).
16. Обробка переривань (програмно-апаратна послідовність дій).
17. Зміна пріоритетів переривань.
18. Режими роботи таймера-лічильника.
19. Регістр керування таймера.
20. Регістр статуса таймера.
21. Частота змінювання вмісту таймера/лічильника при роботі в режимі таймера.
22. Формування часової затримки за допомогою таймера
23. Організація часової затримки: програмно й за допомогою таймера (порівняльний аналіз).

24. Максимальна тривалість часової затримки, що реалізується за допомогою таймера (для режимів 0, 1, 2).
25. Організація регістру керування TCON.
26. Організація регістру керування SCON.
27. Організація регістру дозволу переривань IE.
28. Організація регістру пріоритетів переривань IP.
29. Описати сигнали, які формують вміст таймера.
30. Видача сигналів керування на зовнішні пристрої на прикладі ADuC 841.

Завдання 2

Написати програму для реалізації секундоміра (00…59). Відповідно до рисунку 9. Початковий стан індикаторів згідно індивідуального завдання (таблиця 2). Натискання кнопки «пуск/стоп» запускає відлік часу, повторне натискання – зупиняє. Натискання кнопки «скидання» – обнуляє. Лічильник не круговий, при досягненні максимального значення – зупиняється. Перевищення періоду відліку відображається блиманням індикаторів.

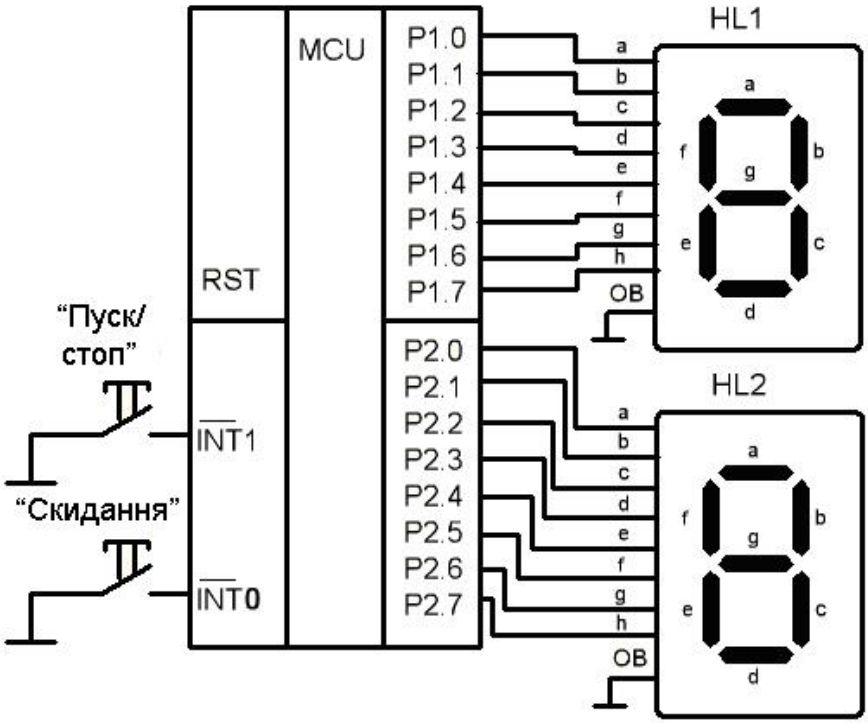


Рисунок 9 – Схема з'єднання МК з органами керування та індикаторами

Таблиця 2 – Дані для виконання індивідуального завдання

№ варіанта	Початковий стан
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30

Для зручності можна написати підпрограму-дешифратор для ССІ, яка брала б число з регістра А і повертала б відповідний цьому числу код ССІ, назвемо її, відповідно, Decode

```
DECODE:MOV DPTR,#TABSEMISEG;вказівник в
                           ;початок таблиці
MOVC A,@A+DPTR             ;зчитати код
RET                        ;і повернути через
                           ;акумулятор
                           ;0123456789
TABSEMISEG:
DB 00111111b ; '0'
DB 00000110b ; '1'
DB 01011011b ; '2'
DB 01001111b ; '3'
DB 01100110b ; '4'
DB 01101101b ; '5'
DB 01111101b ; '6'
DB 00000111b ; '7'
DB 01111111b ; '8'
DB 01101111b ; '9'
```

Цей фрагмент кода зручно помістити в кінець програм, в яких використовується цей тип індикаторів.

Тепер, якщо наш індикатор буде підключено, наприклад, до порту 0, нам для введення числа «5» будет достатньо трьох рядків

```
MOV A, #5
CALL DECODE
MOV P0, A
```

Це набагато зручніше, ніж щоразу задавати значення відповідне кожному числу байтів.

Короткі теоретині відомості.

Формування часової затримки за допомогою таймеру.

Недоліком програмного способу реалізації часової затримки є нераціональне використання ресурсів мікроконтролера: під час формування затримки він практично простоює, бо не може виконувати

жодних задач керування об'єктом. З іншого боку апаратні засоби дозволяють реалізовувати часові затримки на фоні роботи основної програми.

Режими роботи таймера/лічильника

Режим 0

У цьому режимі таймерний регістр має розрядність 13 біт. В момент переходу зі стану «всі одиниці» в стан «всі нулі» встановлюється флаг переривання від таймера TF1 (рисунок 10). Вхідний синхросигнал таймера 1 дозволений (поступає на вхід таймера/лічильника (Т/Л), коли керуючий біт TR1 встановлений в 1, а також, або керуючий біт GATE (блокування) є 0, або на зовнішньому виводі запиту переривання INT1 присутній рівень 1.

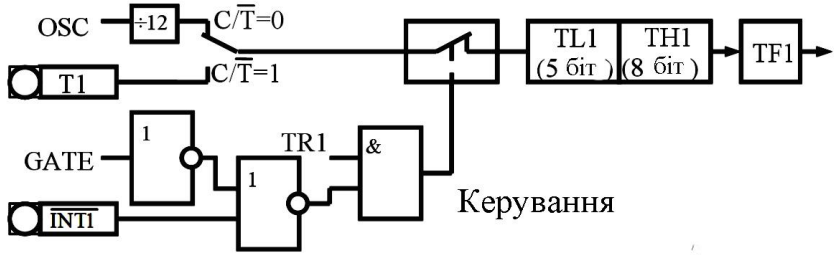


Рисунок 10 – Структура таймера/лічильника в режимі 0

Режим 1

Робота будь-якого Т/Л в режимі 1 такаж сама, як і в режимі 0, із-за того, що таймерний регістр (таблиця 3) має розрядність 16 біт.

Режим 2

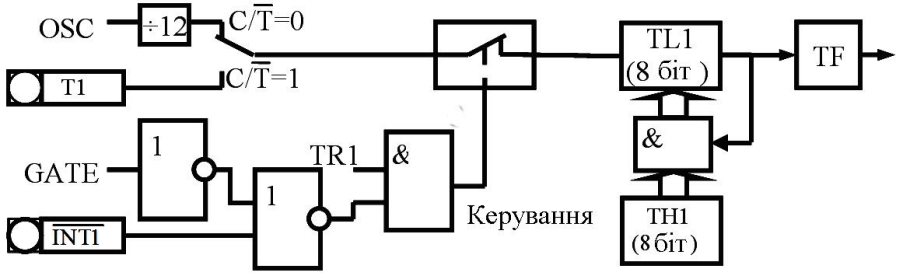


Рисунок 11 – Структура таймера / лічільника в режимі 2

В режимі 2 робота організована таким чином, що переповнення (перехід зі стану «всі одиниці» в стан «всі нулі») 8-бітного лічильника TL1 призводить не тільки до установки флагу TF1, але й автоматично

перезавантажує в TL1 вміст старшого байту (TH1) таймерного регістра, яке попередньо було задано програмним шляхом. Перезавантаження не змінює вміст TH1 (рисунок 11).

Таблиця 3 – Регістр керування/статуса таймера

Символ	Позиція	Ім'я та призначення
TF1	TCON.7	Флаг переповнення таймера 1. Встановлюється (стає рівним 1) апаратурно при переповненні таймера/лічильника. Скидається (стає рівним 0) при обробці переривання апаратно.
TR1	TCON.6	Біт керування таймера 1. Встановлюється/ скидається програмою для пуску/зупинки.
TF0	TCON.5	Флаг переповнення таймера 0. Встановлюється апаратурно. Скидається при обробці переривання.
TR0	TCON.4	Біт керування таймера 0. Встановлюється/ скидається програмою для пуску/зупинки таймера/лічильника.
IE1	TCON.3	Флаг фронту переривання 1. Встановлюється апаратурно, коли детектується зріз зовнішнього сигналу (INT1). Скидається при обробці переривання.
IT1	TCON.2	Біт керування типом переривання 1. Встановлюється/скидається програмно для специфікації запиту INT1 (зріз/ низький рівень).
IE0	TCON.1	Флаг фронту переривання 0. Встановлюється апаратурно, коли детектується зріз зовнішнього сигналу (INT0). Скидається при обробці переривання.
IT0	TCON.0	Біт керування типом переривання 0. Встановлюється/скидається програмно для специфікації запиту INT0 (зріз/ низький рівень).

На вхід таймера/лічильника (Т/Л) можуть подаватися сигнали синхронізації з частою 1 МГц (Т/Л в режимі таймера) або сигнали від

зовнішнього джерела (Т/Л в режимі лічильника). Обидва ці режими можуть використовуватися для формування затримок. Якщо використовувати Т/Л в режимі таймера повного формату (16 біт), то можна одержати затримки в діапазоні 1 – 65 536 мкс.

Як приклад розглянемо організацію часової чатримки 50 мс (TIMER). Вважається, що біт IE.7 встановлений, тобто знято блокування всіх переривань. Слід звернути увагу, що в даному випадку використана команда переводу мікроконтролера в режим холостого ходу, який припиняється після перебігу 50 мс. Цей режим реалізований в мікроконтролері Intel 80C51, який виготовлений по технології КМОП, тому при налагодженні програми слід вибирати вказаний тип мікроконтролера (пункт Debug меню Options).

```
 ;организація переходу на мітку NEXT при
                        ;переповнені Т/Л0
      ORG 0BH           ;адреса вектора переривання
                        ;від Т/Л0
      CLR TCON.4        ;зупинка Т/Л0
      RETI              ;вихід з процедури обробки
                        ;переривання
      ORG 30H           ;початкова адреса програми
TIMER:MOV TMOD,#01H     ;налаштування Т/Л0
      MOV TL0, #LOW(NOT(50000-1)) ;завантаження
      MOV TH0, #HIGH(NOT(50000-1));таймера для
                                  ;затримки 50 мс
      SETB TCON.4             ;старт Т/Л0
      SETB IE.1               ;дозвіл переривання
                              ;від Т/Л0
      SETB PCON.0             ;перехід в режим
                              ;холостого ходу
      NEXT:
```

Для вимірювання тривалості сигналу також може використовуватися таймер. Особливо ефективно використання з цією метою таймера в Intel 8051, що має вхід дозволу підрахунку (альтернативна функція входу INT). Сигнал, тривалість якого вимірюється, можна, наприклад, подавати на вхід INT0, а вимірювання тривалості при цьому буде виконуватися в Т/Л0. Програма

вимірювання тривалості «додатнього» імпульсу (MSTIMER) матиме такий вигляд:

```
MSTIMER:  MOV TMOD, #00001001B  ; налаштування Т/Л0
          MOV TH0, #0           ;скидання
          MOV TL0, #0           ;таймера
          SETB TCON.4           ;старт Т/Л0
WAIT0:    JNB P3.2, WAIT0       ;очікування 1
WAITC:    JB P3.2, WAITC        ;очікування 0
          CLR TCON.4            ;стоп Т/Л0
EXIT:                           ;вихід з процедури
```

Керування повинно бути передано програмі за умови, що на вході INT0 присутній низький рівень. Переривання від Т/Л0 і зовнішнє переривання по входу INT0 повинні бути заборонені.

Після завершення програми в Т/Л0 буде знаходитися число, пропорційне тривалості «додатнього» импульсу на вході INT0. Верхня межа вимірювання дорівнює 65 536 мкс, а максимальна похибка 1 мкс.

Якщо потрібно виміряти інтервал часу більшої тривалості, можна програмним способом підрахувати число переповнень таймера, тобто розширювати його разрядність за рахунок робочого регістра або комірки резидентної пам’яті даних.

Приклад:

При натисканні кнопки SB1 ініціювати переривання INT1 (рисунок 12) й засвітити через секунду світлодіод VD1, що підключений до P0.0 (затримку реалізувати за допомогою таймера).

Приклад для 16-бітного таймера

```
 ORG 0
 JMP MAIN   ;безумовний перехід на мітку
            ;main
 ORG 0BH    ;переривання від таймера ТЛ0
 JMP TC0    ;безумовний перехід на мітку
            ;TC0
 ORG 13H    ;зовнішнє переривання
 SETB TCON.4;запуск таймера
 RETI       ;повернення з переривання
```

```
MAIN: MOV IE, #00000110B    ;встановлення флагів
                ;переривань,які дозволені
      MOV TH0,#HIGH (65535-50000); налаштування
      MOV TL0,#LOW (65535-50000); таймера на
                ;режим 16-ти бітного
                ;без перезавантаження 2
      MOV TMOD, #00000001B;
      MOV R2,#20
      SETB TCON.2;переривання INT1 по фронту
                ;сигнала
      SETB IE.7  ;розблокувати всі дозволені
                ;переривання
      CLR P0.0
LOOP: JMP LOOP   ;бескінечний цикл в очікуванні
                ;переривання
TC0:  MOV TH0,#HIGH(65535-50000);перезавантаження
      MOV TL0,#LOW (65535-50000) ;таймера, бо
           ;автоперезавантаження - відсутнє

      DJNZ R2, L3;підрахунок кількості
                ;«спрацьовувань» таймера
      SETB P0.0  ;запалити світлодіод
      CLR TCON.4
L3:   RETI  ;повернення з п/п переривання
      END   ;кінець програми
```

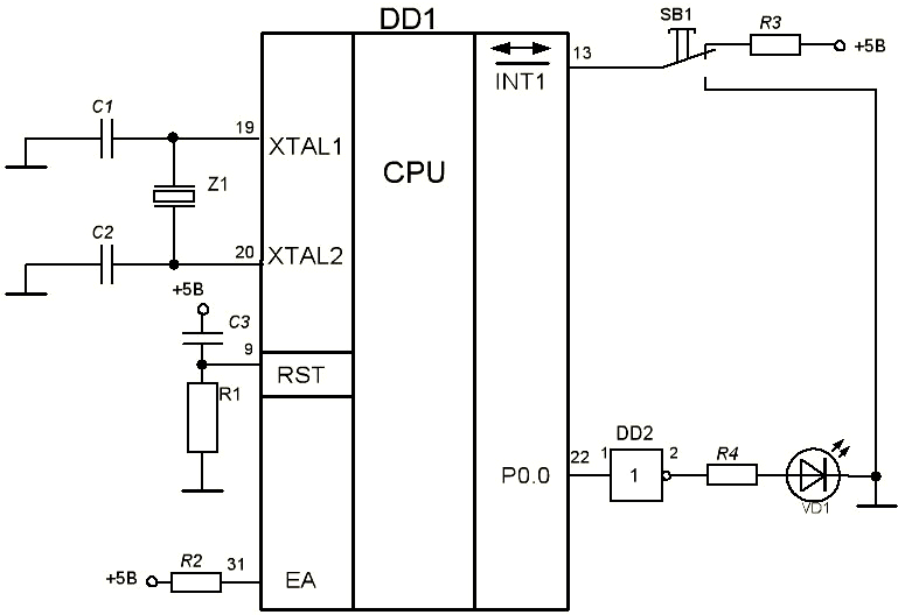


Рисунок 12 – Схема з'єднання МК з органами керування та світлодіодом

Зверніть увагу, як настроюється таймер.

Двобайтне число можна розділити на старший и молодший байт за допомогою директив відповідно High і Low. Таким чином, число, яке обчислюється відповідно до виразу в дужках, розділяється на 2 окремих байта. Зважаючи, що таймер інкрементний, то необхідно в байти таймера записувати число, з якого починається відлік до переповнення (65535 мкс). В наведеному прикладі потрібна затримка 50000 мкс, тому в дужках від максимального числа (65535 мкс) віднімається необхідне значення (50000 мкс).

```
mov TH0,#HIGH (65535-50000)  ;Заносимо число в старший
                             ;байт,
mov TL0,#LOW (65535-50000)   ;Заносимо число в молодший
                             ;байт.
```

Приклад для 8-бітного автоперезавантажувального таймера

```
      ORG 0
      JMP MAIN   ;безумовний перехід на мітку
                 ;MAIN
      ORG 0BH    ;переривання від таймера
      JMP TC0    ;безумовний перехід на мітку
                 ;TC0
      ORG 13H    ;зовнішнє переривання
      SETB TCON.4;запуск таймера
      RETI       ;повернення з переривання
MAIN: CLR P0.0   ;погасити світлодіод
      CLR IE.7
      MOV IE, #00000110B    ;дозволити переривання
                            ;від INT1 і Т/Л0
      MOV TMOD, #00000010B;режим таймера
      ;8-ми бітний з автоперезавантажуванням
      MOV TL0, #NOT(250-1)  ; налаштування на час
                            ;250мкс
      MOV TH0, #NOT(250-1)  ; налаштування на час
                            ;250мкс
      MOV R3, #LOW(4000)    ;4000разів по 250мкс=1с
      MOV R2, #HIGH(4000)   ;4000разів по 250мкс=1с
      SETB TCON.2           ;переривання INT1 по
                            ;фронту сигнала

      SETB IE.7  ; розблокувати всі переривання
L1:   JMP L1     ;безкінечний цикл в очікуванні
                 ;переривання
TC0:  DJNZ R3, L3
      DJNZ R2, L3
      SETB P0.0  ;запалити світлодіод
      CLR TCON.4 ;зупинка таймера 0
L3:   RETI       ;повернення з п/п переривання
      END        ;кінець програми
```

ПЕРЕЛІК ПОСИЛАНЬ

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

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

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

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

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

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

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

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

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

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

11. Осадчий, В. В. Ідентифікація ступеня завантаження двошвидкісного ліфта / В. В. Осадчий, О. С. Назарова, С. С. Шульженко // Електротехнічні та комп'ютерні системи. – 2018. – № 27(103). – С. 103-111. DOI: https://doi.org/10.15276/eltecs.27.103.2018.11

12. Осадчий, В. В. Лабораторный стенд для исследования микропроцессорных систем управления двухмассовым электроприводом / В. В. Осадчий, Е. С. Назарова, В. В. Брылистый, Р. И. Савилов // Електротехнічні та комп'ютерні системи. – 2016. – № 22(98).– С. 33-38. http://dx.doi.org/10.15276/eltecs.22.98.2016.05

Додаток А

Зразок оформлення титульної сторінки

Міністерство освіти і науки України
Запорізький національний технічний університет

Кафедра ЕПА

Контрольна робота
з дисципліни
«Мікропроцесорні пристрої»

Виконав:
студ. гр. Е-333 Іваненко І.І.

Перевірив:
доцент Петренко П.П.

2018

Додаток Б

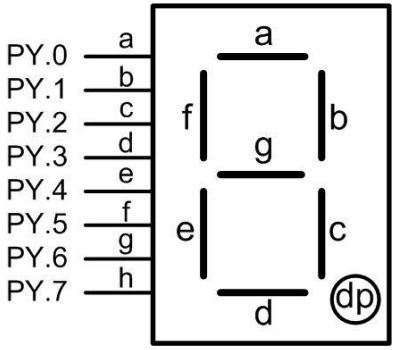
Представлення даних у двійковій, шістнадцятковій та десятковій системі числення

Таблиця Б.1 – Відповідність представлення даних у різних системах числення

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

Додаток В

Принцип представлення даних для виведення на семисегментний індикатор



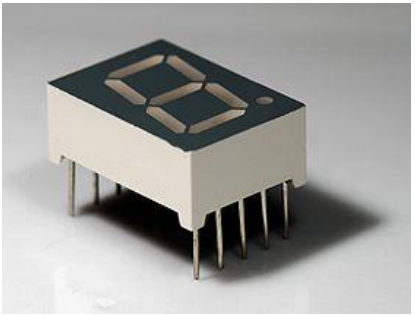


Рисунок В.1 - Семисегментий індикатор

Таблиця В.1 – Відповідність сегментів індикатора для виведення цифри «5»

Біт порту	PY.7	PY.6	PY.5	PY.4	PY.3	PY.2	PY.1	PY.0
Сегмент індикатора	DP	G	F	E	D	C	B	A
Бажаний стан	0	1	1	0	1	1	0	1

Додаток Г

Перелік команд мікроконтролера Intel 8051

Таблиця Г.1 - Група команд пересилання даних

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

Продовження таблиці Г.1

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

Таблиця Г.2 - Команди арифметичних операцій

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

Продовження таблиці Г.2

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

Продовження таблиці Г.2

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

Таблиця Г.3 - Команди логічних операцій

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

Продовження таблиці Г.3

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

Продовження таблиці Г.3

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

Таблиця Г.4 - Команди роботи з бітами

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

Продовження таблиці Г.4

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

Таблиця Г.5 - Команди передачі керування

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

Продовження таблиці Г.5

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

Продовження таблиці Г.5

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

Продовження таблиці Г.5

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

Т – тип команди;
Б – формат у байтах;
Ц – кількість машинних циклів, необхідних для оброблення команди.