**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет «Запорізька політехніка»**

# МЕТОДИЧНІ ВКАЗІВКИ

з виконання лабораторних робіт дисципліни

# «МІКРОКОНТРОЛЕРИ В СИСТЕМАХ КЕРУВАННЯ»

для студентів спеціальності
173 – Авіоніка
освітньої програми «Електротехнічні комплекси та системи літальних апаратів»
усіх форм навчання

**2023**

Методичні вказівки з виконання лабораторних робіт дисципліни «Мікроконтролери в системах керування» для студентів спеціальності 173 – Авіоніка освітньої програми «Електротехнічні комплекси та системи літальних апаратів» усіх форм навчання. /Укл: В.В. Осадчий, О.С. Назарова, Е.М. Кулинич - Запоріжжя: НУ «Запорізька політехніка», 2023. – 73 с.

Укладачі: В.В. Осадчий, к.т.н., доцент
О.С. Назарова, к.т.н., доцент
Е.М. Кулинич, к.т.н., доцент

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

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

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 06 від 22.02.2023 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 07 від 23.03.2023 р.

# ЗМІСТ

Передмова……. 4
1. Лабораторна робота №1
Ознайомлення із емулятором Franklin Software, вивчення його функціональних можливостей…….. 5
2. Лабораторна робота №2
Вивчення команд логічних операцій …… 12
3. Лабораторна робота №3
Вивчення команд роботи з бітами …… 15
4. Лабораторна робота №4
Вивчення команд непрямої адресації. Робота із масивами даних ……..... 18
5. Лабораторна робота №5
Паралельне введення-виведення інформації……..,……. 22
6. Лабораторна робота №6
Огранізація часових затримок програмним методом….. 26
7. Лабораторна робота №7
Система переривань……..…….. 35
8. Лабораторна робота №8
Апаратний таймер-лічильник……..…….. 41
9. Лабораторна робота №9
Послідовне пересилання інформації. Регістр зсуву….. 51
Перелік посилань…….. 61
Додаток А Перелік команд мікроконтролера Intel 8051…… 62
Додаток Б Представлення даних у двійковій, шістнадцятковій та десятковій системі числення ….. 73

# ПЕРЕДМОВА

Методичні вказівки містять опис дев’яти лабораторних робіт з дисципліни «Мікроконтролери в системах керування» у відповідності до навчальних планів ОКР бакалаврів спеціальності 173 – Авіоніка, рекомендації до їх виконання і мають два додатки.

В першій лабораторній роботі наведені короткі відомості про інструментальне програмне забезпечення для розробки та налагодження програм. Решта лабораторних робіт містить короткі теоретичні відомості, індивідуальні завдання та рекомендації щодо їх виконання.

У додатках подано перелік команд мікроконтролера Intel 8051, який включає назву, мнемокод, опис дій, що виконуються, та таблиця представлення даних у двійковій, шістнадцятковій та десятковій системі числення.

Для студентів спеціальності 173 – Авіоніка освітньої програми «Електротехнічні комплекси та системи літальних апаратів» усіх форм навчання.

# 1. ЛАБОРАТОРНА РОБОТА №1

## Ознайомлення із емулятором Franklin Software, вивчення його функціональних можливостей

**Мета:** ознайомитись із емулятором Franklin Software, вивчити його функціональні можливості.

### Короткі теоретичні відомості

ProView фірми Franklin Software Inc. – інтегрована середа розробки програмного забезпечення для однокристальних мікроконтролерів сімейства Intel 8051 і його клонів. Вона включає у себе все, що необхідно для створення, редагування, компіляції, трансляції, компонування, завантаження та відлагодження програм. А саме, стандартний інтерфейс Windows; повнофункціональний редактор вихідних текстів з виділенням синтаксичних елементів кольором; організатор проекту; транслятор з мови С; асемблер; відлагоджувач; вбудовану довідкову систему.

### Початок роботи

Запуск інтегрованого середовища ProView32 здійснюється через меню Пуск(Windows):Пуск →Програми → Franklin Software → ProView32.

Далі необхідно створити новий файл. Для цього у розділі меню «File» треба вибрати пункт «New», а також у вікні, що з’явилося, вибрати тип файлу «Assembler Files» (рисунок 1.1).

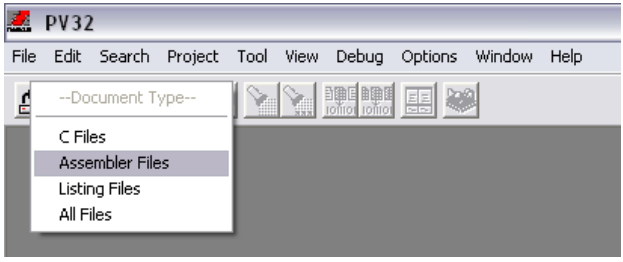



Рисунок 1.1 – Вибір типу файлу

Новий файл необхідно зберегти на диск (File→Save as…), присвоївши йому ім’я, яке складається з латинських букв та цифр. Довжина імені не повинна перевищувати 8 символів, розширення файла «.asm».

Після цього можна переходити до написання програми для мікроконтролера на мові Асемблер. Згідно завдання та з урахуванням особливостей синтаксису цієї мови складається програма. Після того, як текст програми набрано та збережено, переходимо до компіляції програми – переводу тексту програми у машинний код, який призначено для завантаження у пам’ять програм мікроконтролера.

Для компіляції програми необхідно вибрати пункт меню «Project»→ Build all. Коли процес компіляції завершиться у вікні повідомлень «Message» (рисунок 1.2) буде відображено повідомлення про завершення цього процесу. У випадку виявлення будь-якої помилки, повідомлення про них буде відображено також у цьому вікні.


Рисунок 1.2 – Вікно повідомлень Message

У разі, якщо програма не містить помилок, можна запускати відлагоджувальник. Для цього у меню «Debug» вибираємо пункт «Start». При першому запуску з’явиться віно (рисунок 1.3), у якому необхідно вказати тип мікроконтролера «Microcontroller: 8051», а також тактову частоту у мегагерцах.

Запустивши відлагоджувальник з’являються декілька вікон (рисунок 1.4): вікно з текстом програми; вікно коду «Code», у якому кожній інструкції на асемблері відповідає її машинний код; вікно основних регістрів «Main Registers», у якому відображено стан кожного з регістрів. У рядку стану показано час, за який реальний мікроконтролер виконав би команди програми з моменту старту до команди, що виконується саме зараз (виділена синім фоном).

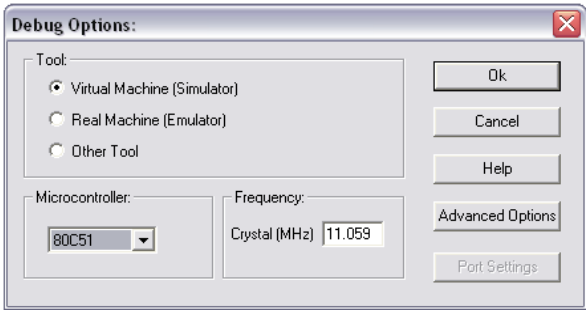



Рисунок 1.3 – Вибір типу контролера та частоти

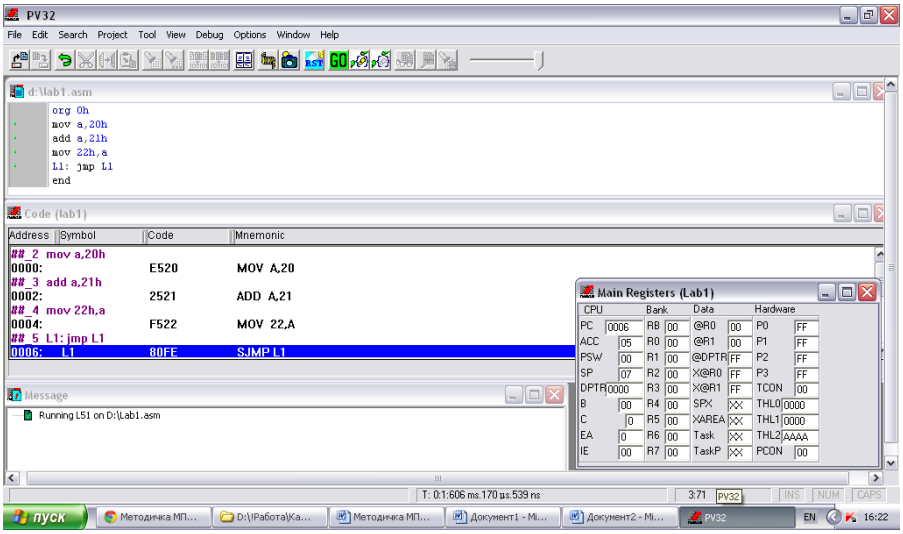



Рисунок 1.4 – Екран режиму «Debug».

Використовуючи меню «View» можна додавати нові вікна. Наприклад, View→Data dump дозволяє відобразити вікно з вмістом різних областей пам’яті даних мікроконтролера: Xdata – зовнішня пам’ять (ОЗП) даних; Data – внутрішня пам’ять даних (внутрішнє ОЗП); Sfr – область регістрів спеціальних функцій (РСФ); Bit – область прямо адресованих бітів. У завданнях необхідно вводити тестові числа у комірки внутрішньої пам’яті даних (ВПД) для перевірки правильності написаних програм. Для цього виконуємо View→Data

dump→ Data view. У вікні «Data view» (рисунок 1.5) відображається вміст комірок ВПД, який згруповано по 8 комірок у рядку із зазначенням першої комірки у групі (адреса вказана зліва з двокрапкою). Після введення даних у комірку треба натиснути кнопку «ENTER» на клавіатурі.

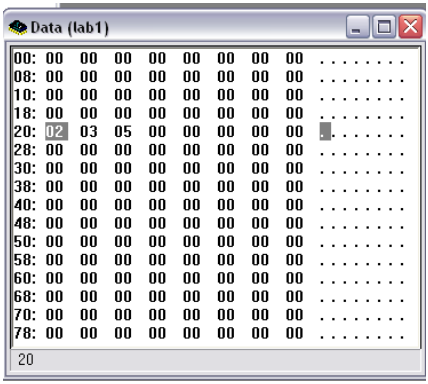



Рисунок 1.5 – Вікно відображення вмісту комірок ВПД «Data view».

Перед запуском програми необхідно натиснути кнопку «Animate»  , потім кнопку скидання «Reset» 

і запустити програму кнопкою «Run» 

.

Для перевірки отриманого після виконання програми результату відкриваємо калькулятор Windows: Пуск → Програми → Стандартні → Калькулятор. Переводимо його у інженерний режим (рисунок 1.6). Вмикаємо режим роботи у шістнадцятковій системі числення (перемикач «Hex») та обмежуємо розмір чисел одним байтом (перемикач «1 байт»).

Програму перевіряємо у відповідності до завдання з урахуванням пріоритетів. Результат має співпасти з тим, що виведений у комірку ВПД згідно завдання. У випадку, коли результат не співпадає, необхідно з’ясувати причину помилки. Відлагодження програми у цьому випадку зручно проводити у покроковому режимі (кнопка «Step into»  або F7).


Рисунок 1.6 – Вікно калькулятора Windows.

У цьому разі при кожному натисненні кнопки буде виконуватись тільки одна інструкція на мові асемблер. При цьому у відкритих вікнах з регістрами і вмістом пам'яті буде оновлюватися інформація у відповідності з ходом виконання програми. Порівнюючи результати виконання кроку у вікні «Main Registers» з результатами проміжних обчислень знаходимо помилки у програмі. Якщо під час відлагодження програми з'ясувалося, що алгоритм необхідно скорегувати, то треба перервати процес відлагодження, вибравши пункт меню Debug→Terminate. Після внесення необхідних змін у програму її потрібно повторно компілювати (Project → Build all) і повторити процес відлагодження.

Якщо програма занадто велика і необхідно відлагодити тільки деякі її фрагменти, з метою економії часу, використовують крапки зупинки «Breakpoint», які встановлюються на початку фрагменту, що треба виправити. При цьому перехід до крапки зупинки відбувається натисненням кнопки «Go». Далі відлагодження відбувається у покроковому режимі як було описано раніше.

**Основні правила синтаксису мови Асемблер**

1. Кожна команда пишеться у окремому рядку
2. Кожний рядок розбивається на чотири поля, які утворюють чотири колонки. Укожного поля своє призначення:

- перше поле – поле мітки;
- друге поле – поле команди (мнемокода);
- третє поле – поле параметрів команди;

- четверте поле - поле коментарів команди, заповнювати його необов'язково. Воно може займати місце до кінця сторінки. Для відокремлення коментаря від параметрів команди необхідно ставити знак «;». Информація після «;» не обробляється транслятором, що також зручно використовувати при відлагоджуванні програми для тимчасового виключення з неї команди (досить поставити перед командою «;»).

**Завдання**

Необхідно внести зміни у готову програму додавання вмісту двох комірок у відповідності з індивідуальним завданням (табл. 1.1) та отримати результат у вказаній комірці.

Перевірити правильність роботи зміненої програми шляхом письмового розрахунку у двійковій системі числення.

Таблиця 1.1 – Дані для виконання індивідуального завдання

| № | Завдання | Дані для перевірки | |
|---|---|---|---|
| 1 | (22H)+(24H)→(23H) | A9 | D1 |
| 2 | (21H)+(20H)→(25H) | BC | DA |
| 3 | (23H)+(21H)→(22H) | D0 | A6 |
| 4 | (20H)+(24H)→(25H) | DE | B3 |
| 5 | (21H)+(25H)→(24H) | DD | DC |
| 6 | (23H)+(22H)→(27H) | FA | AB |
| 7 | (27H)+(21H)→(20H) | E1 | D9 |
| 8 | (24H)+(22H)→(26H) | DC | B4 |
| 9 | (25H)+(21H)→(22H) | A5 | B4 |
| 10 | (28H)+(20H)→(27H) | CD | CA |
| 11 | (23H)+(21H)→(22H) | E5 | C2 |
| 12 | (28H)+(27H)→(26H) | DD | BD |
| 13 | (26H)+(25H)→(20H) | B9 | E6 |
| 14 | (21H)+(23H)→(20H) | EC | D4 |
| 15 | (24H)+(23H)→(29H) | C3 | BD |
| 16 | (28H)+(26H)→(25H) | EE | EB |
| 17 | (25H)+(21H)→(23H) | D3 | FB |
| 18 | (22H)+(21H)→(27H) | F1 | AF |
| 19 | (20H)+(25H)→(28H) | B6 | BF |

Продовження таблиці 1.1.

| | | | |
|---|---|---|---|
| 20 | (24H)+(23H)→(29H) | F0 | C8 |
| 21 | (22H)+(21H)→(24H) | C8 | D5 |
| 22 | (23H)+(27H)→(25H) | E0 | C3 |
| 23 | (28H)+(24H)→(20H) | CB | F1 |
| 24 | (25H)+(23H)→(22H) | DB | CF |
| 25 | (26H)+(23H)→(28H) | F3 | DB |
| 26 | (25H)+(26H)→(29H) | EF | E5 |
| 27 | (22H)+(21H)→(27H) | D3 | CC |
| 28 | (25H)+(21H)→(23H) | FE | FC |
| 29 | (21H)+(25H)→(24H) | E5 | DA |
| 30 | (22H)+(23H)→(27H) | C9 | DF |

Приклад письмового розрахунку у двійковій системі числення:

0AAH → 1010 1010B
+
055H → 0101 0101B
---
111111111B → 0FFH

**Зміст звіту з лабораторної роботи**

Зміст з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

**Контрольні запитання**

1. Призначення емулятора.
2. Створення нового файлу.
3. Трансляція тексту програми.
4. Редагування тексту програми.
5. Компіляція програми.
6. Основні правила синтаксису команд Асемблер.
7. Введення даних у комірки пам’яті ОЗП.

# 2. ЛАБОРАТОРНА РОБОТА №2

## Вивчення команд логічних операцій

**Мета:** здобути навички використання команд логічних операцій.

### Короткі теоретичні відомості

Логічні команди (Таблиця А.3 додатку А) реалізують операції булевої алгебри «І» (AND), «АБО» (OR), «АБО, що виключає» (XOR), побітне зрушення вмісту акумулятора ліворуч або праворуч, інверсію та очищення акумулятора, а також перестановку його тетрад.

### Завдання

Написати програму для вирішення індивідуального завдання згідно варіанту (табл. 2.1).

Таблиця 2.1 – Дані для виконання індивідуального завдання

| № | Вираз | Дані для перевірки |
|---|---|---|
| 1. | $(20\text{H}) * 02\text{H} + 8\text{AH} \wedge \overline{(21\text{H})} \forall (23\text{H}) \rightarrow (22\text{H})$ | 12H; 0A2H; 1BH |
| 2. | $\overline{((20\text{H}) - 8\text{AH})} / 02\text{H} \vee 3\text{FH} \forall (21\text{H}) \rightarrow (23\text{H})$ | 0FCH; 88H |
| 3. | $(21\text{H}) \vee (22\text{H}) + \overline{10\text{H}} \forall (23\text{H}) / 02\text{H} \rightarrow (24\text{H})$ | 43H; 56H; 78H |
| 4. | $21\text{H} * (21\text{H}) \forall (22\text{H}) + \overline{10\text{H} \wedge 35\text{H}} \rightarrow (23\text{H})$ | 12H; 53H |
| 5. | $\overline{\overline{(20\text{H})} \wedge (21\text{H})} - 7\text{BH} \forall 38\text{H} * 02\text{H} \rightarrow (22\text{H})$ | 11H; 05H |
| 6. | $(21\text{H}) \vee \overline{(82\text{H} / (22\text{H}) + \overline{\overline{58\text{H}}} \forall (23\text{H}) \rightarrow (24\text{H})}$ | 3BH; 02H; 0CDH |
| 7. | $(22\text{H}) \forall (23\text{H}) - \overline{21\text{H}} * 03\text{H} \wedge 5\text{DH} \rightarrow (24\text{H})$ | 4BH; 58H |
| 8. | $4\text{AH} \forall \overline{05\text{H}} * \overline{(21\text{H}) \vee (22\text{H})} - 15\text{H} \rightarrow (23\text{H})$ | 87H; 23H |
| 9. | $10\text{H} \vee (\overline{(21\text{H}) + (22\text{H}) \forall 0\text{C2H}}) / 02\text{H} \rightarrow (23\text{H})$ | 35H; 2DH |
| 10. | $28\text{H} + (21\text{H}) * \overline{0\text{AAH}} \forall \overline{11\text{H} \vee (22\text{H})} \rightarrow (23\text{H})$ | 03H; 73H |
| 11. | $\overline{(23\text{H})} - 31\text{H} \forall (21\text{H}) \wedge 25\text{H} / 03\text{H} \rightarrow (22\text{H})$ | 30H; 1CH |

Продовження таблиці 2.1

| | | |
|---|---|---|
| 12 | $(21H) \wedge (22H) + \overline{(23H)} \forall 0ACH * 02H \rightarrow (24H)$ | 72H; 0ADH; 12H |
| 13 | $\overline{(((20H) + 21H) * 02H \forall (21H)) \wedge 54H} \rightarrow (22H)$ | 43H; 0CDH |
| 14 | $15H * (20H) + \overline{35H} \forall (21H) \vee (23H) \rightarrow (24H)$ | 03H; FAH; 93H |
| 15 | $25H/02H \wedge \overline{\overline{(21H)} \forall (22H) - 11H} \rightarrow (23H)$ | 32H; 0CCH |
| 16 | $\overline{(10H - 08H} \forall (21H) * 02H) \vee (22H) \rightarrow (23H)$ | 05H; 1FH |
| 17 | $(20H) \forall (\overline{22H}) - 21H \wedge 0ABH / 02H \rightarrow (23H)$ | 2BH; 0FDH |
| 18 | $\overline{(20H) \forall (21H) - 04H} \vee 0CAH / 02H \rightarrow (22H)$ | 3AH; 49H |
| 19 | $\overline{(20H) + 23H} \wedge (22H) * 02H \forall 0CDH \rightarrow (21H)$ | 1AH; 08H |
| 20 | $10H \vee (21H) + \overline{(22H) * 02H \forall 0CDH} \rightarrow (23H)$ | 3FH; 23H |
| 21 | $(22H) - \overline{(21H)} \wedge 7BH / 03H \forall 0FFH \rightarrow (23H)$ | 0F4H; 8EH |
| 22 | $(20H) / 03H + \overline{\overline{(21H)} \forall 12H} \vee 0CEH \rightarrow (22H)$ | 09H; 0EFH |
| 23 | $\overline{(21H) - 0D2H \forall (22H)} \wedge 0FDH * 02H \rightarrow (23H)$ | 7EH; 19H |
| 24 | $(20H) \vee \overline{(21H)} + ABH \forall CEH - \overline{0FFH} \rightarrow (22H)$ | 7CH; 63H |
| 25 | $(\overline{32H - (20H) * 02H}) \wedge (21H) \forall 0CDH \rightarrow (22H)$ | 04H; 0B7H |
| 26 | $(0FFH - (\overline{21H}) \forall 89H) \vee \overline{(22H) / 03H} \rightarrow (23H)$ | 73H; 09H |
| 27 | $14H \forall 0DFH + \overline{\overline{((21H) \wedge 18H)} * 02H} \rightarrow (22H)$ | 34H |
| 28 | $(21H) \vee 0C6H - 03H \forall (22H) / 02H \rightarrow (23H)$ | 0ADH; 12H |
| 29 | $(\overline{10H + (23H) \forall (24H)} * 02H) \wedge 58H \rightarrow (25H)$ | 67H; 0A5H |
| 30 | $((24H) \wedge (\overline{25H}) - 10H) \forall \overline{0DDH / 02H} \rightarrow (26H)$ | 98H; 35H |

Дані для перевірки підставляються послідовно у відповідні комірки внутрішньої пам’яті даних згідно індивідуального завдання.

Приклад:

$$(20H) * \overline{(21H) - 10H} \rightarrow (23H)$$

```
ORG 0H
MOV A, 21H          ;(21H)→(A)
SUBB A, #10H        ;(A)-10H-(C)→(A)
CPL A               ;(Ā)→(A)
MOV B, 20H          ;(20H)→(B)
MUL AB        ;(A)*(B) →(B).(A)
MOV 23H, A          ;(A) →(23H)
L1: JMP L1
END
```

Дані для перевірки слід підставляти у комірки (20H), (21H) і виконувати дії на калькуляторі з урахування пріоритетів виконання дій.

**Зміст звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

**Контрольні запитання**

1. Назвати логічні операції.
2. Логічне множення. Навести приклади команд з різними методами адресації.
3. Логічне додавання. Навести приклади команд з різними методами адресації.
4. Логічне «АБО, що виключає». Навести приклади команд з різними методами адресації.
5. Логічне заперечення (інверсія) та його команда.
6. Маскування.

# 3. ЛАБОРАТОРНА РОБОТА №3

## Вивчення команд роботи з бітами

**Мета:** здобути навички використання команд роботи з бітами.

### Короткі теоретичні відомості

Команди роботи з бітами (Таблиця А.4 додатку А) забезпечують встановлення (запис одиниці), скидання або очищення (запис нуля) та інвертування бітів внутрішньої пам'яті, виконують операції булевої алгебри «І» (AND) та «АБО» (OR) з прапором переносу та бітами внутрішньої пам'яті, здійснюють пересилання між прапором переносу і бітом внутрішньої пам'яті. Адреса біту може бути вказана тільки прямим способом.

### Завдання

Згідно заданої функції алгебри логіки (табл. 3.1) скласти:
- комбінаційну схему;
- електричну схему;
- таблицю істинності для всіх можливих комбінацій бітів порта P2;
- написати програму.

Таблиця 3.1 - Дані для виконання індивідуального завдання

| № | Завдання |
|---|---|
| 1. | $\overline{\overline{P2.1 \vee P2.2} \wedge P2.3} \rightarrow P0.0$ |
| 2. | $\overline{P2.1} \wedge P2.2 \vee \overline{P2.3} \rightarrow P0.0$ |
| 3. | $\overline{\overline{P2.1} \wedge P2.2 \vee P2.3} \rightarrow P0.0$ |
| 4. | $P2.4 \wedge \overline{P2.3 \vee \overline{P2.2}} \rightarrow P0.0$ |
| 5. | $\overline{P2.4} \vee \overline{P2.3} \wedge P2.2 \rightarrow P0.0$ |
| 6. | $\overline{P2.4} \wedge \overline{P2.3 \vee P2.2} \rightarrow P0.0$ |
| 7. | $P2.2 \vee \overline{\overline{P2.3} \wedge P2.4} \rightarrow P0.0$ |

Продовження таблиці 3.1

| | |
|---|---|
| 8. | $\overline{\overline{\overline{\overline{P2.3 \wedge \overline{P2.2} \vee P2.4}}}} \rightarrow P0.0$ |
| 9. | $\overline{\overline{\overline{P2.1} \vee P2.2} \wedge P2.3} \rightarrow P0.0$ |
| 10. | $\overline{P2.0 \vee \overline{\overline{P2.7}} \wedge P2.6} \rightarrow P0.0$ |
| 11. | $\overline{\overline{P2.0} \wedge P2.7 \vee P2.6} \rightarrow P0.0$ |
| 12. | $P2.0 \vee \overline{\overline{P2.7} \wedge P2.6} \rightarrow P0.0$ |
| 13. | $\overline{P2.0 \wedge P2.7 \vee \overline{P2.6}} \rightarrow P0.0$ |
| 14. | $\overline{P2.5 \vee \overline{P2.4 \wedge P2.3}} \rightarrow P0.0$ |
| 15. | $\overline{P2.5} \wedge \overline{P2.4 \vee P2.3} \rightarrow P0.0$ |
| 16. | $\overline{P2.5 \vee P2.3} \wedge \overline{P2.4} \rightarrow P0.0$ |
| 17. | $P2.1 \vee \overline{P2.2} \wedge \overline{P2.3} \rightarrow P0.0$ |
| 18. | $\overline{P2.1} \vee \overline{P2.2} \wedge P2.3 \rightarrow P0.0$ |
| 19. | $\overline{P2.1} \vee P2.2 \wedge \overline{P2.3} \rightarrow P0.0$ |
| 20. | $P2.4 \wedge \overline{P2.2} \vee \overline{P2.3} \rightarrow P0.0$ |
| 21. | $\overline{P2.4} \wedge P2.2 \vee \overline{P2.3} \rightarrow P0.0$ |
| 22. | $\overline{P2.0 \wedge P2.7} \vee \overline{P2.6} \rightarrow P0.0$ |
| 23. | $\overline{P2.0 \wedge P2.7 \vee \overline{\overline{P2.6}}} \rightarrow P0.0$ |
| 24. | $\overline{P2.7 \wedge P2.5 \vee \overline{\overline{P2.3}}} \rightarrow P0.0$ |
| 25. | $P2.7 \wedge \overline{\overline{P2.5} \vee P2.3} \rightarrow P0.0$ |
| 26. | $\overline{P2.6} \vee P2.4 \wedge \overline{P2.2} \rightarrow P0.0$ |
| 27. | $\overline{P2.6 \vee P2.4 \wedge \overline{P2.2}} \rightarrow P0.0$ |
| 28. | $\overline{P2.6 \vee \overline{P2.4} \wedge P2.2} \rightarrow P0.0$ |
| 29. | $\overline{P2.6 \wedge P2.2} \vee \overline{P2.4} \rightarrow P0.0$ |
| 30. | $\overline{\overline{P2.1 \wedge P2.7} \vee P2.3} \rightarrow P0.0$ |

**Зміст звіту з лабораторної роботи**

Зміст з лабораторної роботи повинен містити:
- титульний лист, тему, мету лабораторної роботи;
- умову індивідуального завдання;
- комбінаційну схему;
- електричну схему;
- таблицю істинності;
- текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми;
- висновки з лабораторної роботи.

**Контрольні запитання**

1. Назвати команди пересилання при роботі з бітами.
2. Назвати команди логічних операцій при роботі з бітами.
3. Назвати команди установки та очищення біта.
4. Призначення портів вводу/виводу інформації.
5. Скласти таблицю істиності та комбінаційну схему за функцією алгебри логіки згідно завдання викладача.
6. Пояснити принцип складання електричних схем за функцією алгебри логіки.

# 4. ЛАБОРАТОРНА РОБОТА №4

## Вивчення команд непрямої адресації. Робота із масивами даних

**Мета:** здобути навички використання команд непрямої адресації при роботі з масивами даних.

### Короткі теоретичні відомості

Непряма реєстрова адресація використовується для звертання до комірок внутрішньої пам'яті даних, причому у якості адреси комірки, у якій знаходиться операнд, використовується вміст одного з індексних регістрів R0, R1. Ознакою непрямої регістрової адресації є наявність у команді знаку «@».

### Завдання

Зі всіма елементами початкового масиву виконати вказані дії і сформувати кінцевий масив згідно індивідуального завдання (табл.4.1).

Таблиця 4.1 - Дані для виконання індивідуального завдання

| № | Початкова адреса початкового масиву | Елементи початкового масиву | Дії | Початкова адреса кінцевого масиву |
|---|---|---|---|---|
| 1. | 2DH | E6, BE, BE, C3, FF, DB, D4, C2, BA, 0 | (...+49h)∀68h | 5BH |
| 2. | 36H | 97, AF, AF, B6, F2, C7, AF, C4, AB, F1 | (...+77h)∀49h | 68H |
| 3. | 2FH | B4, DC, DC, D7, 9B, D7, DA, D2, 9A | (...+1Ch)∀97h | 5CH |
| 4. | 2BH | C7, DA, E3, EE, 15, E8, DA, E1, E1, 16 | (...+CDh)∀C2h | 62H |
| 5. | 21H | 7B, 60, 60, 9C, 4A, 63, 55, 64, 48, 9B | (...+BBh)∀77h | 64H |
| 6. | 39H | 14, 2C, 2C, 33, 6F, 26, 33, 32, 2E, 6E | (...+7Ah)∀C9h | 69H |

Продовження таблиці 4.1

| | | | | |
|---|---|---|---|---|
| 7. | 24H | 49, 3D, 34, 43, 3E, 75, 2C, 42, 28, 74 | (... + DCh)∀71h | 6AH |
| 8. | 22H | 2C, 38, 41, 3D, 4B, 41, 7C, 4A, 41, 4C, 41, 3D, 50 | (... + 6Eh)∀CAh | 59H |
| 9. | 34H | 8C, A9, A5, 9D, 60, A9, AF, 60, A3, AA, 9D, AE | (... + 46h)∀86h | 62H |
| 10. | 29H | 7D, 5F, 53, 5B, 20, 57, 71, 20, 55, 6E, 5B, 72 | (... + 6Dh)∀ADh | 6FH |
| 11. | 3DH | 0, E5, D6, D9, 9C, E5, CB, 9C, D7, DF, DF, D8 | (... + 7Ah)∀36h | 69H |
| 12. | 39H | 57, 65, 78, 74, 70, 66, B3, 76, 72, 70, 78, B3, 67, 65, 68, 78 | (... + 45h)∀D8h | 6AH |
| 13. | 2DH | 71, 9F, 85, DA, 99, 8C, 95, DA, 9B, 9E, 95, 88, 95, 8C, D9 | (... + 1Bh)∀D5h | 65H |
| 14. | 34H | 81, 93, 9D, CA, 89, 98, 8D, CA, 96, 9D, 87, 8F, A1, C9 | (... + 99h)∀43h | 69H |
| 15. | 34H | A1, CB, C5, 18, D9, C6, D5, 18, C3, D1, CA, CA, D5, C6, 19 | (... + E6h)∀DEh | 5EH |
| 16. | 28H | C, 27, 25, 2D, 16, 31, 1A, 67 | (... + 8Ch)∀D2h | 66H |
| 17. | 23H | 92, 84, 84, 3F, 68, 7E, 74, 3F, 7B, 80, 73, 84, 71 | (... + D9h)∀38h | 6EH |
| 18. | 2FH | 3C, 6, 20, D5, 23, 10, 14, 11, D5, 8, 1C, D5, 8, C, 7, 11, 22 | (... + BAh)∀AFh | 70H |
| 19. | 26H | 1D, E1, E8, FD, A9, E0, FA, A9, FB, E0, EE, E1, FD, A8 | (... + 60h)∀29h | 68H |

Продовження таблиці 4.1

| | | | | |
|---|---|---|---|---|
| 20. | 3AH | DE, AA, B3, BE, 72, AB, C1, 72, AC, AB, A4, AF, 73 | (...+9Ch)∀2Eh | 6DH |
| 21. | 20H | 4A, 3C, 33, 36, 32, 34, 3C, 77, 4F, 32, 34, 3C, 78 | (...+63h)∀FAh | 5FH |
| 22. | 23H | FE, F0, E9, EA, E6, E8, F0, 2D, E1, E6, 2D, 10, FD, C, 13, 24 | (...+AEh)∀FBh | 58H |
| 23. | 2FH | E0, D2, C9, D4, C8, CA, D2, 95, C1, C8, 95, DB, E7, E1, E2 | (...+89h)∀3Eh | 61H |
| 24. | 40H | 59, C2, 81, 87, 94, 85, 85, C2, 97, 79, 96, 7A, C2, 89, 7F, 95 | (...+A7h)∀49h | 60H |
| 25. | 35H | 25, 84, 47, 39, 34, 84, 53, 4A, 84, 47, 53, 4A, 4A, 49, 49 | (...+6Ch)∀D0h | 55H |
| 26. | 25H | AB, 4C, 8D, 97, 9C, 4C, 91, 8A, 4C, 4C, 98, 87, 87 | (...+B9h)∀25h | 68H |
| 27. | 23H | BD, E1, EE, 29, DC, DE, E7, 29, DC, E1, E2, E7, EE, DC | (...+CFh)∀D8h | 6BH |
| 28. | 2BH | FC, D5, C4, C4, CD, 14, FE, D9, C7, 14, ED, D9, D5, C2, 15 | (...+3Eh)∀72h | 60H |
| 29. | 28H | 22, 36, 27, 27, 2E, 75, 18, 3D, 27, 3E, 28, 29, 42, 36, 28, 76 | (...+7Bh)∀D0h | 5DH |
| 30. | 32H | 4F, 36, 71, 30, 32, 3F, 36, 33, 46, 2D, 72 | (...+59h)∀EAh | 58H |
| 31. | 3DH | 2C, 59, 59, 48, 4F, 59, 54, 4E, 4F, 8D, 5D, 51, 48, 4C, 5A, 48, 8C | (...+DAh)∀47h | 6EH |

**Зміст звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

**Контрольні запитання**

1. Дати визначення методу непрямої адресації.
2. Навести приклади команд непрямої адресації.
3. Пояснити доцільність застосування непрямої адресації.
4. Дати визначення поняттю «масив даних».
5. Команди пересилання даних з використанням непрямої адресації.
6. Команди арифметичних операцій з використанням непрямої адресації.
7. Команди логічних операцій з використанням непрямої адресації.
8. Команди передачі керування з використанням непрямої адресації.

# 5. ЛАБОРАТОРНА РОБОТА №5

## Паралельне введення-виведення інформації

**Мета:** ознайомитися з емулятором семисегментного індикатора та особливостями його взаємодії з програмним забезпеченням Franklin Software.

### Короткі теоретичні відомості

Семисегментний індикатор (ССІ) - це один з простіших індикаторів, який використовується для виведення інформації з мікроконтролера (МК). Він підключається до паралельного порту таким чином, що кожному сегменту відповідає певний біт порту (рисунок 5.1).

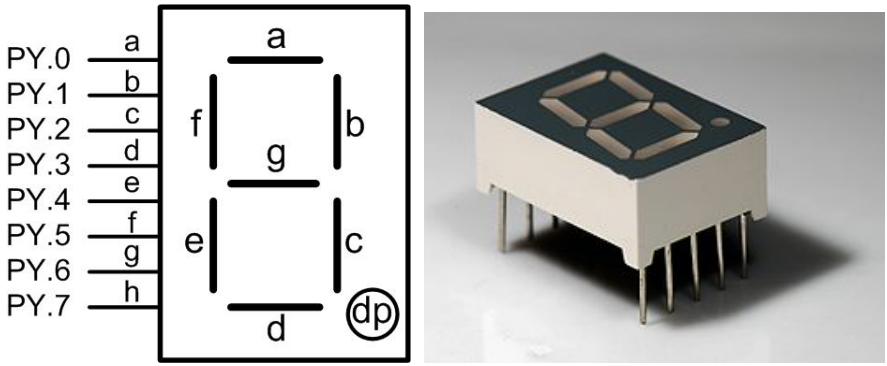


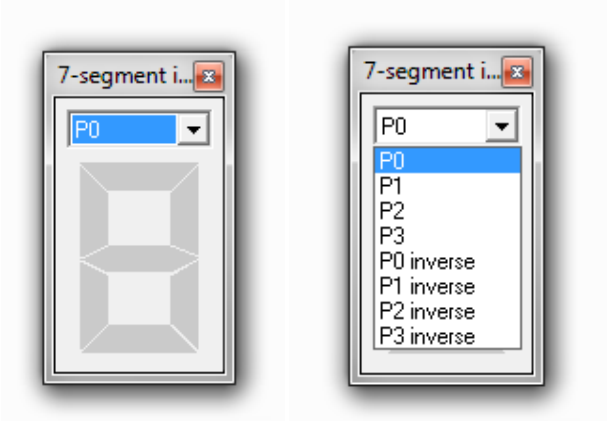

Рисунок 5.1 - Семисегментий індикатор

Наприклад, для того, щоб на семисегментному індикаторі з'явилась цифра «5», треба в один із чотирьох портів МК вивести двійкове число «01101101b» (таблиця 5.1).

Таблиця 5.1 – 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 |

Для сполучення Proview Franklin Software з емулятором семисегментного індикатора треба запустити файл 7segment_1.exe. Після цього відкриється вікно, зовнішній вигляд якого показано на рисунку 5.2, а. Зі списку, що відкривається вибирається потрібний порт (рисунок 5.2, б).




а б

а – зовнішній вигляд; б – список можливих для використання портів

Рисунок 5.2 – Вікно емулятора CCI

Після запуску програми індикації, буде здійснено виведення необхідних знаків на індикатор, при цьому сегменти будуть змінювати колір.

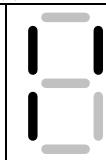
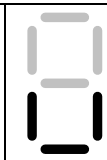
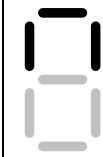
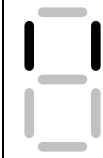
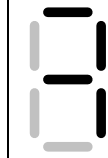
**Завдання**

Написати програму для індикації знаків на семисегментному індикаторі згідно варіанта індивідуального завдання (таблиця 5.2), використавши для цього порти P0 і P1.

Таблиця 5.2 – Дані для виконання індивідуального завдання

| № | Символи | | № | Символи | | № | Символи | |
|---|---|---|---|---|---|---|---|---|
| | P0 | P1 | | P0 | P1 | | P0 | P1 |
| 1. |  |  | 11. |  |  | 21. |  |  |
| 2. |  |  | 12. |  |  | 22. |  |  |
| 3. |  |  | 13. |  |  | 23. |  |  |
| 4. |  |  | 14. |  |  | 24. |  |  |
| 5. |  |  | 15. |  |  | 25. |  |  |
| 6. |  |  | 16. |  |  | 26. |  |  |

Продовження таблиці 5.2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7. |  |  | 17. |  |  | 27. |  |  |
| 8. |  |  | 18. |  |  | 28. |  |  |
| 9. |  |  | 19. |  |  | 29. |  |  |
| 10. |  |  | 20. |  |  | 30. |  |  |

**Зміст звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди, рисунок, отриманий шляхом Print Screen з результатами лабораторної роботи, висновки з лабораторної роботи.

**Контрольні запитання**

1. Призначення семисегментного індикатора.
2. Підключення семисегментного індикатора.
3. Процедура виведення інформації на семисегментний індикатор.

# 6. ЛАБОРАТОРНА РОБОТА №6

## Організація часових затримок програмним методом

**Мета:** ознайомитися з принципом роботи портів МК родини х51, навчитися робити часові затримки програмним методом.

**Короткі теоретичні відомості.**

**Порти введення-виведення інформації**

Всі чотири порти МК51 призначені для введення або виведення інформації побайтно. Порти 1 і 2 мають приблизно таку ж структуру, як і порт 3. Кожний порт містить керовані регістр-фіксатор, вхідний буфер і вхідний драйвер.

Вихідні драйвери портів 0 і 2, а також вхідний буфер порта 0 використовуються при звертанні до зовнішньої пам'яті (ЗП). При цьому через порт 0 в режимі часового мультиплексування спочатку виводиться молодший байт даних. Через порт 2 вводиться старший байт адреси у тих випадках, коли розрядність адреси дорівнює 16 біт.

Всі виводи порта 3 можуть бути використані для реалізації альтернативних функцій (таблиця 6.1), які можуть були задіяні шляхом запису у відповідні біти регістра-фіксатора (Р3.0-Р3.7) порта 3

Порт 0 є двонапрямленим, а порти 1, 2 і 3 – квазідвонапрямленими. Кожна лінія портів може бути використана незалежно для введення або виведення інформації. Для того, щоб деяка лінія порта використовувалася для введення, в в D-тригер регістра-фіксатора порта повинна бути записана 1, яка закриває МОП-транзистор вихідного кола.

За сигналом RESET (RST) у регістри-фіксатори всіх портів автоматично записуються одиниці, які настроюють їх тим самим на режим введення.

Всі порти можуть бути використані для організації введення-виведення інформації за двонапрямленими лініями передачі. Однак порти 0 і 2 не можуть бути використані з цією метою у випадку, якщо МК-система має зовнішню пам'ять, зв'язок з якою організується через загальну розділену шину адреси/даних, яка працює у режимі часового мультиплексування.

**Запис у порт**

При виконанні команди, яка змінює вміст регістра-фіксатора порта, нове значення фіксується у регістрі у момент S6P2 останнього циклу команди. Однак опитування вмісту регістра-фіксатора вихідною схемою здійснюється під час фази P1 і новий вміст регістра-фіксатора з'явиться на вихідних контактах порта тільки в момент S1P1 наступного машинного цикла.

Таблиця 6.1 - Альтернативні функції порта 3

| Символ | Позиція | Ім'я і призначення |
|---|---|---|
| RD | P3.7 | Читання. Активний сигнал низького рівня формується апаратурно при звертанні до зовнішньої пам'яті даних |
| WR | P3.6 | Запис. Активний сигнал низького рівня формується апаратурно при звертанні 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 |
| TXD | P3.1 | Вихід передавача послідовного порта в режимі УАПП Вихід синхронізації в режимі регістра зсуву |
| RXD | P3.0 | Вхід приймача послідовного порта в режимі УАПП. Введення/виведення данних в режимі регістра зсуву. |

**Особливості роботи портів**

Звертання до портів введення/виведення можливе з використанням команд, які оперують з байтом, окремим бітом і вільною комбінацією бітів. При цьому у тих випадках, коли порт є одночасно операндом і місцем призначення результата, пристрій керування автоматично реалізує спеціальний режим, який називається

«читання-модифікація-запис». Цей режим звертання передбачає введення сигналів не з зовнішніх виводів порта, а з його регістра-фіксатора, що дозволяє виключити неправильне зчитування раніше виведеної інформації.

Подібний механізм звертання до портів реалізовано у таких командах:

ANL – логічне І, наприклад ANL P1 ,A;

ORL – логічне АБО, наприклад ORL P2,A;

XRL – виключне АБО, наприклад XRL P3,A;

JBC – перехід, якщо в адресованому біті одиниця, і наступне скидання біта, наприклад JBC P1.1, LABEL;

CPL – інверсія біта, наприклад CPL P3.3;

INC – інкремент порта, наприклад INC P2;

DEC – декремент порта, наприклад DEC P2;

DJNZ – декремент порта і перехід, якщо його вміст не дорівнює нулю, наприклад DJNZ P3, LABEL;

MOV PX.Y, C – передача біта переносу в біт Y порту X;

SET PX.Y – встановлення біта Y порта X;

CLR PX.Y – скидання біта Y порта може порту X.

Зовсім не очевидно, що останні три команди у наведеному списку є командами «читання-модифікація-запис». Однак це саме так. За цими командами спочатку считується байт із порта може порту, а потім записується новий байт у регістр-фіксатор.

**Опитування двійкового давача**

У приладах і системах логічного керування об’єктами події у об’єкті керування фіксується з використанням різноманітних цифрових давачів типу так/ні, наприклад, кінцевих вимикачів, які підімкнені до МК. Схема двійкового давача наведена на рисунку 6.1.

**Очікування статичного сигнала**

Типова процедура очікування події (WAIT) складається з таких дій: введення сигнала від давача, аналізу значення сигналу і передачі керування у залежності від стану давача. На рисунку 6.2 представлена блок-схема алгоритма процедури очікування події замиканням контакта двійкового давача. Конкретна програмна реалізація

процедури залежить не тільки від типу МК, але і від того, яким чином давач підключено до МК. Він може бути підключений до однієї з ліній портів МК або до спеціальних тестованих входів (T0, T1 для МК51).

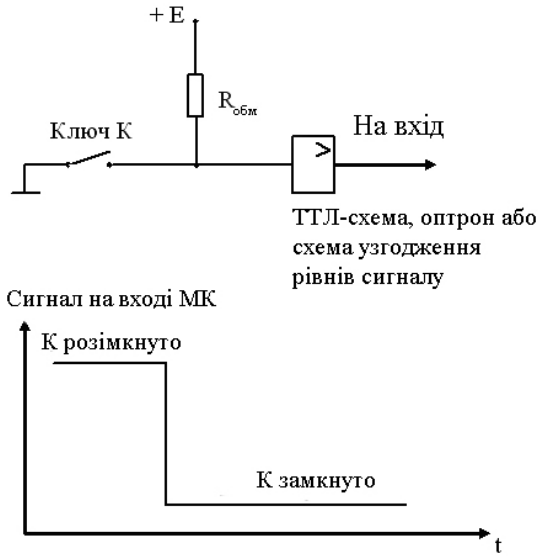



Рисунок 6.1 - Схема двійкового давача


Рисунок 6.2 – Структурно-алгоритмічна схема процедури очікування події

Далі наведений приклад програмної реалізації описаного алгоритму:

```
WAITC:JB P1.3,WAITC    ;Очікування   замикання
                       ;контакта
;Аналогічна реалізація у випадку розмикання
WAITC:JNB P1.3, WAITC  ;Очікування   розмикання
                       ;контакта
```

**Реалізація часових затримок програмним методом**

Програмна затримка реалізується шляхом виконання певної кількості команд, за умови, що час виконання кожної команди відомий. Наприклад, для процесорів 8051 при тактовій частоті 12 МГц час виконання команди додавання складає один обчислювальний такт і дорівнює 1 мкс.

**Часова затримка малої тривалості**

Для виконання дій, що повторюються, (у нашому випадку це певна кількість команд) зручно використовувати цикли. Наприклад у наступному програмному фрагменті

```
DELAY:          MOV R2, #X
COUNT:          DJNZ R2, COUNT
                RET
```

Число #X, яке записується у регістр R2, визначає скільки разів буде виконана команда DJNZ, що у свою чергу, впливає на загальний час виконня фрагменту. Знайдемо значення X, яке забезпечує затримку тривалістю 100 мкс.

Виконання кожної з команд CALL, MOV, DJNZ, RET становить два машинних цикла (тобто 2 мкс при тактовій частоті 12 МГц). Загальний час виконання фрагменту становить 2+2+2X+2 [мкс], тобто 6+2X [мкс]. Для бажаної затримки, що дорівнює 100 мкс X=(100-6)/2, тобто X=47. Аналогічно занаходяться значення X для затримок іншої тривалості. При цьому, якщо отримане значення є дробовим, його округляють до цілого, що вносить невелику похибку.

Розглянутий вище фрагмент є готовою підпрограмою часової затримки і може бути виконаний у будь-якому місці основної програми за допомогою команди CALL DELAY.

**Часова затримка великої тривалості**

Оскільки розмір регістра = 1 Байт, то час такої затримки не може перевищити 2+2+255*2+2=516 мкс. Якщо у тіло цикла додати два порожніх оператора NOP, час буде дорівнювати t=2+2+4X+2=6+4X. При X=249 час затримки складе 1 мс. Тоді затримку в 1 секунду отримаємо, викликавши 250 разів затримку в 4 мс (рисунок 6.3)

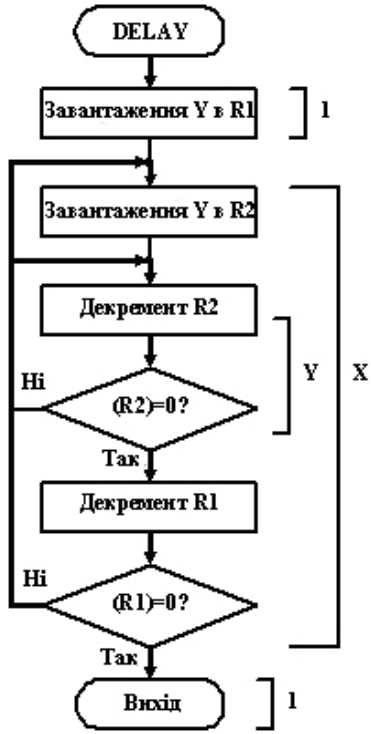



Рисунок 6.3 - Структурно-алгоритмічна схема формування часової затримки

```
DELAY:      MOV R2, #249
COUNT:      NOP
            NOP
            DJNZ R2, COUNT
            RET

ONESEC:     MOV R1, #250
LOOP:       CALL DELAY      ;1 мс
            CALL DELAY      ;+1мс
            CALL DELAY      ;+1мс
            CALL DELAY      ;+1мс=4мс
            DJNZ R1, LOOP   ;250 циклів
                            ;затримки по 4 мс
                            ;складає одну
                            ;секунду
```

**Завдання**

Для апаратної частини, що зображена на рисунку 6.4 написати програму для реалізації наступних дій,:при натисканні кнопки SB1 здійснити індикацію послідовності знаків на семисегментному індикаторі згідно варіанта індивідуального завдання (таблиця 6.2). Індикацію здійснити у циклі. Між індикацією послідовності знаків організувати затримку за часом.

**Зміст звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

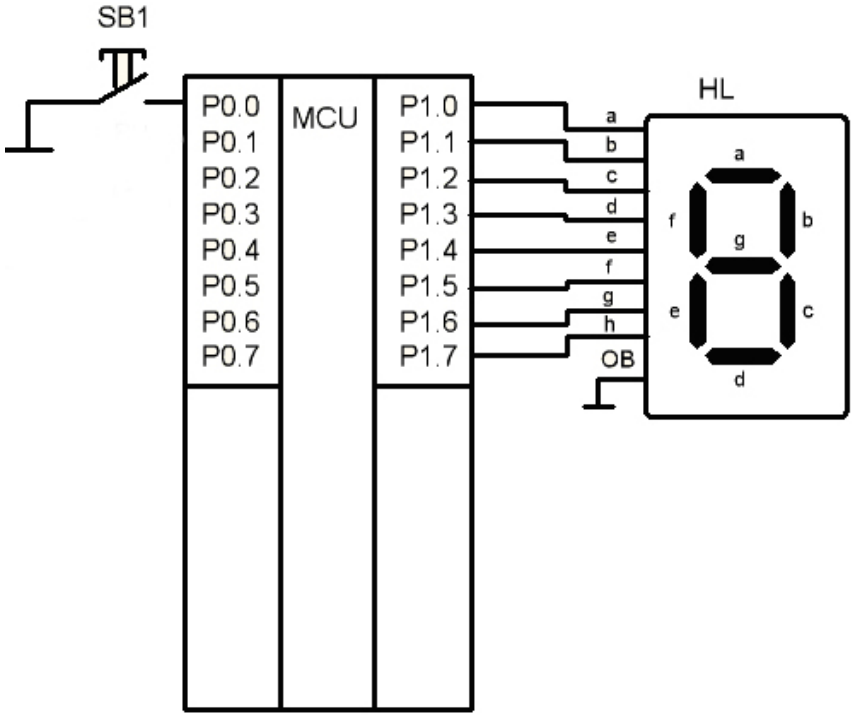



Рисунок 6.4 – Апаратна частина завдання

**Контрольні запитання**

1. Принцип роботи портів МК родини x51.
2. Запис у порт. Альтернативні функції порта.
3. Особливості роботи портів.
4. Опитування двійкового давача. Очікування статичного сигналу.
5. Принцип роботи семисегментних індикаторів.
6. Формування часових затримок малої тривалості (до 100 мкс) програмним методом.
7. Формування часової затримки великої тривалості (до декількох секунд) програмним методом.

Таблиця 6.2 – Дані для виконання індивідуального завдання

| № варіанта | Послідовність знаків | Затримка за часом |
|---|---|---|
| 1 | 13579 | 10 мкс |
| 2 | 02468 | 20 мкс |
| 3 | LABEL | 30 мкс |
| 4 | GLOBAL | 40 мкс |
| 5 | FLOPPY | 50 мкс |
| 6 | 12345 | 10 мкс |
| 7 | 67890 | 20 мкс |
| 8 | COFFEE | 30 мкс |
| 9 | APPLE | 40 мкс |
| 10 | JUICE | 50 мкс |
| 11 | GLOBAL | 10 мкс |
| 12 | FLOPPY | 20 мкс |
| 13 | COFFEE | 30 мкс |
| 14 | 02468 | 40 мкс |
| 15 | 13579 | 50 мкс |
| 16 | JUICE | 10 мкс |
| 17 | APPLE | 20 мкс |
| 18 | LABEL | 30 мкс |
| 19 | 67890 | 40 мкс |
| 20 | 12345 | 50 мкс |
| 21 | HOBBY | 10 мкс |
| 22 | GLASS | 20 мкс |
| 23 | CLOUD | 30 мкс |
| 24 | 98765 | 40 мкс |
| 25 | 43210 | 50 мкс |
| 26 | CLOUD | 10 мкс |
| 27 | 98765 | 20 мкс |
| 28 | 43210 | 30 мкс |
| 29 | HOBBY | 40 мкс |
| 30 | GLASS | 50 мкс |

# 7. ЛАБОРАТОРНА РОБОТА № 7

## Система переривань

**Мета:** навчитися використовувати систему переривань МК сімейства x51 для взаємодії з периферійними пристроями.

### Короткі теоретичні відомості

### Система переривань МК-51

Система переривань МК дозволяє реагувати на зовнішні (наприклад, від датчика) і внутрішні (від вбудованого таймера) події, що позбавляє від необхідності регулярно опитувати датчики, на що витрачалося чимало ресурсів, а також виконувати ітерації під час відліку часової затримки.

Спрощена схема переривань МК51 показана на рис. 7.1

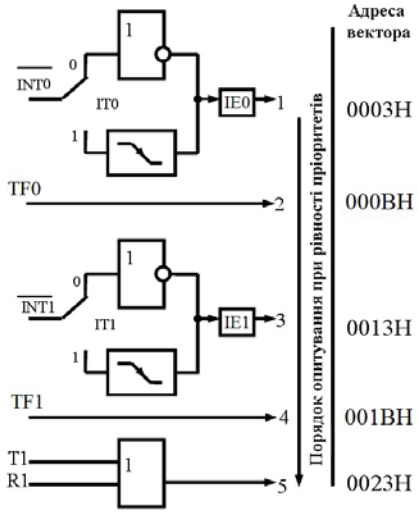



Рисунок 7.1 - Система переривань МК-51

Зовнішні переривання INT0 і INT1 можуть бути викликані або рівнем, або переходом сигналу з 1 в 0 на входах МК51 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 підпрограми обслуговування переривання. Скидання цих прапорів виконується апаратурно тільки в тому випадку, якщо переривання було викликано по переходу (зрізу) сигналу.

Якщо ж переривання викликане рівнем вхідного сигналу, то скиданням прапора IE управляє 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 УАПП апаратурно, але скидатися повинні програмою. Переривання можуть бути викликані або скасовані програмою, тому що всі перераховані прапори програмно-доступні і можуть бути встановлені / скинуті програмою з тим же результатом, що і якби вони були б встановлені / скинуті апаратними засобами.

У блоці регістрів спеціальних функцій є два регістри (таблиці 7.1 і 7.2), призначених для керування режимом переривань і рівнями пріоритету. Можливість програмної установки / скидання будь-якого керуючого біта в цих двох регістрах робить систему переривань МК51 виключно гнучкою.

Для користувача обробка переривань схожа з викликом підпрограм (CALL), але для повернення з переривання використовується команда RETI, а не RET. Слід звернути увагу на те, що у векторі адрес переривань під кожне переривання виділено мало пам’яті, раціонально в перериванні робити безумовний перехід на вільну область пам’яті, де можна розмістити підпрограму переривань, а потім в кінці цієї підпрограми зробити повернення з переривання.

Наприклад:

```
ORG 13H
JMP LABEL1
  ;Команди основної програми «main»
LABEL1:
  ;Код, який необхідно обробити по перериванню
RETI ;Повернення з переривання
```

Таблиця 7.1 - Регістр масок переривання (РМП)

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| EA | IE.7 | Розблокування переривань. Скидається програмно для заборони всіх переривань незалежно від станів IE4-IE0 |
| - | IE.6 | Не використовується |
| - | IE.5 | Не використовується |
| ES | IE.4 | Біт дозволу переривання від УАПП. Установка / скидання програмою для дозволу / заборони переривань від прапорів T1 або R1 |
| 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 | Біт дозволу переривання від таймера 0. працює аналогічно IE.3 |
| EX1 | IE.0 | Біт дозволу зовнішнього переривання 0. Працює аналогічно IE.2 |

Таблиця 7.2 - Регістр пріоритетів переривань

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| - | IP.7-IP.5 | Не використовуються |
| PS | IP.4 | Біт пріоритету УАПП. Установка / скидання програмою для присвоювання перериванню від УАПП вищого / нижчого пріоритету |
| PT1 | IP.3 | Біт пріоритету таймера 1. Установка / скидання програмою для присвоювання перериванню від таймера 1 вищого / нижчого пріоритету |
| PX1 | IP.2 | Біт пріоритету зовнішнього переривання 1. Установка / скидання програмою для присвоювання вищого / нижчого пріоритету зовнішньому перериванню INT1 |
| PT0 | IP.1 | Біт пріоритету таймера 0. Працює аналогічно IP.3 |
| PX0 | IP.0 | Біт пріоритету зовнішнього переривання 0. Працює аналогічно IP.2 |

Приклад: При натисканні кнопки, яка підключена до входу переривання INT1, вивести на ССІ, що підключений до Р1 знак «4», а у випадку натискання кнопки, яка підключена до входу переривання INT0, вимкнути всі сегменти ССІ.

```
      ORG 0
      JMP MAIN          ;Перейти на основну
                        ;програму
      ORG 003           ;Вектор переривання по INT0
      JMP INTR0         ;Перехід з області адрес
                        ;векторів переривань в п/п
                        ;переривання від INT0
      ORG 013H          ;Вектор переривання по INT1
      JMP INTR1         ;Перехід з області адрес
                        ;векторів переривань в п/п
                        ;переривання від INT1
MAIN: SETB IE.2         ;Дозволити преривання по INT1
      SETB IE.0         ;Дозволити преривання по INT0
      SETB TCON.0       ;Переривання INT0 по зрізу
                        ;сигналу
      SETB TCON.2       ;Переривання INT1 по зрізу
                        ;сигналу
      MOV P1, # 0H      ;Погасити індикацію
      SETB IE.7         ;Розблокувати всі
                        ;дозволені переривання
L1:   JMP L1            ;Зациклення
INTR0: MOV P1, # 0H     ;Погасити індикацію
      RETI              ;Повернення з підпрограми
                        ;переривання
INTR1: MOV P1, #01100110B;Виведення знака "4" у Р1
      RETI              ;Повернення з п/п
                        ;переривання
      END               ;Кінець тексту програми
```

**Завдання**

Для схеми (рисунок 7.2) написати програму для шістнадцяткового реверсивного лічильника (0 ... F). Початковий стан індикатора - «0». При натисканні кнопки «+» значення на індикаторі збільшуються, після натискання кнопки «-» - зменшуються. Лічильник - круговий, тобто при збільшенні після «F» йде «0», при зменшенні після «0» йде «F».

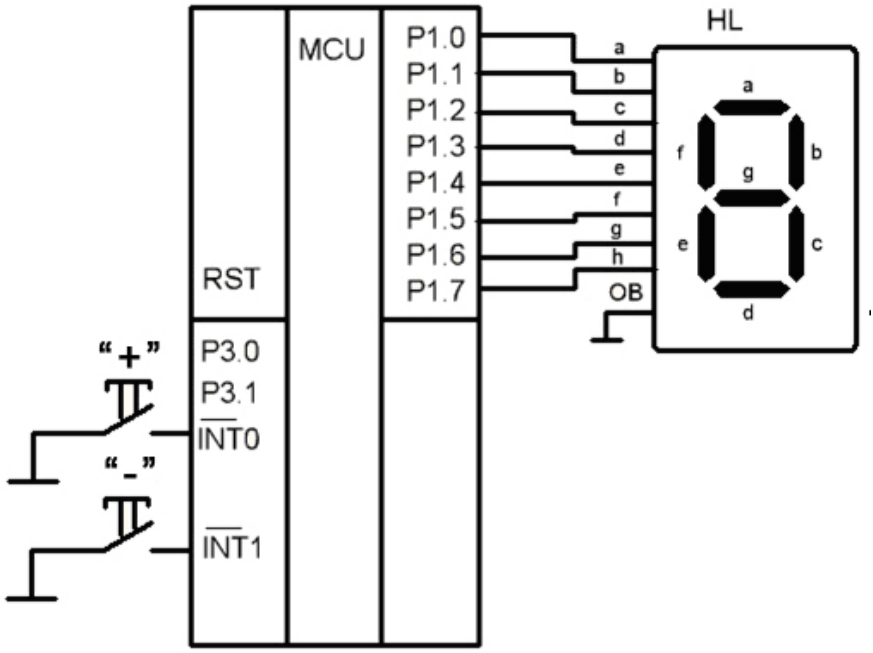



Рисунок 7.2 – Схема з'єднання МК з органами керування та індикатором

**Вимоги до звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання одної з умовний частин всієї програми, висновки з лабораторної роботи.

**Контрольні запитання**

1. Робота з перериваннями. Поняття вектора переривання.
2. Джерела переривання (чим воно може бути викликане).
3. Регістр масок переривання.
4. Регістр пріоритетів переривань.
5. Програмна реалізація обробки переривання (порівняння викликів підпрограм).
6. Механізм обробки переривань (програмно апаратна послідовність дій).
7. Зміна пріоритетів переривань.

# 8 ЛАБОРАТОРНА РАБОТА №8

## Апаратний таймер-лічильник

**Мета:** навчитись застосовувати апаратный таймер-лічильник.

### 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 8.1). 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.

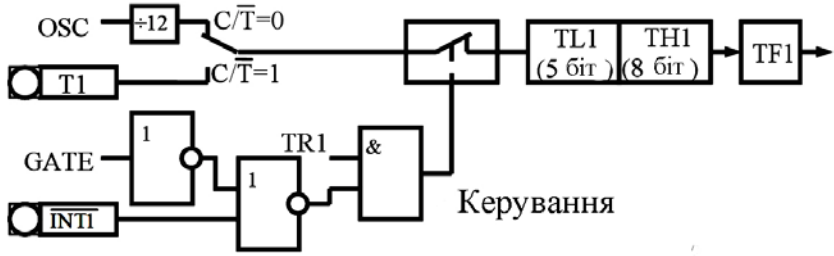



Рисунок 8.1 – Структура таймера/лічильника в режимі 0

**Режим 1**

Робота будь-якого Т/Л в режимі 1 такаж сама, як і в режимі 0, із-за того, що таймерний регістр (таблиця 8.1) має розрядність 16 біт.

**Режим 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 8.2).

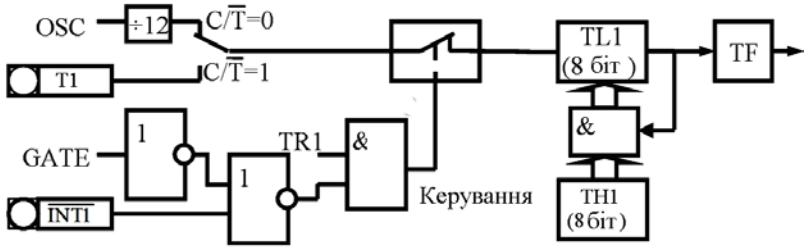



Рисунок 8.2 – Структура таймера / лічільника в режимі 2

Таблиця 8.1 – 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. |

Продовження таблиці 8.1

| | | |
|---|---|---|
| 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     ;настройка 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     ;настройка 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 8.3) й засвітити через секунду світлодіод VD1, що підключений до P0.0 (затримку реалізувати за допомогою таймера).

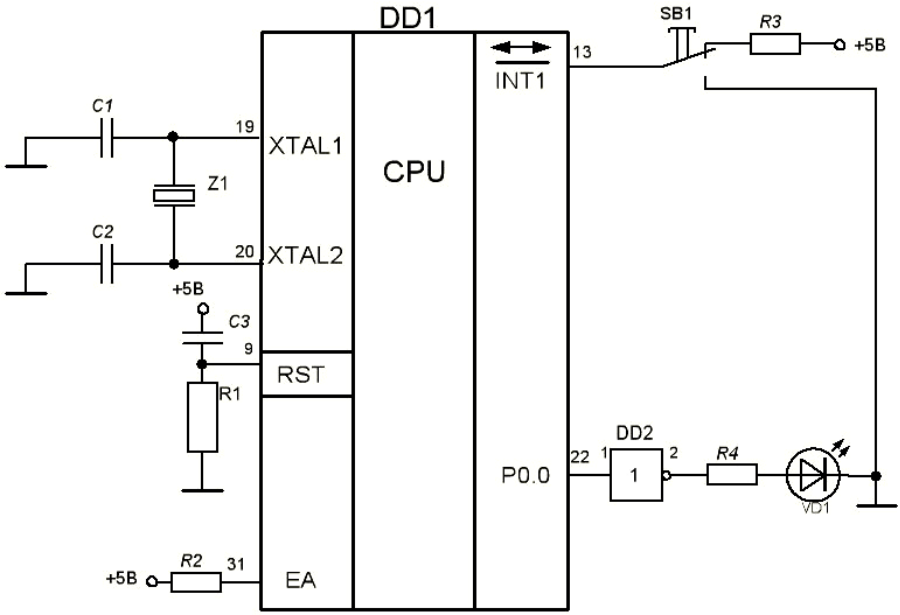



Рисунок 8.3 – Схема з’єднання МК з органами керування та світлодіодом

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);настройка
      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
```

Зверніть увагу, як налаштовується 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)   ;настройка на час
                             ;250мкс
      MOV TH0, #NOT(250-1)   ;настройка 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
```

**Завдання**

Написати програму для реалізації секундоміра (00…59). Відповідно до рисунку 8.4 Початковий стан індикаторів – «00». 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.

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.

**Вимоги до звіту з лабораторної роботи**

Звіт з лабораторної робоиі повинен містити титульный лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програмы з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

**Контрольні запитання**

1. Режими роботи таймера-лічильника.
2. Регістр керування таймера.
3. Регістр статуса таймера.
4. Частота змінювання вмісту таймера/лічильника при роботі в режимі таймера.
5. Формування часової затримки за допомогою таймера
6. Організація часової затримки: програмно й за допомогою таймера (порівняльний аналіз).
7. Максимальна тривалість часової затримки, що реалізується за допомогою таймера (для режимів 0, 1, 2).

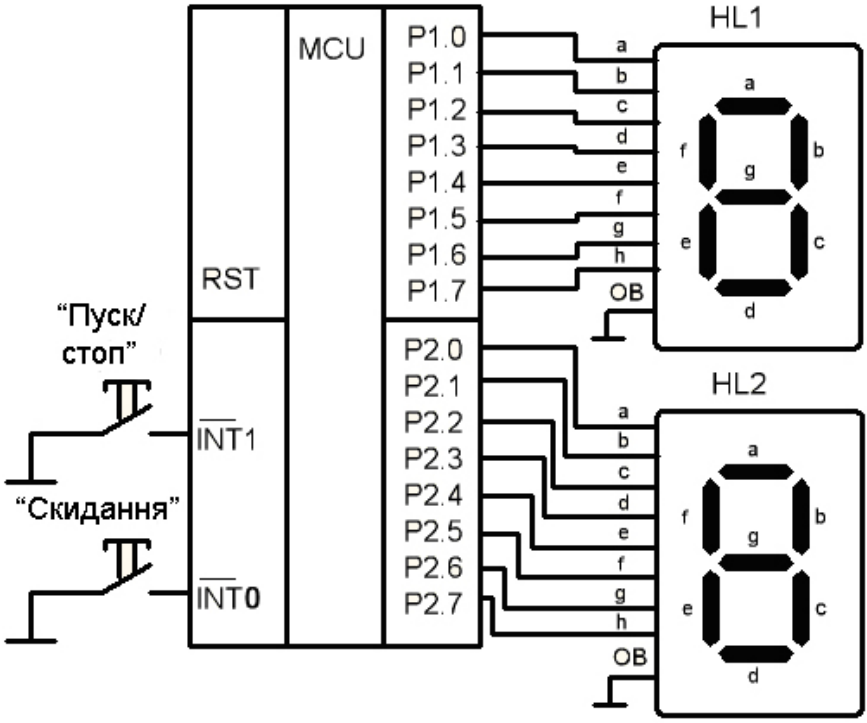



Рисунок 8.4 – Схема з’єднання МК з органами керування та індикаторами

# 9 ЛАБОРАТОРНА РОБОТА №9

## Послідовне пересилання інформації. Регістр зсуву.

**Мета:** ознайомитися з способом послідовного пересилання інформації, навчитися заносити інформацію в регістр зсуву шляхом змінювання стану його вхідних сигналів.

### Короткі теоретичні відомості

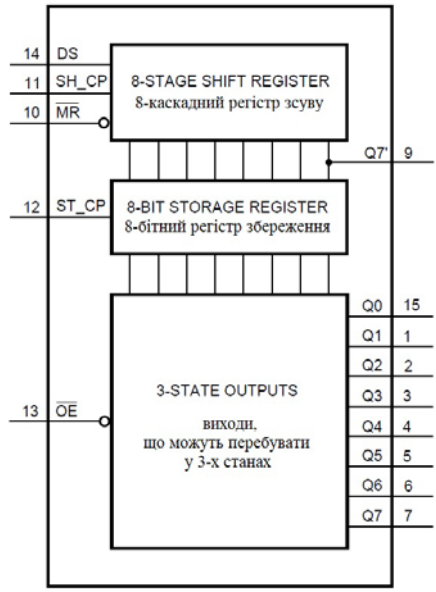
В мікропроцесорних системах передача інформації може здійснюватися паралельно або послідовно.

При паралельній передачі інформація одночасно йде декількома (зазвичай 8, 16, 32, 64) паралельними лініями зв'язку. Характерними особливостями паралельної передачі інформації є: висока швидкість передачі даних; коротка довжина шин; високі витрати на реалізацію з'єднань з огляду на велику кількість ліній зв'язку.

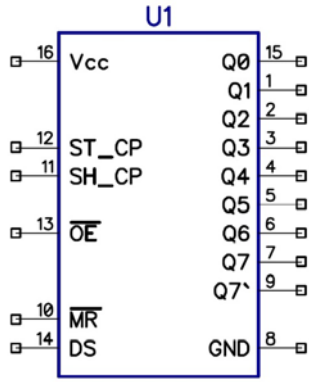
При послідовному пересиланні інформація біт за бітом йде однією лінією зв'язку. Характерними особливостями послідовної передачі інформації є: більш низька швидкість у порівнянні з паралельної передачею такої самої тактової частоти; менші витрати на реалізацію з'єднань з огляду на малу кількість ліній зв'язку; більша їх протяжність, яка може коливатися в межах від декількох десятків метрів до декількох кілометрів; використання паралельно-послідовних та послідовно-паралельних перетворювачів даних, оскільки інформація в передавачеві та приймачеві, як правило, обробляється в паралельній формі.

Одним з різновидів послідовно-паралельних перетворювачів є мікросхема (МС) 74HC595 (рисунок 9.1), яка містить в собі 8-каскадний послідовний регістр зсуву та регістр зберігання, виходи якого можуть перебувати у 3-х станах («0», «1». «від'єднаний»).

Зсувний регістр і регістр зберігання мають окремі входи для тактових імпульсів (SH_CP, ST_CP). Дані зміщуються при додатному переході (від «0» до «1») входу SH_CP. Вміст регістра зсуву передається в регістр зберігання при додатному переході входу ST_CP. Якщо обидві тактові входи з'єднані між собою, регістр зсуву завжди буде на один тактовий імпульс попереду регістра зберігання.




а)




б)

а - структурна схема; б - позначення мікросхеми 74HC595 на принципових схемах.

Рисунок 9.1 – Мікросхема 74HC595

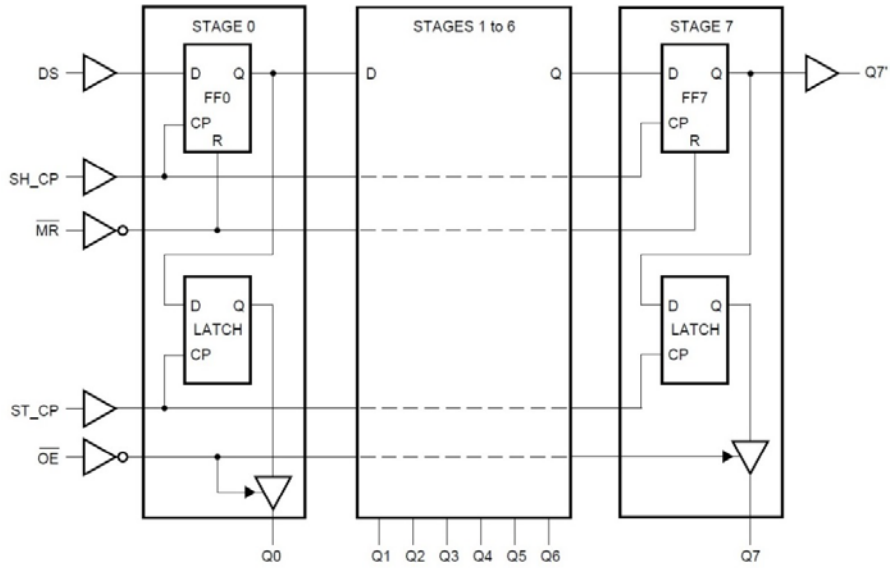



Рисунок 9.2 – Логіка функціонування мікросхеми 74HC595

Регістр зсуву має послідовний вхід (DS) і послідовний стандартний вихід (Q7') для каскадного з'єднання. Також є вхід асинхронного скидання (активний рівень - «низький») для всіх 8-ми каскадів регістра зсуву.

Регістр зберігання має 8 паралельних виходів з драйверами шини, що можуть перебувати у 3-х станах. Дані, які знаходяться в регістрі зберігання з'являються на виході завжди, якщо рівень входу дозволу виходу (OE) «низький».

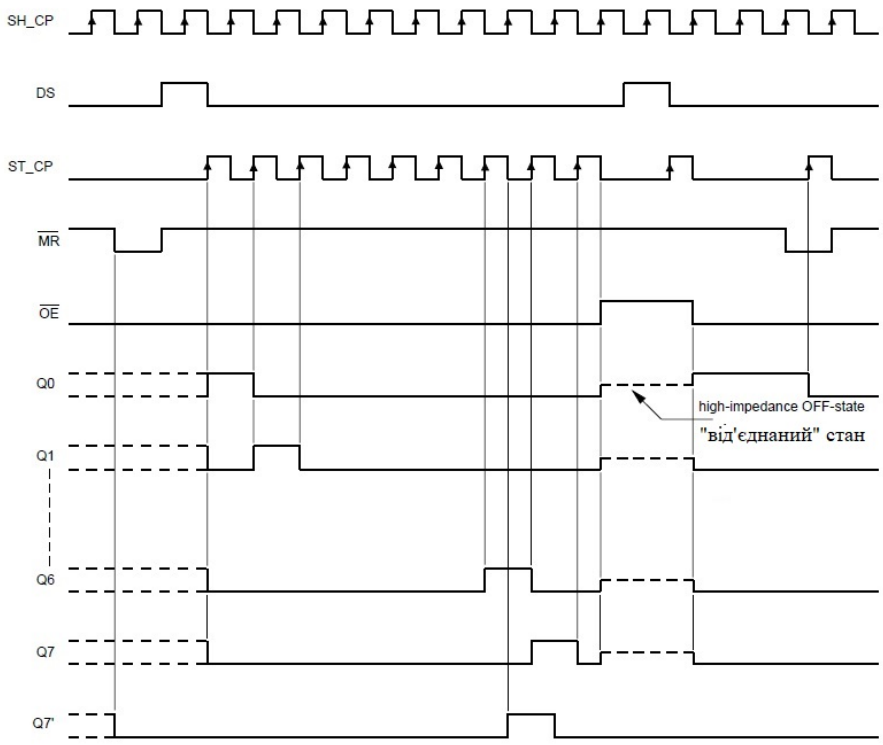



Рисунок 9.3 – Графіки вхідних і вихідних сигналів мікросхеми 74HC595

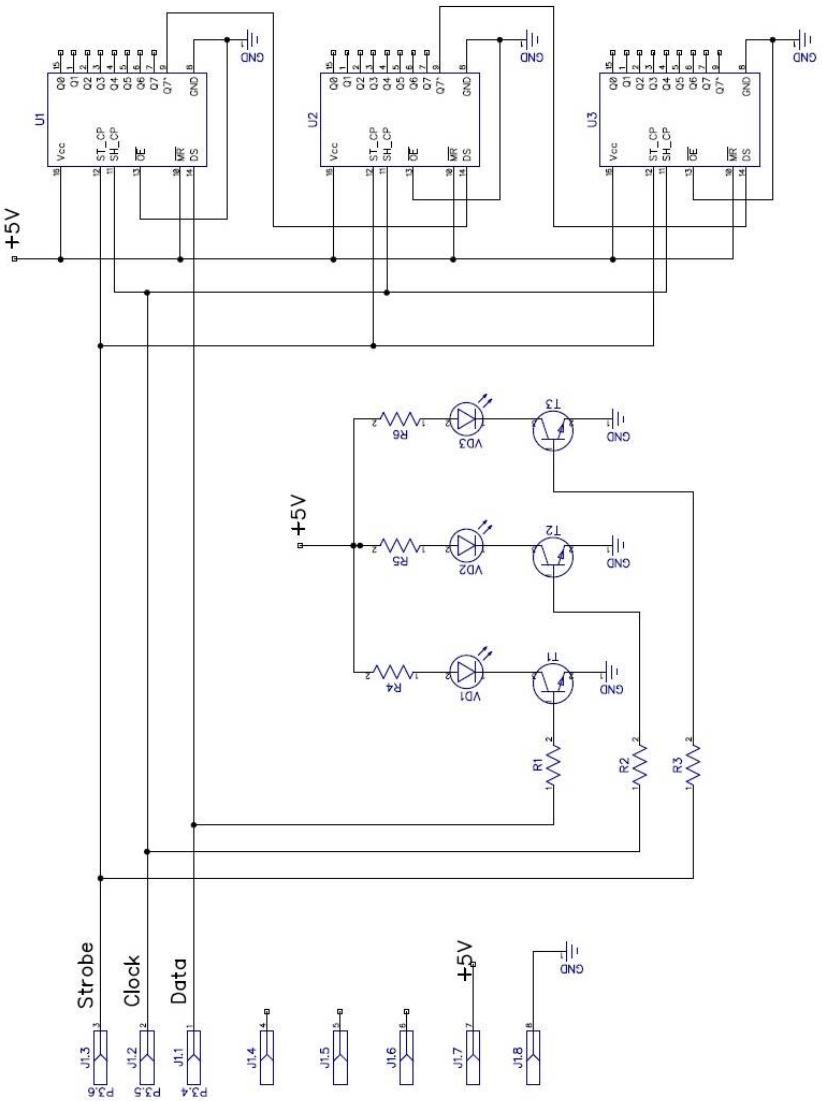



Рисунок 9.4 – Регістри зсуву (фрагмент електричної принципової схеми модуля індикації)

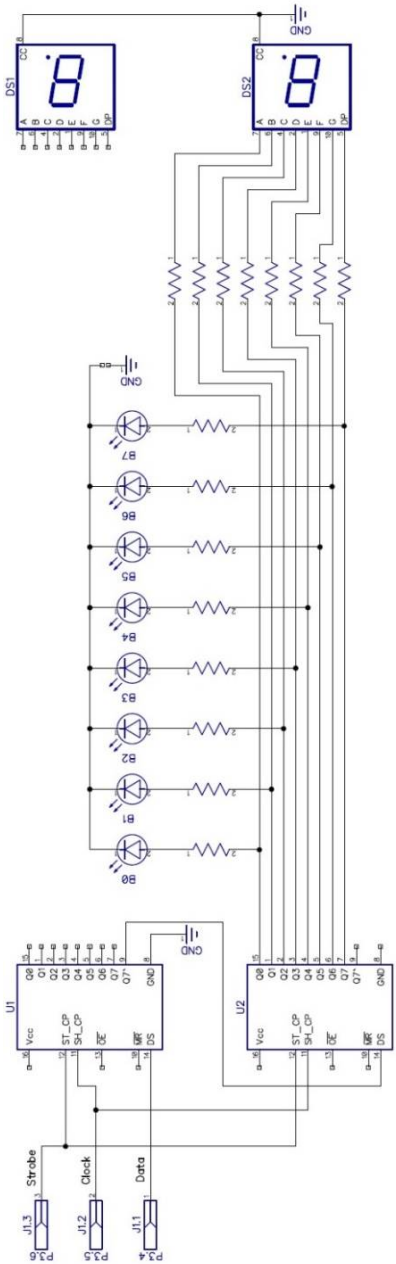



Рисунок 9.5 – Під’єднання семи-сегментних індикаторів до регістрів зсуву (фрагмент електричної принципової схеми модуля індикації)


а) б)

а - умовне зображення; б - зовнішній вигляд.

Рисунок 9.6 – Семисегментний індикатор

На часовій діаграмі (рисунок 9.7), показано як, при змінюванні стану входів SH_CP, DS та ST_CP мікросхеми, можна встановлювати на її виходах Q0 ... Q7 значення 01001111, яке відповідає символу «3» на семи-сегментному індикаторі.

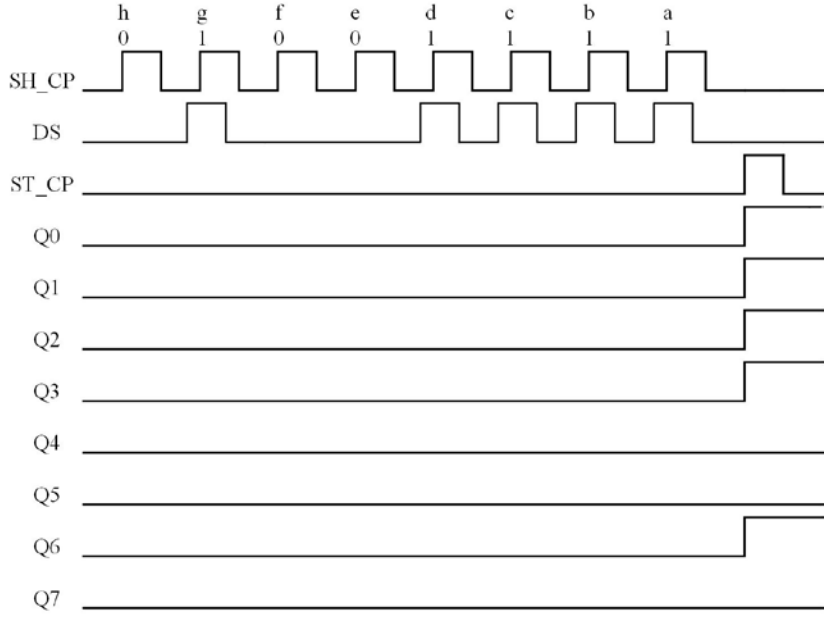



Рисунок 9.7 – Часова діаграма вхідних і вихідних сигналів регістру зсуву

**Виконання лабораторної роботи**

Перед початком виконання лабораторної роботи переконатися, що пульт, модуль індикації та з'єднувальні кабелі не мають механічних пошкоджень.

Спочатку з'єднати пульт з модулем індикації (рисунок 9.8), потім, за допомогою USB-кабелю – з комп'ютером.

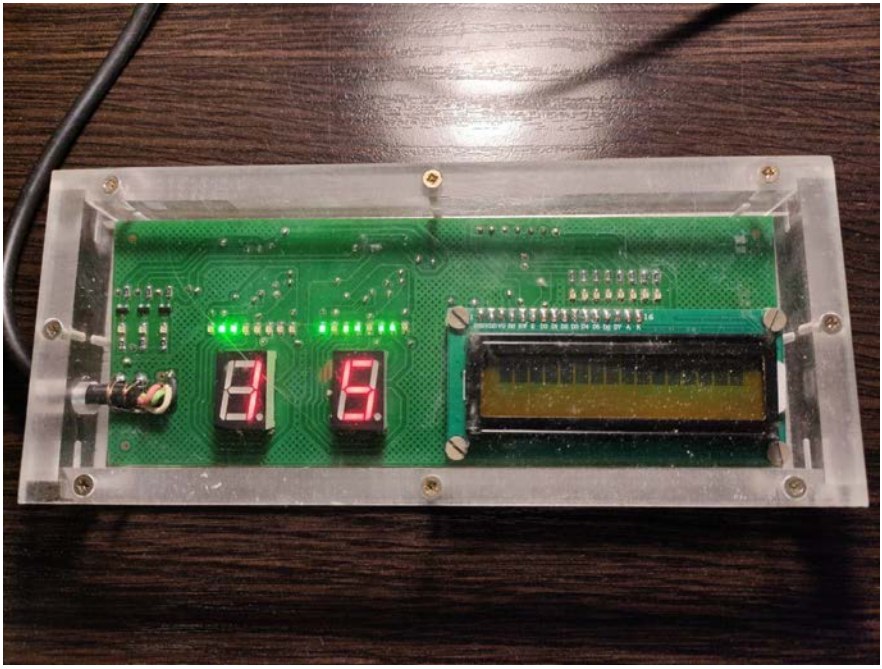


Рисунок 9.8 - Зовнішній вигляд модуля індикації

У середовищі Franclin Software, використовуючи текст програми, що наведений далі, згенерувати HEX-файл.

Положення перемикачів SB1, SB2, SB3 (порт 2) відповідно визначає; логічні рівні сигналів DS, SH_SP, ST_SP.

```
D_S     equ     p3.4    ; дані
SH_SP   equ     p3.5    ; зсув
ST_SP   equ     p3.6    ; фіксація
```

```
        org 0h
                mov     p2, #0FFh
        m1:
                mov c, p2.0
                cpl c
                mov D_S, c
                mov c, p2.1
                cpl c
                mov SH_SP, c
                mov c, p2.2
                cpl c
                mov ST_SP, c
                call wait
                jmp m1
        wait:
                mov r5,#2h
                mov r4,#0ffh
                mov r3,#0ffh
        w1:
                djnz r3,w1
                djnz r4,w1
                djnz r5,w1
                ret
END
```

За допомогою програми WSD (Windows Serial Downloader) завантажити отриманий HEX-файл і запустити програму. Змінюючи положення перемикачів SA1, SA2 та SA3, що відповідають сигналам SH_CP, DS та ST_CP, відтворити послідовність показаних на часовій діаграмі вхідних сигналів регістру зсуву. Переконатися, що на семи-сегментному індикаторі отримано символ «3».

З метою закріплення навичок, вибрати, на власний розсуд, два символи і вивести їх на індикатори.

**Завдання**

Згідно варіанту вивести на індикатори символи, наведені у таблиці 5.2 (лаб. роб. №5 с. 20-21).

**Зміст звіту з лабораторної роботи**

Звіт з лабораторної роботи повинен містити титульний лист; мету лабораторної роботи; текст програми; часову діаграму, що відповідає завданню; фото результату; висновки.

**Контрольні запитання**

1. Як відбувається паралельне передавання інформації?
2. Переваги і недоліки паралельного передавання інформації.
3. Як відбувається послідовне передавання інформації?
4. Переваги і недоліки послідовного передавання інформації.
5. Які пристрої застосовуються для перетворення інформації, що передається послідовно, до паралельного виду?
6. Основні структурні елементи мікросхеми 74HC595.
7. Призначення входів DS, SH_CP, ST_CP, OE.
8. Алгоритм підпрограми **wait** та її призначення в основній програмі.

По скільки імпульсів треба подати на входи DS, SH_CP, ST_CP щоб гарантовано отримати на семи-сегментних індикаторах «десять».

# ПЕРЕЛІК ПОСИЛАНЬ

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

2. Мікропроцесорна техніка: Електронний підручник / В.Я. Жуйков, Т.О. Терещенко, Ю.С. Ямненко, А.В. Заграничний ; відп. ред. О.В. Борисов. – К. : НТУУ «КПІ», 2016. – 440 с.

3. Мікропроцесорна техніка: Навчальний посібник з дисципліни для всіх форм навчання та студентів іноземців напряму підготовки 6.050701 “Електротехніка та електротехнології”/ Укл. В.В.Кирик. – К.: ІВЦ «Видавництво «Політехніка», 2014. – 183с.

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

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

6. Хіхловська, І.В. Обчислювальна техніка та мікропроцесори. Підручник. – [2-ге вид.]. / І.В. Хіхловська, О.С. Антонов – Одеса: Одеська національна академія зв’язку ім. О.С. Попова, 2011. – 440 с.

7. Nazarova, O. Software and Hardware Complex for The Study of Electropneumatic Mechatronic Systems / O. Nazarova, V. Osadchyy, S. Shulzhenko, M. Olieinikov // 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), Kremenchuk, Ukraine, 2022, pp. 1-6, doi: 10.1109/MEES58014.2022.10005698.

8. Назарова, О.С. Використання програмно-апаратного комплексу електропневматичних мехатронних систем при роботі зі здобувачами вищої освіти / О.С. Назарова, В.В. Осадчий, М.О. Олєйніков, С.С. Шульженко // Мехатронні системи : інновації та інжиніринг : тези доповідей VI Міжнародної наук.-практ. конф., 24 листопада 2022 р. - Київ : КНУТД, 2022. – С. 35-36.

9. Новацький А. О. Мікропроцесорні та мікроконтролерні системи : підручник. У 2 ч. Ч. 1. Мікропроцесорні системи [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 16,7 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2020. – 361с.

10. Ельперін, І.В. Автоматизація виробничих процесів [Текст]: підручник / І.В. Ельперін, О.М. Пупена, В.М. Сідлецький, С.М. Швед. – Вид. 2-ге, виправлене. – К.: Вид. Ліра-К, 2015. – 378 с.

11. Osadchyy, V. Adjustable Vibration Exciter Based on Unbalanced Mo-tors / V. Osadchyy, O. Nazarova, T. Hutsol, S. Glowacki, K. Mudryk, A. Bry´s, A. Rud, W. Tulej, M. Sojak // Sensors, 2023. – Vol. 23. – P. 2170. https://doi.org/10.3390/s23042170

12. Невлюдов І. Ш. Виробничі процеси та обладнання об’єктів автоматизації : Підручник для студентів вищих навчальних закладів. – Кривий Ріг: Криворізький коледж НАУ, 2017. – 444 с.

13. Гончаренко Б. М., Осадчий С. І., Віхрова Л. Г. Автоматизація виробничих процесів: навч. посіб. – Кіровоград: Лисенко В.Ф., 2016. – 352 с.

14. Островерхов М.Я. Електротехнічні системи на основі електромагнітних виконавчих пристроїв для керування параметрами технологічних процесів: монографія. – К.: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2017. – 348 с.

Додаток А

## Перелік команд мікроконтролера Intel 8051

Таблиця А.1 - Група команд пересилання даних

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

Продовження таблиці А.1

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

Таблиця А.2 - Команди арифметичних операцій

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

Продовження таблиці А.2

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

Продовження таблиці А.2

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

Таблиця А.3 - Команди логічних операцій

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

Продовження таблиці А.3

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

Продовження таблиці А.3

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

Таблиця А.4 - Команди роботи з бітами

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

Продовження таблиці А.4

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

Таблиця А.5 - Команди передачі керування

| Назва команди | Мнемокод | Т | Б | Ц | Операція |
|---|---|---|---|---|---|
| Довгий перехід | LJMP | 12 | 3 | 2 | (PC)← dir 16 |
| Абсолютний перехід всередині сторінки у 2 Кбайта | AJMP dir11 | 6 | 2 | 2 | (PC)←(PC)+2<br>($PC_{0..10}$)← dir 11 |
| Короткий відносний перехід всередині сторінки у 256 байт | SJMP rel | 5 | 2 | 2 | (PC)←(PC)+2<br>(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,<br>якщо (A)=0, то<br>(PC)← (PC)+rel |

Продовження таблиці А.5

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

Продовження таблиці А.5

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

Продовження таблиці А.5

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

Додаток Б

**Представлення даних у двійковій, шістнадцятковій та десятковій системі числення**

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

| BIN (двійкова) | | | | HEX (шістнад-цяткова) | 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 |