

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

**МЕТОДИЧНІ ВКАЗІВКИ**  
до лабораторних робіт  
із дисципліни «Мікропроцесорна техніка»  
для студентів спеціальності  
171 "Електроніка"  
денної форми навчання

Суми  
Сумський державний університет  
2017

Методичні вказівки до лабораторних робіт із дисципліни  
«Мікропроцесорна техніка» / укладачі І.А. Кулик, В.В. Гриненко. –  
Суми : Сумський державний університет, 2017. – 66 с.

Кафедра електроніки і комп'ютерної техніки

## Зміст

С.

### **Лабораторна робота 1**

ПРОЕКТУВАННЯ КОМБІНАЦІЙНИХ ЦИФРОВИХ  
ПРИСТРОЇВ У ЗАДАНОМУ БАЗИСІ ЛОГІЧНИХ ЕЛЕМЕНТІВ.....4

### **Лабораторна робота 2**

ПРОЕКТУВАННЯ ЛІЧИЛЬНИКІВ ІЗ ЗАДАНИМ  
ПОРЯДКОМ ЗМІНИ СТАНІВ.....8

### **Лабораторна робота 3**

ВИВЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРОГРАМИ ISIS  
PROTEUS. ЗАПИС І ВИКОНАННЯ ПРОСТИХ ПРОГРАМ..... 15

### **Лабораторна робота 4**

ОРГАНІЗАЦІЯ РОБОТИ ЗІ СТЕКОМ. РОБОТА З БІТОВИМ  
ПРОСТОРОМ. ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ..... 29

### **Лабораторна робота 5**

ВВЕДЕННЯ-ВИВЕДЕННЯ, МАСКУВАННЯ ДАНИХ І  
ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ.....37

### **Лабораторна робота 6**

СИСТЕМА ПЕРЕРИВАНЬ. ОБРОБКА ЧАСОВИХ СИГНАЛІВ.  
ПРОГРАМУВАННЯ ПОСЛІДОВНОГО ПОРТУ ОЕОМ.....42

Додаток А.....53

## Лабораторна робота 1

### ПРОЕКТУВАННЯ КОМБІНАЦІЙНИХ ЦИФРОВИХ ПРИБОРІВ У ЗАДАНОМУ БАЗИСІ ЛОГІЧНИХ ЕЛЕМЕНТІВ

**Мета роботи** – вивчення методів проектування цифрових пристроїв комбінаційного типу в заданому функціонально повному наборі логічних елементів (ЛЕ):

- основному функціонально повному наборі (ОФПН), що включає ЛЕ «І», «АБО», «НІ»;

- монофункціональних наборах ЛЕ, «І-НІ» або «АБО-НІ».

#### 1 Теоретичні основи лабораторної роботи

Комбінаційним цифровим пристроєм (КЦП) називається пристрій, вихідні сигнали якого в певний момент часу роботи однозначно визначаються лише сигналами, що діють у той самий момент часу на його входах. У КЦП відсутні елементи пам'яті, тому вихідні сигнали таких пристроїв формуються і зберігаються тільки у період дії вхідних.

КЦП застосовуються для виконання цілого ряду логічних та арифметичних перетворень над вхідними сигналами і використовуються як шифратори, дешифратори, суматори, мультиплексори та інші функціональні вузли.

У загальному випадку спроектоване КЦП може бути представлено у вигляді чорного ящика (ЧЯ), що має  $n$  входів і  $m$  виходів. Єдине, що спочатку відомо про цей ЧЯ, це необхідний алгоритм його функціонування, тобто характер зв'язку між вхідними впливами і вихідними сигналами (реакціями). Проектування зводиться до визначення оптимальної (у певному сенсі) структури (схеми) КЦП (ЧЯ), реалізованої у заданому базисі ЛЕ. Іншими словами, проектування КЦП зводиться до знаходження схеми КЦП, що задовольняє необхідний алгоритм функціонування при двох таких обмеженнях: по-перше, схема КЦП повинна бути реалізована за допомогою ЛЕ заданого функціонального повного набору, по-друге, оскільки необхідний алгоритм функціонування в загальному випадку може бути реалізований за допомогою різних схем, то повинна бути обрана

деяка, у певному сенсі оптимальна схема, наприклад, схема, що відрізняється мінімумом апаратурних витрат, тобто мінімальним числом ЛЕ.

Процес проектування КЦП у загальному випадку включає такі етапи:

1 Словесний опис алгоритму функціонування КЦП, тобто опис роботи пристрою в понятійній формі (на звичайній мові).

2 Оцінка розмірності задачі і вирішення питання про проектування КЦП у цілому або по частинах, чому передуює поділ (умовне) КЦП на складові частини. В окремих випадках для зниження трудомісткості і громіздкості завдання проектування КЦП розбивається на ряд більш простих пристроїв (вузлів), які в сукупності реалізують потрібний алгоритм функціонування та їх проектування не становить особливих труднощів.

3 Перехід від словесного до формалізованого завданням алгоритму функціонування КЦП за допомогою логічних (булевих) функцій.

4 Мінімізація логічних функцій.

5 Перетворення мінімальних форм логічних функцій до виду поданого набору ЛЕ, які складають функціонально повний набір.

6 Побудова схеми КЦП за отриманими (етапи 1-5) логічними функціями.

### **Проектування схеми КЦП у заданому базисі ЛЕ**

Кожна з елементарних логічних функцій, що утворюють ОФПН, може бути відтворена (реалізована) відповідними ЛЕ: інверторами (*НИ*), диз'юнкторами (*АБО*) і кон'юнкторами (*І*), що утворюють ОФПН ЛЕ.

Аналогічно можуть бути реалізовані функції монофункціональних наборів: функції штрих Шеффера – за допомогою ЛЕ «І-НИ», функції стрілка Пірса – за допомогою ЛЕ «АБО-НИ», що утворюють монофункціональні набори ЛЕ.

Основою для проектування КЦП в ОФПН ЛЕ та в монобазисних наборах ЛЕ служать мінімальні форми логічної функції – МДНФ або МКНФ.

## 2 Завдання до лабораторної роботи

2.1 Для кожного КЦП, передбаченого завданням (див. табл. 1):

2.1.1 Скласти таблицю істинності.

2.1.2 Скласти логічні вираження функцій, що реалізуються КЦП, представлені в ДДНФ і ДКНФ.

2.1.3 Мінімізувати за можливості отримані вирази, тобто отримати вирази для МДНФ та МКНФ, використовуючи метод карт Карно.

2.1.4 Перетворити отримані в п. 2.1.3. МДНФ вирази до вигляду, що можуть бути реалізовані в монофункціональному базисі ЛЕ «*I-NI*».

2.1.5 Скласти схему КЦП, використовуючи:

а) ЛЕ ОФПН;

б) монофункціональний набір ЛЕ «*I-NI*».

2.1.6 Зібрати схеми і перевірити правильність їх функціонування.

Таблиця 1 – Варіанти завдань

| №  | Вхідне слово                          | Вихідне слово                         |
|----|---------------------------------------|---------------------------------------|
| 1  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 4,12,6,13,2,8,10,5,0,14,11,3,7,15,9,1 |
| 2  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 7,5,12,9,8,0,4,11,3,13,1,6,14,2,10,15 |
| 3  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 3,12,0,5,9,4,6,11,7,10,2,13,14,1,15,8 |
| 4  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 5,10,2,0,11,15,4,3,8,1,9,12,7,13,14,6 |
| 5  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 9,2,4,10,8,3,0,11,6,12,7,14,1,15,5,13 |
| 6  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 2,10,4,8,12,6,3,13,14,1,11,5,7,15,0,9 |
| 7  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 5,9,15,2,8,3,10,4,11,0,1,12,6,13,7,14 |
| 8  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 4,9,14,0,6,2,8,11,7,10,3,5,15,1,12,13 |
| 9  | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 8,10,2,15,9,0,6,4,14,7,11,3,5,12,1,13 |
| 10 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 | 3,9,10,5,7,14,0,12,8,15,2,11,1,4,13,6 |

### **3 Зміст звіту**

Для кожного спроектованого і дослідженого відповідно до завдання КЦП повинні бути наведені:

3.1 Таблиця істинності та логічні вирази функції, що реалізують КЦП, представлений в ДДНФ і ДКНФ.

3.2 Карти Карно, що відображають хід мінімізації логічних функцій.

3.3 Схеми КЦП, реалізовані в ОФПН ЛЕ і монофункціональному наборі ЛЕ «І-НІ».

3.4 Висновки.

### **4 Контрольні питання**

1 Основні аксіоми та закони алгебри логіки.

2 Диз'юнктивні і кон'юнктивні нормальні форми представлення функцій.

3 Мінімізація логічних функцій за допомогою карт Карно.

4 Поняття функціонального повного набору (ФПН). Приклади ФПН.

5 Послідовність (алгоритм) приведення МДНФ до вигляду, що реалізується в монофункціональному наборі ЛЕ.

6 Реалізувати в монофункціональному наборі ЛЕ «І-НІ» логічні функції: інверсія, диз'юнкція трьох змінних, кон'юнкція трьох змінних.

7 Реалізувати в монофункціональному наборі ЛЕ «АБО-НІ» логічні функції: інверсія, диз'юнкція трьох змінних, кон'юнкція трьох змінних.

## Лабораторна робота 2

### ПРОЕКТУВАННЯ ЛІЧИЛЬНИКІВ ІЗ ЗАДАНИМ ПОРЯДКОМ ЗМІНИ СТАНІВ

**Мета роботи** – вивчення методів проектування цифрових приладів з пам'яттю.

#### 1 Теоретичні основи лабораторної роботи

Цифровий лічильник імпульсів – це цифровий вузол, який здійснює підрахунок імпульсів, що надходять на його вхід. Результат підрахунку формується лічильником у заданому коді і може зберігатися необхідний час. Лічильники будуються на тригерах, при цьому кількість імпульсів, яку може підрахувати лічильник, визначається із виразу

$$N = 2^n - 1,$$

де  $n$  – число тригерів, а мінус один, тому що в цифровій техніці за початок відліку береться 0.

Лічильники бувають підсумовуючі, коли рахунок йде на збільшення, та віднімаючі – рахунок на зменшення.

Якщо лічильник може переключатися в процесі роботи із підсумовуючого на віднімаючий і навпаки, то він називається реверсивним.

#### Тригери

Тригером називають послідовну схему з позитивним зворотним зв'язком і двома стійкими станами 0 і 1 (тобто тригер має властивість пам'яті). У загальному випадку тригер може мати асинхронні входи попередньої установки, тактовий або синхронні та інформаційні входи.

До основних типів тригерів відносять:

- тригер із роздільною установкою станів (*RS*-тригер);
- тригер «затримки» (*D* - тригер);
- універсальний тригер (*JK* - тригер);
- тригер із лічильним входом (*T* - тригер).

За способом запису інформації тригери поділяються на асинхронні та синхронні, або тактові, а за способом управ-



лiння – на тригери iз статичним управлiнням (одиничним або рiдше нульовим рiвнем тактового сигналу) i тригери з динамiчним управлiнням (позитивним – iз 0 в 1, або негативним – iз 1 в 0 – фронтом тактового сигналу). В останньому випадку говорять про тригери iз прямим або iнверсним динамiчним входом управлiння.

### RS-тригер

Функцiональне позначення тригера наведено на рисунку 1.

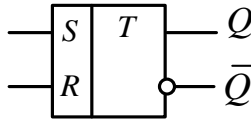


Рисунок 1 – Функцiональне позначення RS-тригера

Таблиця станiв RS-тригера у скороченiй формi включає два вхiдних сигнали (сигнали  $R$  i  $S$ ) i один вихiдний сигнал  $Q$  у даний часовий iнтервал  $Q^t$  та наступний –  $Q^{t+1}$  (див. табл. 2).

Таблиця 2 – Таблиця станiв RS-тригера

| $R$ | $S$ | $Q^t$ | $Q^{t+1}$ |
|-----|-----|-------|-----------|
| 0   | 0   | 0     | 0         |
| 0   | 0   | 1     | 1         |
| 0   | 1   | 0     | 1         |
| 0   | 1   | 1     | 1         |
| 1   | 0   | 0     | 0         |
| 1   | 0   | 1     | 0         |
| 1   | 1   | 0     | –         |
| 1   | 1   | 1     | –         |

Хоча тригери мають два виходи – один прямий  $Q$ , а другий – iнверсний  $\bar{Q}$ , в описi тригера i в таблицi станiв зазначають лише стан прямого виходу  $Q$ .

При подачі на вхід скидання (*Reset*)  $R$  рівня логічної «1» тригер устанавлюється в стан логічного «0», а при подачі керуючого сигналу «1» на вхід установки (*Set*)  $S$  – в стан логічної «1». Якщо на обох входах тригера подаються рівні логічного «0» – цей стан відповідає режиму зберігання, і тригер зберігає попередній стан. При подачі на входи  $R$  і  $S$  одночасно рівня «1» тригер буде знаходитися в невизначеному (або неправильному) стані, тому такий стан сигналів  $R$  і  $S$  називається забороненою комбінацією керуючих сигналів і в таблиці станів позначається «–».

За таблицею станів  $RS$ -тригера складаємо таблицю функцій збудження (табл. 3), яка показує, які сигнали необхідно подати на входи тригера для переходу зі стану  $Q^t$  у стан  $Q^{t+1}$ .

Таблиця 3 – Таблиця функцій збудження  $RS$ -тригера

| $Q^t$ | $Q^{t+1}$ | $R$ | $S$ |
|-------|-----------|-----|-----|
| 0     | 0         | X   | 0   |
| 0     | 1         | 0   | 1   |
| 1     | 0         | 1   | 0   |
| 1     | 1         | 0   | X   |

Символ «X» показує, що для даного переходу немає значення, чому дорівнює зазначений сигнал (рівню логічного «0» або «1»).

### ***JK-тригер***

Функціональне позначення тригера наведено на рисунку 2.

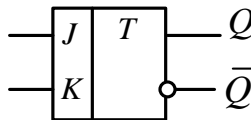


Рисунок 2 – Функціональне позначення  $JK$ -тригера

Таблиця станів  $JK$ -тригера має два вхідних сигнали (сигнали  $J$  і  $K$ ) і один вихідний сигнал  $Q$  (табл. 4).

За таблицею станів *JK*-тригера складаємо таблицю функцій збудження (табл. 5), яка вказує, які сигнали необхідно подати на входи тригера для переходу зі стану  $Q^t$  у стан  $Q^{t+1}$ .

Таблиця 4 – Таблиця станів *JK*-тригера

| $K$ | $J$ | $Q^t$ | $Q^{t+1}$ |
|-----|-----|-------|-----------|
| 0   | 0   | 0     | 0         |
| 0   | 0   | 1     | 1         |
| 0   | 1   | 0     | 1         |
| 0   | 1   | 1     | 1         |
| 1   | 0   | 0     | 0         |
| 1   | 0   | 1     | 0         |
| 1   | 1   | 0     | 1         |
| 1   | 1   | 1     | 0         |

Таблиця 5 – Таблиця функцій збудження *JK*-тригера

| $Q^t$ | $Q^{t+1}$ | $K$ | $J$ |
|-------|-----------|-----|-----|
| 0     | 0         | X   | 0   |
| 0     | 1         | X   | 1   |
| 1     | 0         | 1   | X   |
| 1     | 1         | 0   | X   |

### *T*-тригер

*T*-тригер – це лічильний тригер. *T*-тригер має один вхід, на який подають тактові (лічильні) імпульси (допоміжні входи примусової установки „0” і „1” не розглядаються). Після подачі кожного тактового імпульсу стан *T*-тригера змінюється на протилежний (аналогічно стану *JK*-тригера при комбінації вхідних змінних  $J = 1$  і  $K = 1$ ). Функціональне позначення тригера приведено на рисунку 3.

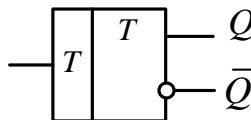


Рисунок 3 – Функціональне позначення *T*-тригера

Таблиця станів  $T$ -тригера у скороченій формі має вхідний сигнал  $T$  і вихідний сигнал  $Q$  (табл. 6).

Таблиця 6 – Таблиця станів  $T$ -тригера

| $T$ | $Q^t$ | $Q^{t+1}$ |
|-----|-------|-----------|
| 0   | 0     | 0         |
| 0   | 1     | 1         |
| 1   | 0     | 1         |
| 1   | 1     | 0         |

За таблицею станів  $T$ -тригера складаємо таблицю функцій збудження, яка показує, які сигнали необхідно подати на вхід тригера для переходу зі стану  $Q^t$  у стан  $Q^{t+1}$  (табл. 7).

Таблиця 7 – Таблиця функцій збудження  $T$ -тригера

| $Q^t$ | $Q^{t+1}$ | $T$ |
|-------|-----------|-----|
| 0     | 0         | 0   |
| 0     | 1         | 1   |
| 1     | 0         | 1   |
| 1     | 1         | 0   |

### ***D*-тригер**

$D$ -тригер (від англійського DELAY) називають інформаційним тригером, також тригером затримки.  $D$  - тригер буває тільки синхронним. Він може керуватися (переключатися) як рівнем тактового імпульсу, так і його фронтом. Функціональне позначення тригера приведено на рисунку 4.

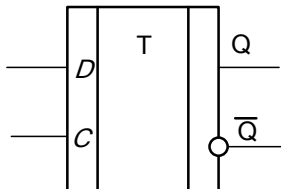


Рисунок 4 – Функціональне позначення  $D$ -тригера

За синхроімпульсом  $D$ -тригер набуває того стану, який надходить на вхід  $D$ , згідно з таблицею станів (таблиця 8).

*Таблиця 8 – Таблиця станів  $D$ -тригера*

| $D$ | $Q^t$ | $Q^{t+1}$ |
|-----|-------|-----------|
| 0   | 0     | 0         |
| 0   | 1     | 0         |
| 1   | 0     | 1         |
| 1   | 1     | 1         |

За таблицею станів  $D$ -тригера складаємо таблицю функцій збудження, яка показує, які сигнали необхідно подати на вхід тригера для переходу зі стану  $Q^t$  у стан  $Q^{t+1}$  (табл. 9).

*Таблиця 9 – Таблиця функцій збудження  $D$ -тригера*

| $Q^t$ | $Q^{t+1}$ | $D$ |
|-------|-----------|-----|
| 0     | 0         | 0   |
| 0     | 1         | 1   |
| 1     | 0         | 0   |
| 1     | 1         | 1   |

## **2 Завдання до лабораторної роботи**

2.1 Для кожного лічильника згідно зі своїм варіантом завдання (табл. 10) виконати такі кроки:

2.1.1 Скласти таблицю переходів.

2.1.2 На основі таблиці переходів для кожного із розрядів скласти таблиці істинності функцій збудження тригерів.

2.1.3 Провести мінімізацію функцій збудження тригерів.

2.1.4 Зібрати схему і перевірити правильність її функціонування.

## **3 Зміст звіту**

Для спроектованого і вивченого у відповідності до завдання лічильника повинні бути наведені:

3.1 Таблиця переходів, таблиця істинності функцій збудження.

3.2 Карти Карно, які відображають хід мінімізації функцій збудження.

3.3 Схема лічильника.

3.4 Висновки.

Таблиця 10 – Варіанти завдань

| Номер варіанта | Порядок рахунку при $x=0$ | Порядок рахунку при $x=1$ | Типи тригерів |            |            |
|----------------|---------------------------|---------------------------|---------------|------------|------------|
|                |                           |                           | 1-й тригер    | 2-й тригер | 3-й тригер |
| 1              | 1,2,3,4,5,7               | 3,1,4,7,5,2               | <i>D</i>      | <i>JK</i>  | <i>D</i>   |
| 2              | 0,1,2,5,6,7               | 5,0,1,6,2,7               | <i>JK</i>     | <i>D</i>   | <i>D</i>   |
| 3              | 0,1,3,4,5,6               | 6,3,1,4,0,5               | <i>D</i>      | <i>D</i>   | <i>JK</i>  |
| 4              | 0,1,2,3,6,7               | 3,0,2,1,7,6               | <i>D</i>      | <i>JK</i>  | <i>D</i>   |
| 5              | 0,1,2,4,5,6               | 4,1,5,0,6,2               | <i>JK</i>     | <i>D</i>   | <i>D</i>   |
| 6              | 1,2,3,4,5,6               | 3,6,2,4,1,5               | <i>D</i>      | <i>D</i>   | <i>JK</i>  |
| 7              | 0,1,2,3,4,6               | 4,0,1,3,6,2               | <i>D</i>      | <i>JK</i>  | <i>D</i>   |
| 8              | 0,1,2,4,6,7               | 7,4,1,0,6,2               | <i>JK</i>     | <i>D</i>   | <i>D</i>   |
| 9              | 0,2,3,4,5,6               | 2,5,3,0,6,4               | <i>JK</i>     | <i>D</i>   | <i>D</i>   |
| 10             | 0,1,2,3,5,6               | 2,0,6,5,3,1               | <i>D</i>      | <i>JK</i>  | <i>D</i>   |

#### 4 Контрольні запитання

1. Тригер. Синхронні, асинхронні тригери.
2. Типи входів тригерів.
3. *D*-, *T*-, *RS*-, *JT*-тригери.
4. Регістри. Класифікація регістрів.
5. Лічильники. Класифікація лічильників.

### Лабораторна робота 3

#### ВИВЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРОГРАМИ ISIS PROTEUS. ЗАПИС І ВИКОНАННЯ ПРОСТИХ ПРОГРАМ

**Мета роботи** – ознайомитися з можливостями програми *ISIS PROTEUS* для моделювання роботи мікропроцесорних пристроїв. Дослідження виконання окремих команд і простих програм. Дослідження різних методів адресації у програмах.

#### 1 Теоретичні основи лабораторної роботи Proteus VSM. Система віртуального моделювання

Proteus VSM створена фірмою Labcenter Electronics на основі ядра SPICE3F5 університету Berkeley та є середовищем наскрізного проектування, що дозволяє проводити створення пристрою, починаючи з його графічного зображення (принципової схеми) і закінчуючи виготовленням друкованої плати пристрою з можливістю контролю на кожному етапі виробництва.

Proteus VSM складається з двох самостійних програм ISIS і ARES. ARES – це трасувальник друкованих плат з можливістю створення своїх бібліотек корпусів. Основною програмою є ISIS, у ній передбачено зв'язок з ARES для передачі проекту для трасування (розводки) друкованих плат.

При запуску програми з'являється головне вікно (рис. 5).

Найбільше простір відведено під вікно редагування EDIT WINDOW. Саме в ньому відбуваються всі основні процеси створення, редагування та налагодження схеми пристрою.

Зліва вгорі розміщене маленьке вікно попереднього перегляду Overview Window (рис. 6), із його допомогою можна переміщатися по вікну редагування.

Переміщати вікно редагування за схемою можна так – утримуючи кнопку SHIFT, рухати курсор мишки, не натискаючи її кнопок, по вікну редагування.

Наближати і віддаляти схему у вікні можна відповідно кнопками F6 і F7 або ж колесом мишки, F5 центрує схему у вікні, а натискання F8 підлаштовує розмір схеми під вікно редагування.

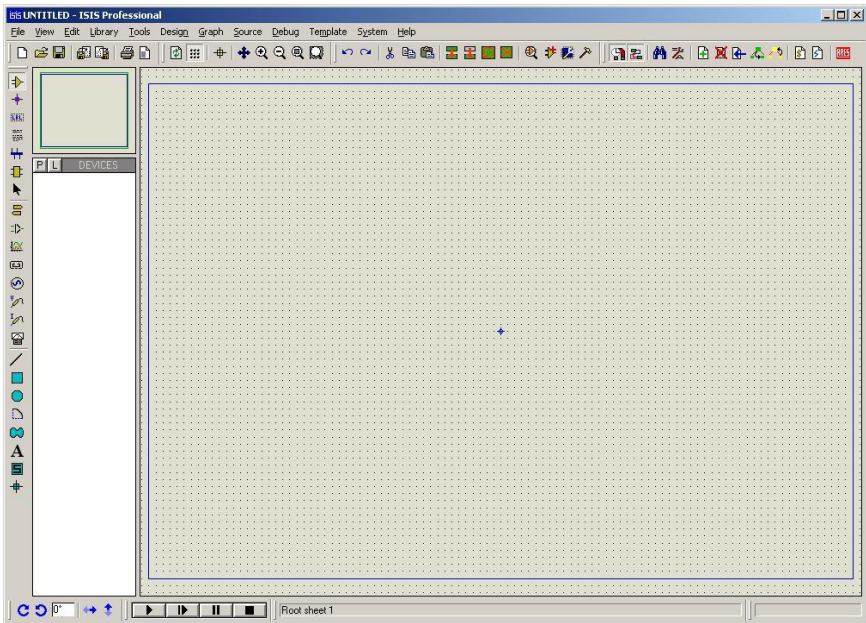


Рисунок 5 – Основне вікно Proteus VSM

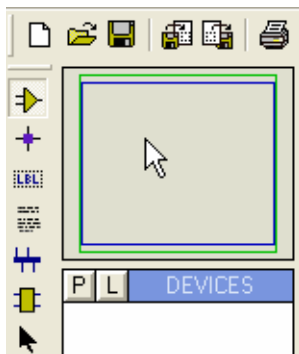


Рисунок 6 – Вікно попереднього перегляду Overview Window

Під вікном попереднього перегляду знаходиться **Object Selector** – список обраних у даний момент компонентів, символів та інших елементів. Виділений у списку об'єкт відображається у вікні попереднього перегляду.



Усі можливі функції та інструменти Proteus VSM доступні через меню розміщеного вгорі основного вікна програми, через піктограми, що знаходяться під меню і в лівому куті основного вікна, і через гарячі клавіші, які можуть перепризначатися користувачем.

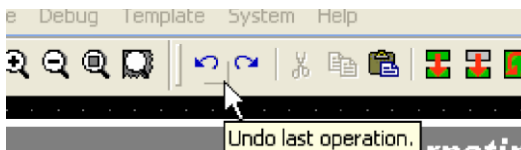
Знизу від основного вікна розміщені: зліва направо кнопки обертання і розвороту об'єкта навколо своєї осі, панель управління інтерактивною симуляцією (наприклад, ПУСК – ПОКРОКОВИЙ РЕЖИМ – ПАУЗА – СТОП), рядок статусу (в якому відображаються: помилки, підказки, поточний стан процесу симуляції і т.п.) і координати курсора, які відображаються в дюймах. Запуск проекту здійснюється натисканням кнопки ПУСК (рис. 7).



*Рисунок 7 – Запуск проекту*

Для того щоб маніпулювати об'єктами, їх потрібно спочатку виділити. Це можна зробити тільки на зупиненому проекті. Для виділення одного об'єкта треба клацнути по ньому правою кнопкою мишки. Для виділення групи можна або, утримуючи CTRL, послідовно клацати правою кнопкою по всіх об'єктах, або, утримуючи праву кнопку, протягнути область виділення по необхідних об'єктах. Виділяти об'єкти треба обережно, повторне клацання правою кнопкою мишки по виділеному об'єкту видалить його (видалити виділені об'єкти можна ще, натиснувши кнопку DELETE).

Скасувати останні і всі попередні дії по порядку можна за допомогою кнопок відміни (UNDO, REDO) (рис. 8). Кнопки скасування діють як назад, по хронології, так і вперед.



*Рисунок 8 – Розміщення кнопок UNDO, REDO*

Виділені об'єкти можна переміщати за схемою, схопивши їх лівою кнопкою мишки, пересунувши в потрібне місце, відпустити кнопку. А за допомогою цих кнопок виробляються групові операції з виділеними об'єктами. По порядку: копіювання, переміщення, поворот і видалення.

Усі елементи знаходяться, як на «складі», в бібліотеці компонентів. Щоб потрапити на цей «склад», перейдемо в режим COMPONENT (компоненти), натиснувши відповідну піктограму (рис. 9).

Тепер, або клацнувши по піктограмі P (Pick devices), або двічі клацнувши лівою кнопкою в полі вибору компонентів Object Selector, ми потрапимо на «склад» (рис. 10).

Компоненти можна вибирати за категоріями Category, підкатегоріями-Sub category, за виробником Manufacturer або ж шукати за ключовими словами Keywords. Двічі клацнувши по рядку з назвою об'єкта, ви підтверджуєте вибір компонента. Компоненти набираються по одному примірнику, розмножити їх можна вже потім, просто вибираючи у списку Object Selector. Закрийте бібліотеку, натиснувши ОК або ж закривши вікно.

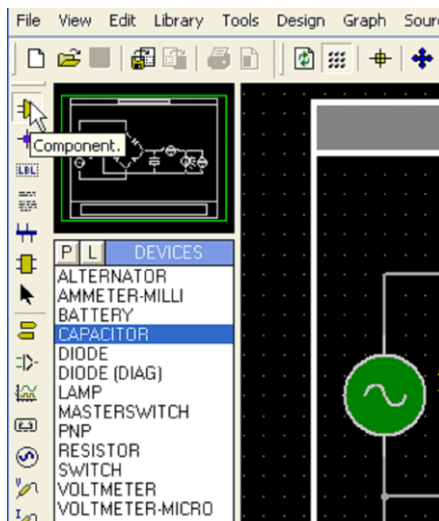


Рисунок 9 – Підключення бібліотеки компонентів

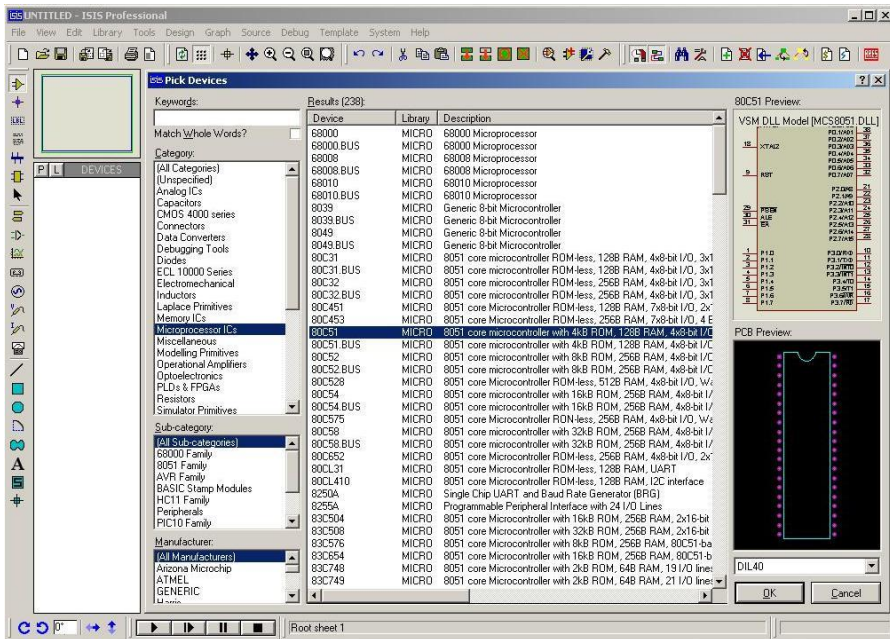


Рисунок 10 – Бібліотека компонентів

Елементи типу терміналів («землі» або «корпусу») вибираються в режимі **INTER SHEET TERMINAL** (рис. 11).

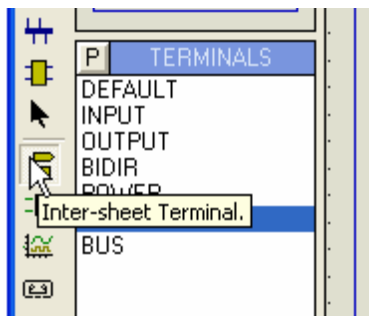
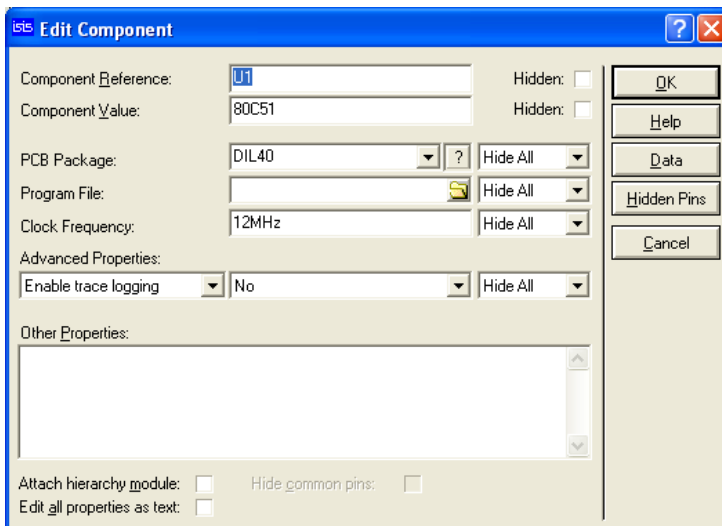


Рисунок 11 – Підключення терміналів у режимі **INTER SHEET TERMINAL**

Щоб відкрити вікно редагування компонента, потрібно або, виділивши компонент, клацнути по ньому лівою кнопкою мишки, або, помістивши на нього курсор, не натискаючи кнопок мишки, натиснути *CTRL + E* (рис. 12).



*Рисунок 12 – Вікно редагування компонента*

З'єднання компонентів. Помістіть курсор на необхідний вивід компонента, на кінці курсора з'явиться хрестик, який показує, що з'єднання можливе. Клацніть лівою кнопкою, пересуньте курсор, з'явиться тонка лінія, що показує можливі з'єднання. Підведіть курсор до необхідного виводу та натисніть ліву кнопку. Проведене з'єднання відобразиться чорним кольором.

З'єднання типу BUS (шина). Виберіть тип з'єднання (рис. 13). Відзначаючи лівою кнопкою точки, проведемо шину. Кінець шини відзначається клацанням правої кнопки (рис. 14). Для нумерації проводів існують мітки. Перейдіть у режим WIRE LABEL (мітка проводу), клацнувши по піктограмі з написом LBL. Клацаючи по виводу, ви відкриваєте вікно введення і редагування мітки.



Рисунок 13 – Вибір з'єднання «шина»

Для створення нового проекту використовується опція меню **FILE> NEW DESIGN**.

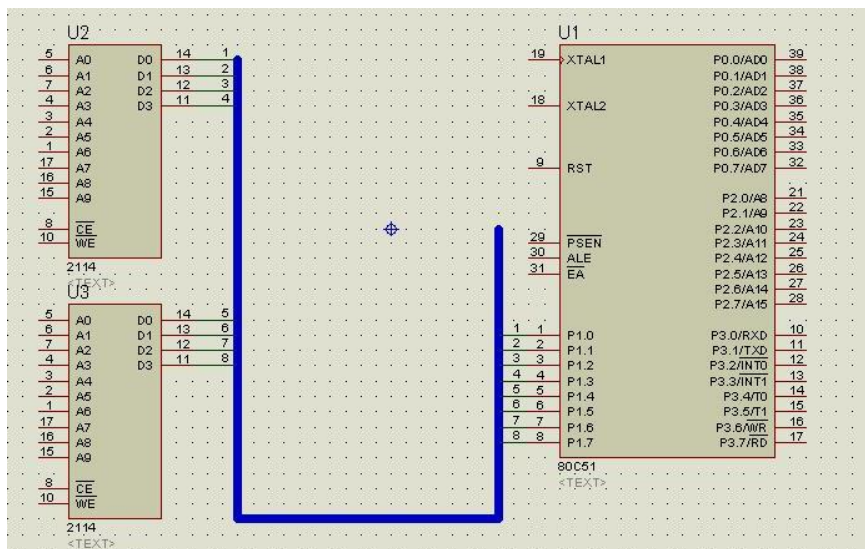
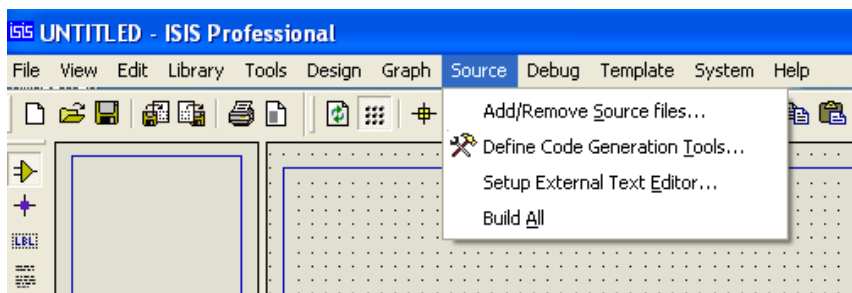


Рисунок 14 – З'єднання «шина»

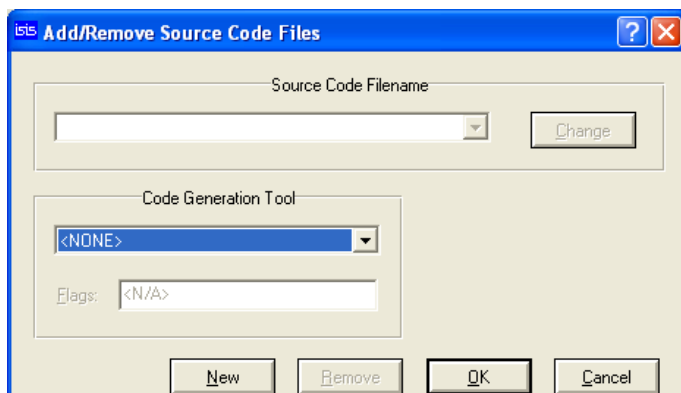
### Створення мікропроцесорних схем

Наберіть на робочому столі необхідну мікропроцесорну схему. Для запису коду програми створіть порожній текстовий файл майбутньої програми та надайте йому розширення **.asm**. Додайте створений файл у проект. Для цього в меню **SOURCE** (рис. 15) виберіть **ADD/REMOVE SOURCE FILE** (дати/видалити файл).



*Рисунок 15 – Меню SOURCE*

У вікні натисніть кнопку **NEW** (новий). У рядку **SOURCE CODE FILENAME** виберіть ваш створений файл із розширенням **.asm** за допомогою кнопки **CHANGE** (змінити), а в рядку **CODE GENERATION TOOL** – компілятор **ASEM51**. Підтвердіть вибір, натиснувши **OK** (рис. 16).



*Рисунок 16 – Вікно ADD / REMOVE SOURCE FILES*

Для створення файлу прошивки відкрийте меню **SOURCE** і натисніть **BUILD ALL** (рис. 15). Відкриється вікно компілятора, повідомляючи, що все в порядку, якщо ви не допустили помилок, або ж з'являться рядки з помилками. (В **PROTEUS** можна підключити зовнішній редактор, замінивши вбудований. Для цього зайдіть у меню **SOURCE** пункт **SETUP EXTERNAL TEXT EDITOR**. Натисніть **BROWSE** (перегляд) і знайдіть свій редактор).

Закриваємо вікно компілятора та завантажуюємо скомпільований файл у мікроконтролер. Для цього у вікні властивостей мікроконтролера (рис. 12) у рядку **PROGRAMM FILE** виберемо файл прошивки, який з'явився в нашій папці після компіляції. У рядку **PROCESSOR CLOCK FREQUENCY** (тактова частота процесора) виставляється тактова частота процесора.

### Можливості PROTEUS із налагодження програми і перегляду стану внутрішніх вузлів мікроконтролера

Поставте проект на паузу. З'явиться вікно налагоджувача (рис. 17), якщо цього не сталося, то в меню **DEBUG** відзначте пункт **CPU SOURCE CODE**.

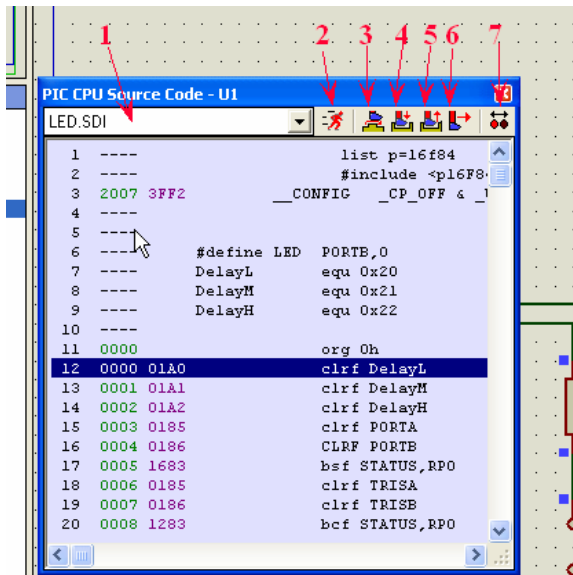


Рисунок 17 – Вікно налагоджувача

На рисунку під цифрами:

1. Вікно вибору файла налагодження.
2. Продовжити виконання програми.
3. Крок без входу в підпрограму.
4. Крок із входом у підпрограму.

5. Виконувати код до повернення з підпрограми. Фактично зупинка відбувається на наступній за RETURN команді. Використовувати цю опцію, природно, можна тільки перебуваючи в підпрограмі.

6. Виконувати, поки не буде досягнутий виділений рядок.

7. Тригер точок зупинення (брейк-пойнт).

Натиснувши правою кнопкою по вікну налагоджувача, можна змінити його налаштування.

У меню є такі опції:

|                                |   |
|--------------------------------|---|
| GOTO LINE                      | перейти на лінію                              |
| GOTO ADDRESS                   | перейти на адресу                             |
| FIND                           | знайти  |
| TOGGLE (SET/CLEAR) BREAK POINT | встановити / очистити точку зупинення         |
| ENABLE ALL BREAK POINT         | заборонити всі точки зупинення                |
| DISABLE ALL BREAK POINT        | заборонити всі точки зупинення (не видаляти!) |
| CLEAR ALL BREAK POINT          | видалити всі точки зупинення                  |
| FIX-UP BREAKPOINTS ON LOAD     | зафіксувати (дозволити) точки зупинення       |
| DISPLAY LINE NUMBERS           | показувати номери ліній                       |
| DISPLAY ADDRESSES              | показувати адреси команд                      |
| DISPLAY OPCODES                | показувати опкод команд                       |
| SET FONT                       | вибір шрифту                                  |
| SET COLOR                      | вибір кольору тексту, фону                    |

Щоб бачити символи кирилиці, виберіть шрифт **COURIER NEW**, а для того щоб усе прибрати у вікні, поставте розмір шрифту 8.



Для перегляду стану внутрішніх реєстрів у меню **DEBUG** відзначте пункт **CPU REGISTER**. Тепер при трасуванні коду ви зможете спостерігати вміст реєстрів спеціальних функцій. Є два способи побачити вміст реєстрів користувача та реєстрів спеціальних функцій одночасно, але перший не дуже зручний. Відзначте в меню **DEBUG** пункт **CPU DATA MEMORY**. Незручність полягає в тому, що відображаються вони усі відразу, у вигляді таблиці і лише під час паузи або на точці зупинення.

Другий спосіб – це **WATCH WINDOW**. Позначте відповідний пункт у меню **DEBUG**.

Клацніть правою кнопкою по вікні **WATCH WINDOW**.

|                          |   |
|--------------------------|---|
| ADD ITEMS (BY NAME)      | додати елемент по імені                                   |
| ADD ITEMS (BY ADDRESSES) | додати елемент за адресою                                 |
| WATCHPOINT CONDITION     | умова для зупинення                                       |
| SELECT ALL               | вибрати всі елементи                                      |
| RENAME ITEM              | переіменувати   |
| COPY CLIPBOARD           | копіювати в буфер обміну                                  |
| DELETE ITEM              | видалити вибране  |
| DATA TYPE                | в якому вигляді подавати дані (рядок, байт, слово і т.д.) |
| DISPLAY FORMAT           | формат даних (двійковий, десятковий і т.д.)               |
| SHOW ADDRESSES           | показувати адресу   |
| SHOW GRIDLINES           | показувати сітку  |
| SHOW WATCH EXPRESSIONS   | показувати умову  |
| MINIMUM SIZE             | мінімізувати розмір                                       |

## 2 Завдання до лабораторної роботи

### Завдання 1

Оберіть у категорії Microprocessor ICs мікроконтролер 8051 і помістіть його на робочий стіл. Створіть текстовий файл (ім'я не більше 8 букв) із розширенням \*.asm. Наберіть такі рядки:

```
org 0000h
jmp start
org 0100h
start:  mov A, # 01H
        jmp start
end
```

Скомпілюйте даний файл. Отриманий файл із розширенням \*.Hex завантажте в контролер. Запустіть програму в покроковому режимі. Перевірте результат виконання програми.

### Завдання 2

Уведіть програму 3.1. Запустіть програму в покроковому режимі. Перевірте результат виконання програми. Уведіть замість числа 95 будь-яке інше число. Перезапустіть програму. Замінюючи в програмі 1 почергово команду **cpl a** на команди **inc a**, **dec a**, **rl a**, **rr a**, **swap a**, досліджуйте результат виконання програми за числом, записаним в комірці з номером 101. Заповніть таблицю 11 за результатами роботи програми.

### Програма 3.1

```
org 0000h
jmp in_begdata
org 0050h
in_begdata:  mov 100, # 95
             jmp start
             org 0060h
start:      mov a, 100
            cpl a
            mov 101, a
            jmp start
end
```

*Таблиця 11*

| Число, записане за адресою 100 | Команда   | Число, записане за адресою 101 |
|--------------------------------|---|--------------------------------|
|                                | cpl a<br>inc a<br>dec a<br>rl a<br>rr a<br>swap a |                                |

*Завдання 3*

Уведіть програму 3.2. Запустіть програму в покроковому режимі. Перевірте результат виконання програми.

**Програма 3.2**

```
org 0000h
jmp in_begdata
org 0050h
in_begdata: mov 100, #95
             jmp start
             org 0060h
start:      mov r0, # 100
             mov a, @r0
             cpl a
             mov r1, #101
             mov @r1, a
             jmp start
             end
```

*Завдання 4*

Напишіть і перевірте програму, що вирішує таке завдання: число, записане за адресою 100, збільшити на 15 і записати результат за адресою 101.

*Завдання 5*

Напишіть і перевірте програму, що вирішує таке завдання: додайте два числа, які записані в комітках пам'яті 100 і 101, а результат запишіть у комірку пам'яті з номером 102.

### **3 Контрольні питання**

1. Способи адресації мікроЕОМ сімейства MCS-51.
2. З якими типами даних може оперувати мікроконтролер.
3. Організація пам'яті мікроЕОМ сімейства MCS-51.
4. Який адресний простір реалізується за допомогою прямої і непрямой адресації.
5. Способи адресації і розташування в ОЗП бітового простору пам'яті.
6. Способи передачі управління в адресному просторі ПЗУ. Приклад.
7. Способи адресації зовнішньої пам'яті ОЗП. Приклад.
8. Перелічіть реєстри спеціальних функцій мікроЕОМ сімейства MCS-51.
9. З якими типами даних може оперувати мікроконтролер.

### **Зміст звіту**

Звіт про лабораторну роботу повинен містити:

1. Мету і завдання роботи.
2. Вихідні тексти програм із доповненнями, що забезпечують тестування і налагодження.
3. Висновки з роботи.

## Лабораторна робота 4

### ОРГАНІЗАЦІЯ РОБОТИ ЗІ СТЕКОМ. РОБОТА З БІТОВИМ ПРОСТОРОМ. ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ

**Мета роботи** – вивчення методів використання стека при створенні програм. Вивчення команд роботи з бітами. Вивчення організації умовних переходів.

#### 1 Теоретичні основи лабораторної роботи Регістри спеціальних функцій

До адресного простору пам'яті даних примикає адресний простір регістрів спеціальних функцій SFR (Special Function Register) (таблиця 12).

**Акумулятор (ACC)** є джерелом операнда і місцем фіксації результату при виконанні арифметичних, логічних операцій і ряду операцій передачі даних. Крім того, тільки з використанням акумулятора можуть бути виконані операції зсувів, перевірка на нуль, формування прапорцю паритету і т.п. У розпорядженні користувача є 8 регістрів загального призначення R0–R7 одного із чотирьох можливих банків. Під час виконання багатьох команд в арифметико-логічному пристрої (АЛП) формується ряд ознак операції (прапорців), які фіксуються в регістрі PSW.

#### Регістр стану програми (PSW)

Регістр стану програми (PSW) призначений для зберігання інформації про стан АЛП під час виконання програми.

Найбільш «активним» прапорцем PSW є прапорець перенесення, який бере участь і модифікується в процесі виконання більшості операцій, включаючи добуток, частку і зсуви. Крім того, прапорець перенесення (C) виконує функції “булевого акумулятора” в командах, що маніпулюють із бітами. Прапорцем переповнення (OV) фіксується арифметичне переповнення при операціях над цілими числами зі знаком і роблять можливим використання арифметики в додаткових кодах.

Таблиця 12 – Розміщення регістрів спеціальних функцій

| Адреса | Символ | Назва  |
|--------|--------|--|
| 0E0H   | *ACC   | Акумулятор (Accumulator)   |
| 0F0H   | *B     | Регістр – розширювач акумулятора (Multiplication Register)                 |
| 0D0H   | *PSW   | Регістр стану програми (Program Status Word)                               |
| 080H   | *P0    | Порт 0 (SFR P0)  |
| 090H   | *P1    | Порт 1 (SFR P1)  |
| 0A0H   | *P2    | Порт 2 (SFR P2)  |
| 0B0H   | *P3    | Порт 3 (SFR P3)  |
| 081H   | SP     | Регістр – вказівник стека (Stack Pointer)                                  |
| 083H   | DPH    | Старший байт регістра вказівника даних DPTR (Data Pointer High)            |
| 082H   | DPL    | Молодший байт регістра – вказівника даних DPTR (Data Pointer Low)          |
| 08CH   | TH0    | Старший байт таймера 0   |
| 08AH   | TL0    | Молодший байт таймера 0  |
| 08DH   | TH1    | Старший байт таймера 1   |
| 08BH   | TL1    | Молодший байт таймера 1  |
| 089H   | TMOD   | Регістр режимів таймерів лічильників (Timer/Counter Mode Control Register) |
| 088H   | *TCON  | Регістр управління статусу таймерів (Timer/Counter Control Register)       |
| 0B8H   | *IP    | Регістр пріоритетів (Interrupt Priority Control Register)                  |
| 0A8H   | *IE    | Регістр маски переривання (Interrupt Enable Register)                      |
| 087H   | PCON   | Регістр керування потужністю (Power Control Register)                      |
| 098H   | *SCON  | Регістр керування приймача-передавача (Serial Port Control Register)       |
| 099H   | SBUF   | Буфер приймача-передавача (Serial Data Buffer)                             |

АЛП не керує прапорцями селекції банку регістрів (RS0, RS1), їх значення повністю визначається прикладною програмою та використовується для вибору одного з чотирьох регістрових банків.

Таблиця 13 – Перелік прапорців, їх символічні імена і умови формування

| Символ       | Позиція          | Ім'я і призначення   |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|--------------|------------------|--|-----------|------|--------------------|--------------------|---|---|---|-----------|---|---|---|-----------|---|---|---|-----------|---|---|---|-----------|
| P            | PSW.0            | Прапорець пріоритету. Встановлюється і скидається апаратно в кожному циклі команди і фіксує непарне (парне) число одиничних бітів в акумуляторі  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| -            | PSW.1            | Не використовується  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| OV           | PSW.2            | Прапорець переповнення. Встановлюється і скидається апаратно при виконанні арифметичних операцій   |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| RS0 -<br>RS1 | PSW.3 -<br>PSW.4 | Біти вибору банку регістрів. Можуть бути змінені програмним шляхом   |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  | <table border="1"> <thead> <tr> <th>RS0</th> <th>RS1</th> <th>Банк</th> <th>Діапазон адрес ОЗП</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>00H - 07H</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>08H - 0FH</td> </tr> <tr> <td>0</td> <td>1</td> <td>2</td> <td>10H - 17H</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> <td>18H - 1FH</td> </tr> </tbody> </table> | RS0       | RS1  | Банк               | Діапазон адрес ОЗП | 0 | 0 | 0 | 00H - 07H | 1 | 0 | 1 | 08H - 0FH | 0 | 1 | 2 | 10H - 17H | 1 | 1 | 3 | 18H - 1FH |
|              |                  | RS0  | RS1       | Банк | Діапазон адрес ОЗП |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  | 0  | 0         | 0    | 00H - 07H          |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  | 1  | 0         | 1    | 08H - 0FH          |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| 0            | 1                | 2  | 10H - 17H |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| 1            | 1                | 3  | 18H - 1FH |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  |  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  |  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
|              |                  |  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| F0           | PSW.5            | Прапорець користувача. Може бути встановлений, скинутий або перевірений програмою користувача  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| AC           | PSW.6            | Прапорець допоміжного перенесення. Встановлюється і скидається тільки апаратними засобами під час виконання команд добутку і частки і сигналізує про перенесення або заміні в біті 3 акумулятора   |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |
| C            | PSW.7            | Прапорець перенесення. Встановлюється і скидається як апаратно, так і програмним шляхом  |           |      |                    |                    |   |   |   |           |   |   |   |           |   |   |   |           |   |   |   |           |

**Регістр-вказівник стека SP** восьмибітовий. Він може адресувати будь-яку область внутрішньої пам'яті даних. У мікроЕОМ сімейства MCS-51 стек «росте вгору», тобто перед виконанням команди PUSH або CALL вміст SP інкрементується, після чого відбувається запис інформації у стек. Відповідно при вилученні інформації зі стека регістр SP декрементується після вилучення інформації. У процесі ініціалізації мікроЕОМ після сигналу скидання або при включенні живильної напруги в SP заноситься код 07H. Це означає, що перший елемент стека буде розміщуватися в комірці пам'яті з адресою 08H.

Команди PUSH і POP призначені відповідно для запису даних у стек і їх читання зі стека. Розмір стека обмежено лише розміром резидентної пам'яті даних.

### Організація бітового простору пам'яті

Частина пам'яті даних являє собою так звану бітову область, у ній є можливість за допомогою спеціальних бітових команд адресуватися до кожного розряду комірок пам'яті. Адреса прямо адресованих бітів може бути записана або у вигляді (адреса байта).(Розряд), наприклад, вираз 21.3 означає третій розряд комірки пам'яті з адресою 21Н або у вигляді абсолютної бітової адреси. Відповідність цих двох способів адресації можна визначити за таблицею 14.

*Таблиця 14 – Адреса бітових областей пам'яті мікроконтролера MCS-51*

| Адреса байта | Адреса бітів за розрядами |    |    |    |    |    |    |    |
|--------------|---------------------------|----|----|----|----|----|----|----|
| Adr          | D7                        | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 2FH          | 7F                        | 7E | 7D | 7C | 7B | 7A | 79 | 78 |
| 2EH          | 77                        | 76 | 75 | 74 | 73 | 72 | 71 | 70 |
| 2DH          | 6F                        | 6E | 6D | 6C | 6B | 6A | 69 | 68 |
| 2CH          | 67                        | 66 | 65 | 64 | 63 | 62 | 61 | 60 |
| 2BH          | 5F                        | 5E | 5D | 5C | 5B | 5A | 59 | 58 |
| 2AH          | 57                        | 56 | 55 | 54 | 53 | 52 | 51 | 50 |
| 29H          | 4F                        | 4E | 4D | 4C | 4B | 4A | 49 | 48 |
| 28H          | 47                        | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 27H          | 3F                        | 3E | 3D | 3C | 3B | 3A | 39 | 38 |
| 26H          | 37                        | 36 | 35 | 34 | 33 | 32 | 31 | 30 |
| 25H          | 2F                        | 2E | 2D | 2C | 2B | 2A | 29 | 28 |
| 24H          | 27                        | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 23H          | 1F                        | 1E | 1D | 1C | 1B | 1A | 19 | 18 |
| 22H          | 17                        | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| 21H          | 0F                        | 0E | 0D | 0C | 0B | 0A | 09 | 08 |
| 20H          | 07                        | 06 | 05 | 04 | 03 | 02 | 01 | 00 |

**ПРИМІТКА.** Адреса прямо адресованих бітів може бути записана у вигляді виразу (адреса байта).(Розряд), наприклад, вираз 21.3 означає адресу третього розряду комірки пам'яті з адресою 21Н або у вигляді абсолютної бітової адреси.



## *Регістри спеціальних функцій*

*Таблиця 15 – Карта адресованих бітів у блоці регістрів спеціальних функцій*

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

**ПРИМІТКА.** Адреса прямо адресованих бітів може бути записана у вигляді виразу (назва регістра).(Розряд), наприклад, вираз SCON.3 означає адресу третього розряду регістра SCON або у вигляді абсолютної бітової адреси. Крім того, деякі біти керуючих регістрів мають власні назви, так, наприклад, даний біт має назву TB8.

## 2 Завдання до лабораторної роботи

### *Завдання 4.1*

Уведіть програму 4.1 для роботи з регістрами різних банків. Запустіть програму в покроковому режимі. Поспостерігайте момент зміни банків регістрів і змін станів регістрів.

#### **Програма 4.1**

Складіть програму, яка реалізує таке завдання:

- запишіть у R0 0-го банку пам'яті число 51;
- запишіть у R4 1-го банку пам'яті число 30;
- завантажте в акумулятор число 35H;
- перешліть вміст акумулятора в регістр R4 2-го банку пам'яті.

### *Завдання 4.2*

Уведіть програму 4.2 для роботи зі стеком. Запустіть програму в покроковому режимі. Розгляньте, як змінюється вказівник стека під час виконання програми.

#### **Програма 4.2**

Складіть програму, яка реалізує таке завдання:

- установіть вказівник стека на комірку пам'яті 50H;
- запишіть у регістр R3 2-го банку пам'яті число 10;
- запишіть в акумулятор число 20 і додайте його до числа, яке зберігається в регістрі R3 2-го банку пам'яті;
- збережіть вміст акумулятора в стеку;
- запишіть у регістр R2 1-го банку пам'яті число 40;
- вилучіть дані зі стека в акумулятор;
- додайте вміст регістра R2 1-го банку пам'яті до акумулятора.

### *Завдання 4.3*

Уведіть програму 4.3 для дослідження команд роботи з бітами. Запустіть програму в покроковому режимі. Розгляньте, як змінюється вміст 25-ї комірки пам'яті.

#### **Програма 4.3**

Використовуючи команди роботи з бітами, складіть програму, яка реалізує таке завдання: завантажте в комірку пам'яті з номером 25H число 0FH. Установіть в одиницю 6-й та 4-й біти, а в 0 – 1-й та 3-й біти. Проінвертуйте 1-й та 5-й біти.

#### *Завдання 4.4*

Уведіть програму 4.4. Запустіть програму в покроковому режимі. Перевірте результат виконання програми. Замість числа 1 уведіть числа 5, 10 і перевірте роботу програми.

#### **Програма 4.4**

Порівняння вмісту акумулятора з числом 5. Якщо вміст акумулятора дорівнює 5, у регістр r1 запишіть 2, якщо менше – у регістр r1 запишіть 3, якщо більше – у регістр r1 запишіть 1.

```

                                org    0000h
                                jmp    start
                                org    0100h
start:                          mov    a,#1    ;уведення числа в акумулятор
                                cjne   a,#5,neravn
                                mov    r1,#2
                                jmp    start
neravn:                          jc     men
                                mov    r1,#1
                                jmp    start
men:                              mov    r1,#3
                                jmp    start
end
```

#### *Завдання 4.5*

Уведіть програму 4.5. Запустіть програму в покроковому режимі. Перевірте результат виконання програми.

#### **Програма 4.5**

Складіть програму, яка реалізує таке завдання: використовуючи команди для організації циклів, записи в ОЗП за допомогою непрямої адресації, запишіть у комірку ОЗП із 30Н по 40Н числа 1 – 16 відповідно. Прочитайте в акумулятор комірку ОЗП із номером 35Н, використовуючи команду із прямо-адресним байтом.

## **Контрольні запитання**

1. Регістри спеціальних функцій.
2. Порядок роботи з регістрами загального призначення.
3. Стек. Організація стека.
4. Способи адресації бітового простору.

## **Зміст звіту**

Звіт про лабораторну роботу повинен містити:

1. Мету і завдання роботи.
2. Вихідні тексти програм із доповненнями, що забезпечують тестування і налагодження.
3. Висновки з роботи.

## Лабораторна робота 5

### ВВЕДЕННЯ-ВИВЕДЕННЯ, МАСКУВАННЯ ДАНИХ І ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ

**Мета роботи** – вивчення організації умовних переходів і програмних способів маскування даних. Дослідження методів організації обміну з найпростішими пристроями введення-виведення.

#### 1 Короткі теоретичні відомості.

##### **Пристрій паралельних портів мікроконтролера**

Паралельні порти призначені для обміну багаторозрядною двійковою інформацією між мікроконтролером і зовнішніми пристроями, при цьому як зовнішній пристрій може використовуватися інший мікроконтролер. Кожен із портів містить восьмирозрядний регістр, що має байтову і бітову адресацію для установки (запис "1") або скидання (запис "0") розрядів цього регістра за допомогою програмного забезпечення. Виходи цих регістрів з'єднані із зовнішніми виводами мікросхеми. З точки зору зовнішнього пристрою порт являє собою звичайне джерело або приймач інформації зі стандартними цифровими логічними рівнями (як правило, TTL), а з точки зору мікропроцесора – це комірка пам'яті, в яку можна записувати дані або в якій сама посібі з'являється інформація.

Як зовнішній пристрій може застосовуватися будь-який об'єкт управління або джерело інформації (різні кнопки, датчики, мікросхеми, додаткова пам'ять, виконавчі механізми, двигуни, реле і так далі). Залежно від напрямку передачі даних паралельні порти називаються портами введення, виведення, або портами вводу-виводу.

##### **Підключення зовнішніх пристроїв до порту паралельного порту мікроконтролера**

Зовнішніми пристроями називаються будь-які пристрої, якими управляє, від яких отримує або яким передає інформацію мікроконтролер. Як зовнішні пристрої можуть застосовуватися пристрої введення-виведення інформації – індикатор, дисплей, датчик, кнопка, клавіатура, інший мікропроцесор.

Внутрішня схема порту побудована так, щоб максимально спростити підключення зовнішніх пристроїв до мікроконтролера. Наявність у схемі порту вихідного транзистора дозволяє підключити до виведень порту світлодіодні індикатори безпосередньо, без підсилювача потужності, як це показано на рисунку 17. При цьому необхідно стежити за максимальною допустимою потужністю, що розсіюється на мікросхемі та окремих виведень порту. Світлодіод у такій схемі спалахує при низькому потенціалі на виведенні порту, для цього в порт повинен бути записаний логічний нуль. Резистор R1 обмежує струм через світлодіод. Якщо цей резистор не поставити, то струм по ланцюгу індикатора може досягти неприпустимої величини, і світлодіод або внутрішній транзистор вийдуть із ладу.

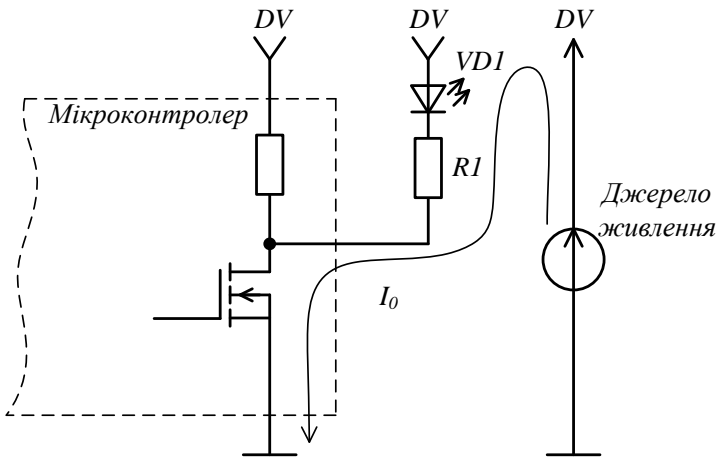


Рисунок 17 – Еквівалентна схема підключення світлодіодного індикатора до паралельного порту

Двонаправлений порт дозволяє отримувати цифрову інформацію від різних джерел – кнопок, датчиків, мікросхем. Узгодження мікросхем між собою не становить труднощів, оскільки практично всі сучасні цифрові мікросхеми за входом і виходом узгоджені між собою і мають строго певні значення логічних рівнів.



## 2 Завдання до лабораторної роботи

Відкрийте проект зі схемою, наведеною на рисунку 19. Напишіть програми відповідно до завдань і виконайте їх налагодження.

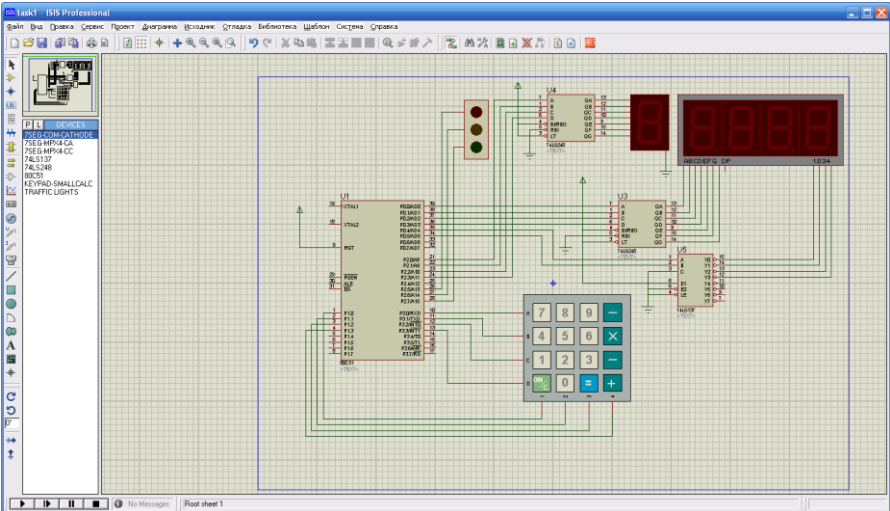


Рисунок 19 – Схема для дослідження портів вводу-виводу

### Завдання 5.1

Уведіть програму 5.1. Запустіть програму. Перевірте результат виконання програми.

### Програма 5.1

Напишіть програму для відображення на чотирьохрозрядному індикаторі рядки з цифр 1234.

### Завдання 5.2

Уведіть програму 5.2. Запустіть програму. Перевірте результат виконання програми.

### Програма 5.2

Напишіть програму для відображення на однорозрядному індикаторі цифри введеної з першого рядка клавіатури.

### Завдання 5.3

Уведіть програму 5.3. Запустіть програму. Перевірте результат виконання програми.



### **Програма 5.3**

Напишіть програму, в якій включається колір світлофора залежно від того, в якому рядку була натиснута кнопка клавіатури.

#### **Контрольні питання**

1. Функції портів P0 і P2 при зверненні до зовнішньої пам'яті.
2. Організація введення даних через порти P0-P3.
3. Альтернативні функції портів P0-P3.

#### **Зміст звіту**

Звіт про лабораторну роботу повинен містити:

1. Мета і завдання роботи.
2. Вихідні тексти програм із доповненнями, що забезпечують тестування і налагодження.
3. Висновки з роботи.

## Лабораторна робота 6

### СИСТЕМА ПЕРЕРИВАНЬ. ОБРОБКА ЧАСОВИХ СИГНАЛІВ. ПРОГРАМУВАННЯ ПОСЛІДОВНОГО ПОРТУ ОЕОМ

**Мета роботи** – вивчення системи переривань, таймера-лічильника й послідовного порту мікроконтролера MCS-51.

#### Таймери / лічильники мікроконтролерів сімейства MCS-51

У базових моделях сімейства є два програмувальних 16-бітових таймери/лічильники (Т/Л0 і Т/Л1), які можуть бути використані як таймери, так і як лічильники зовнішніх подій. У першому випадку стан відповідного таймера/лічильника інкрементується в кожному машинному циклі, тобто через кожні 12 періодів коливань кварцового резонатора, в другому він інкрементується під впливом переходу з 1 в 0 зовнішнього вхідного сигналу, що подається на відповідний (Т0, Т1) вивід мікроЕОМ MCS-51. Оскільки на розпізнавання періоду потрібні два машинних цикли, максимальна частота підрахунку вхідних сигналів дорівнює  $1/24$  частоти резонатора. На тривалість періоду вхідних сигналів обмежень згори немає. Для гарантованого прочитання вхідний сигнал повинен утримувати значення 1, як мінімум, упродовж одного машинного циклу мікроЕОМ.

Для управління режимами роботи Т/Л і для організації їх взаємодії з системою переривань використовуються два регістри спеціальних функцій (ТMOD і TCON), опис яких наведено у таблицях 16 і 17.

#### Режими роботи таймерів-лічильників

Як впливає з опису керуючих бітів ТMOD, для обох Т/Л режими роботи 0, 1 і 2 однакові. Режим 3 для Т/Л0 і Т/Л1 різний. Розглянемо коротко роботу Т/Л у кожному з режимів.

**Режим 0.** У цьому режимі таймерний регістр має розрядність 13 бітів. При переході зі стану "всі одиниці" у стан "усі нулі" встановлюється прапорець переривання від таймера TF 1. Вхідний синхросигнал таймера 1 дозволений (надходить на вхід Т/Л1), коли керуючий біт TR1 установлений в 1 або керуючий біт GATE (блокування) дорівнює 0, або на вхід запиту

переривання INT1 надходить рівень 1. Установка біта GATE в 1 дозволяє використовувати таймер для вимірювання тривалості імпульсного сигналу, що подається на вхід запиту переривання.

Таблиця 16 – Регістр режиму роботи таймера лічильника TMOD

| Символ | Позиція  | Ім'я і призначення   |    |  |
|--------|--|--|----|--|
| GATE   | TMOD.7<br>для<br>Т/Л1 і<br>TMOD.3<br>для<br>Т/Л0 | Керування блокуванням. Якщо біт встановлено, то таймер (лічильник) "х" дозволений до того часу, поки на вході "INTx" високий рівень і біт керування "TRx" встановлено. Якщо біт скинуто, то Т/Л дозволяється, як тільки біт керування "TRx" встановлюється |    |  |
| C/T    | TMOD.6<br>для<br>Т/Л1 і<br>TMOD.2<br>для<br>Т/Л0 | Біт вибору режиму таймера або лічильника подій. Якщо біт скинуто, то працює таймер від внутрішнього джерела сигналів синхронізації. Якщо встановлено, то працює лічильник від зовнішніх сигналів на вході "Tx"   |    |  |
| M1     | TMOD.5<br>для<br>Т/Л1 і<br>TMOD.1<br>для<br>Т/Л0 | Режим роботи   |    |  |
|        |  | M1   | M0 |  |
|        |  | 0  | 0  | 8-розрядний таймер (лічильник) із п'ятирозрядним передділником на 32   |
| 0      | 1  | 16-бітовий таймер (лічильник). "ТНх" і "ТLх" включено послідовно   |    |  |
| M0     | TMOD.4<br>для<br>Т/Л1 і<br>TMOD.0<br>для<br>Т/Л0 | 1  | 0  | 8-бітовий автоперезавантажений таймер (лічильник). "ТНх" зберігає значення, яке повинно бути перезавантажене в "ТLх" кожного разу за переповненням           |
|        |  | 1  | 1  | Таймер (лічильник) 1 заблокований і зберігає своє значення. Таймер (лічильник) 0 являє собою два незалежних пристрої на базі 8-розрядних регістрів TL0 і TH0 |

**Режим 1.** Відмінність від режиму 0 полягає в тому, що встановлення режиму 1 перетворює Т/Л1 на пристрій із 16-розрядним регістром. Для Т/Л0 регістр складається із програмно доступних пар TL0, TH0, для Т/Л1 – із програмно доступних пар TL1, TH1.

**Режим 2.** У режимі 2 Т/Л являє собою пристрій на базі 8-розрядного регістра TL0 для Т/Л0 та TL1 для Т/Л1. При кожному переповненні TLi, крім установаження в регістрі TCON прапорця TFi, здійснюється автоматичне перезавантаження вмісту THi у TLi. Регістри TH0 та TH1 завантажуються програмно. Перезавантаження TL0 з TH0 та TL1 з TH1 не впливає на вміст регістрів TH0 та TH1. У режимі 2 Т/Л0 та Т/Л1 також працюють абсолютно однаково.

**Режим 3.** У режимі 3 Т/Л0 і Т/Л1 працюють по-різному. Т/Л1 зберігає незмінним свій поточний вміст. Іншими словами, ефект такий самий, як і при скиданні керуючого біта TR1 в 0. У режимі 3 TL0 і TH0 функціонують як два незалежних восьмибітових лічильники. Роботу TL0 визначають керуючі біти Т/Л0 (С/Т, GATE TR0), вхідний сигнал INT0 і прапорець переповнення TF0. Роботу TH0, який може виконувати лише функції таймера (підрахунок машинних циклів мікроЕОМ), визначає керуючий біт TR1. При цьому TH0 використовує прапорець переповнення TF1. Режим 3 використовується в тих випадках, коли потрібна наявність додаткового восьмибітового таймера або лічильника подій. Можна вважати, що в цьому режимі мікроЕОМ MCS-51 має у своєму складі три таймери (лічильники). У разі ж, якщо Т/Л0 використовується в режимі 3, Т/Л1 може бути або вимкнений, або переведений у режим 0, 1 або 2, або може бути використаний послідовним портом як генератор частоти передачі.

У модернізованих моделях мікроконтролерів сімейства MCS-51 може мати третій таймер-лічильник Т/Л2 - і (або) блок програмних лічильників, які теж можуть бути використані для відліку часових інтервалів.

Таблиця 17 – Регістр керування/статусу таймера TCON

| Символ | Позиція | Ім'я і призначення   |
|--------|---------|--|
| TF1    | TCON.7  | Прапорець переповнення таймера 1. Встановлюється апаратно при переповненні таймера (лічильника). Скидається при обслуговуванні переривання апаратно    |
| TR1    | TCON.6  | Біт керування таймера 1. Встановлюється (скидається) програмою для запуску (зупинки) таймер-лічильника T/L1  |
| TF0    | TCON.5  | Прапорець переповнення таймера 0. Встановлюється апаратно при переповненні таймера (лічильника). Скидається при обслуговуванні переривання             |
| TR0    | TCON.4  | Біт управління таймера 0. Встановлюється (скидається) програмою для запуску (зупинення) таймера (лічильника) T/L0                                      |
| IE1    | TCON.3  | Прапорець фронту переривання 1. Встановлюється апаратно, коли детектується зріз зовнішнього сигналу INT1.<br>Скидається при обслуговуванні переривання |
| IT1    | TCON.2  | Біт управління типом переривання 1. Встановлюється (скидається) програмно для специфікації запиту INT1 зріз (низький рівень)                           |
| IE0    | TCON.1  | Прапорець фронту переривання 0.<br>Встановлюється по зрізу сигналу INT0.<br>Скидається при обслуговуванні переривання                                  |
| IT1    | TCON.0  | Біт управління типом переривання 0. Встановлюється (скидається) програмно для специфікації запиту INT0 зріз (низький рівень)                           |

### Система переривань мікроконтролера MCS-51

Спрощена схема переривань мікроЕОМ MCS-51 показана на рисунку 20.

У блоці реєстрів спеціальних функцій є два реєстри, призначені для управління режимом переривань: реєстр дозволу переривань ІЕ (таблиця 18) та реєстр пріоритетів переривань ІР (таблиця 19). Можливість програмної установки/скидання будь-якого керуючого біта у цих двох реєстрах робить систему переривань MCS-51 виключно гнучкою.

Зовнішні переривання сприймаються або за переходом сигналу на входах та з високого рівня у низький рівень, або за нульовим рівнем сигналу залежно від стану бітів IT0, IT1 регістра TCON. При перериванні за нульовим рівнем цей рівень повинен утримуватися не менше ніж 12 періодів сигналу тактової частоти CLK. При надходженні одного із сигналів або встановлюється прапорець IE0, або IE1 у регістрі TCON, що викликає відповідне переривання.

Скидання прапорців IE0 або IE1 здійснюється апаратно лише у тому разі, якщо переривання здійснюється за переходом з одиниці в нуль сигналу.

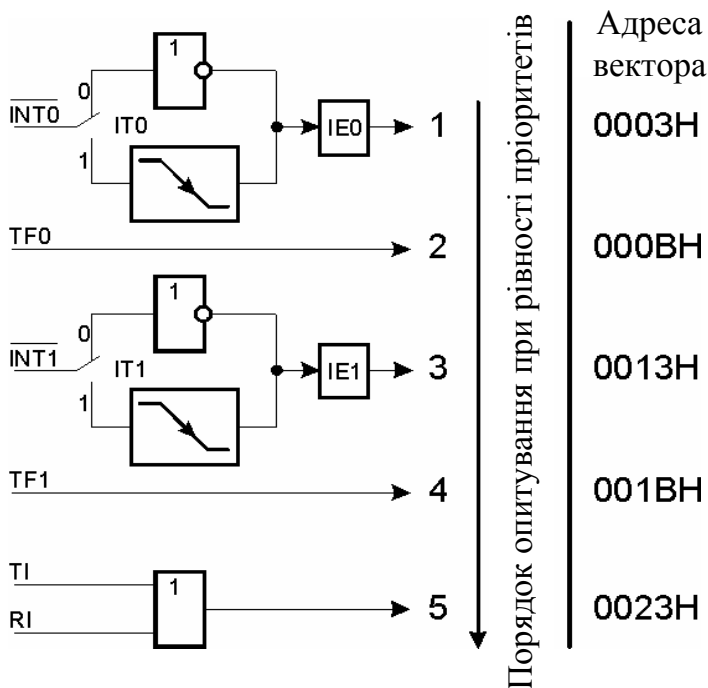


Рисунок 20 – Схема переривань

Таблиця 18 – Призначення бітів регістра ІЕ

| Символ | Позиція | Ім'я і призначення   |
|--------|---------|--|
| EA     | ІЕ.7    | Розблокування переривання. Скидається програмно для заборони всіх переривань незалежно від станів ІЕ.4 – ІЕ.0                        |
|        | ІЕ.6    | Не використовується  |
|        | ІЕ.5    | Не використовується  |
| ES     | ІЕ.4    | Біт дозволу переривання від приймача. Встановлюється (скидається) програмою для дозволу/ заборони переривань від прапорців ТІ або RІ |
|        |         | Біт дозволу переривання від таймера. Встановлюється (скидається) програмою для дозволу/ заборони переривань від таймера 1            |
| EX1    | ІЕ.2    | Біт дозволу зовнішнього переривання 1. Встановлюється (скидається) програмою для дозволу/ заборони переривання 1                     |
| ET0    | ІЕ.1    | Біт дозволу переривання від таймера 0. Встановлюється (скидається) програмою для дозволу/ заборони переривань від таймера 0          |
| EX0    | ІЕ.0    | Біт дозволу зовнішнього переривання 0. Встановлюється (скидається) програмою для дозволу/ заборони переривання 0                     |

Прапорці запитів переривання від таймерів TF0 і TF1 скидаються автоматично при передачі управління підпрограми обслуговування. Прапорці запитів переривання RІ і ТІ встановлюються блоком управління приймачем-передавачем апаратно, але скидатися повинні програмним шляхом.

Переривання можуть бути викликані або скасовані програмою, оскільки всі названі прапорці програмно доступні і можуть бути встановлені/скинуті програмою з тим самим результатом, якщо б вони були встановлені/скинуті апаратними засобами.

Таблиця 19 – Регістр пріоритетів переривань (IP)

| Символ | Позиція     | Ім'я і призначення   |
|--------|-------------|--|
| -      | IP.7 – IP.5 | Не використовується  |
| PS     | IP.4        | Біт пріоритету приймача-передавача. Встановлюється (скидається) програмою для присвоювання переривання від приймача-передавача вищого / нижчого пріоритету |
| PT1    | IP.3        | Біт пріоритету таймера 1. Встановлюється (скидається) програмою для присвоювання переривання від таймера 1 вищого / нижчого пріоритету                     |
| PX1    | IP.2        | Біт пріоритету зовнішнього переривання 1. Встановлюється (скидається) програмою для присвоювання вищого / нижчого пріоритету зовнішньому перериванню INT1  |
| PT0    | IP.1        | Біт пріоритету таймера 0. Встановлюється (скидається) програмою для присвоювання переривання від таймера 0 вищого / нижчого пріоритету                     |
| PX0    | IP.0        | Біт пріоритету зовнішнього переривання 0. Встановлюється (скидається) програмою для присвоювання вищого / нижчого пріоритету зовнішньому перериванню INT0  |

Виконання підпрограми переривання.

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

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

Відзначимо, що якщо прапорець переривання був встановлений, але за однією із зазначених вище умов не отримав обслуговування і до моменту закінчення блокування вже скинутий, то запит переривання губиться й ніде не запам'ятовується.



За апаратно сформованим кодом LCALL система переривання поміщає в стек тільки вміст лічильника команд (PC) і завантажує в нього адресу вектора відповідної підпрограми обслуговування. За адресою вектора повинна бути розташована команда безумовної передачі управління (JMP) до початкової адреси підпрограми обслуговування переривання. У разі необхідності вона повинна починатися командами запису в стек (PUSH) слова стану програми (PSW), акумулятора, розширювача, покажчика даних і т.д. і повинна закінчуватися командами відновлення зі стека (POP). Підпрограми обслуговування переривання повинні завершуватися командою RETI, за якою в лічильник команд перезавантажується зі стека збережена адреса повернення в основну програму. Команда RET також повертає керування перерваної основної програми, але при цьому не знімається блокування переривань, що приводить до необхідності мати програмний механізм аналізу закінчення процедури обслуговування даного переривання.

### **Приклад програми обробки переривання**

Виведіть на індикатор цифру «1», при надходженні переривання за входом INT0 виведіть цифру «5».

```

org 0000h
jmp start
org 0003h           ; перехід до підпрограми
                   ; обробки переривань

jmp s1
org 0100h

start:  mov ie, # 10000001b; дозвіл переривань INT0
        mov p0, # 01h
        jmp start

s1:     mov p0, # 05h      ; підпрограма обробки
                   ; переривання INT0
        reti              ; повернення до виконання
                   ; основної програми

end

```

## Приклад програми програмування таймера

Налаштуйте таймер на режим генератора з частотою 1 Гц  
(тактова частота процесора 120 кГц).

```
org 0000h
jmp start
org 000bh           ;перехід до підпрограми
                   ;обробки
                   ; Переривань від таймера

jmp s2
org 0100h
start: mov tmod, # 00000001b   ; налаштування режиму
                                           ;роботи таймера
       mov ie, # 10000010b    ;дозвіл переривань від
                                           ;таймера
       mov th0, # 0d8h        ;перед установкою
                                           ;таймера
       mov tl0, # 0f0h

       setb tcon.4            ;запуск таймера
st:    nop
       jmp st
s2:    mov th0, # 0d8h        ;підпрограма обробки
                                           ;переривання від
                                           ;таймера
       mov tl0, # 0f0h        ;перед установкою
                                           ;таймера

reti
end
```

## Завдання до лабораторної роботи

### Завдання 6.1

Завантажте проект lab\_W6 (рисунок 21).

Уведіть програму 6.1. Запустіть програму.

Перевірте результат виконання програми.

### Програма 6.1

Напишіть програму для вимірювання часу натискання кнопки, підключеної до входу зовнішнього переривання INT0.

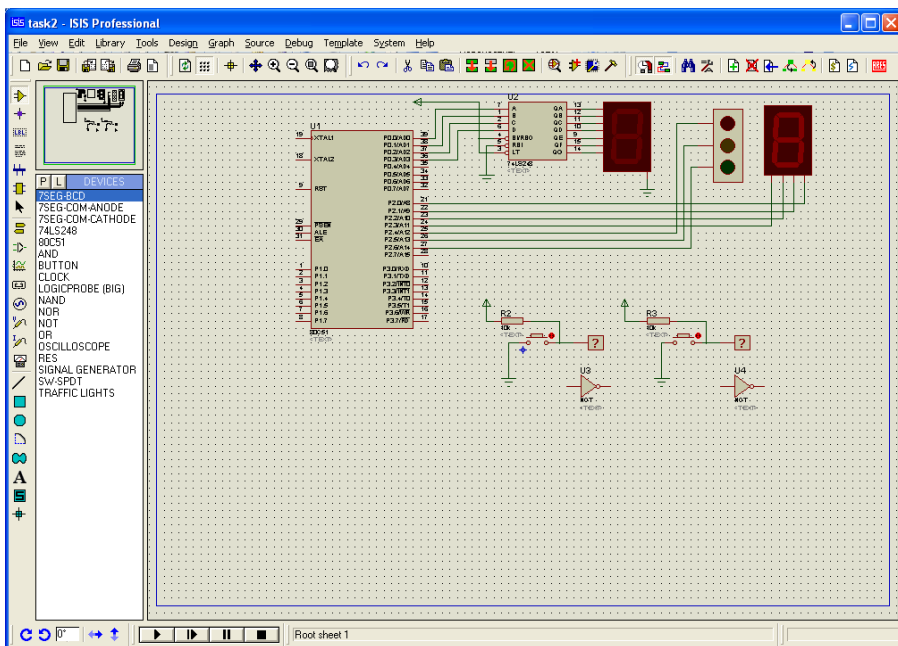


Рисунок 21 - Проект lab\_W6

### Завдання 6.2

Завантажте проект lab\_W6. Уведіть програму 6.2. Запустіть програму. Перевірте результат виконання програми.

### Програма 6.2

Напишіть програму для запуску секундоміра кнопкою, підключеною до входу зовнішнього переривання INT0, і зупинення відліку кнопкою, підключеною до входу зовнішнього переривання INT1.

### *Завдання 6.3*

Завантажте проект lab\_W6. Уведіть програму 6.3. Запустіть програму. Перевірте результат виконання програми.

### **Програма 6.3**

Напишіть програму для підрахунку кількості натискань кнопки, підключеної до входу зовнішнього переривання INT0, за 5 с.

### **Контрольні питання**

- 1 Структура системи переривання ОЕОМ MCS-51.
- 2 Режими роботи таймерів-лічильників.
- 3 Виконання підпрограми переривання.

### **Зміст звіту**

Звіт про лабораторну роботу повинен містити:

1. Мету і завдання роботи.
2. Вихідні тексти програм із доповненнями, що забезпечують тестування і налагодження.
3. Висновки з роботи.

Додаток А  
(довідковий)

**Система команд мікроконтролера сімейства MCS-51**

МікроЕОМ розглянутого сімейства є типовими мікропроцесорними пристроями з архітектурою SISC зі стандартним набором команд. Тому система команд досить обширна і включає в себе 111 основних команд. Їх довжина – один, два або три байти, причому більшість із них (94 %) – одно- або двобайтні. Всі команди виконуються за один або два машинних цикли (відповідно 1 або 2 мкс при тактовій частоті 12 МГц), виняток – команди множення і ділення, які виконуються за чотири машинних цикли (4 мкс). МікроЕОМ сімейства MCS-51 використовують пряму, безпосередню, непряму і неявну адресацію даних.

Як операнди команд мікроЕОМ сімейства MCS-51 можуть використовувати окремі біти, чотирибітові цифри, байти та двобайтні слова.

Усі ці ознаки звичні для набору команд будь-якого SISC-процесора і порівнянно з RISC - набором команд забезпечує більшу компактність програмного коду і збільшення швидкодії при виконанні складних операцій.

У той самий час набір команд сімейства MCS-51 має декілька особливостей, пов'язаних із типовими функціями, виконуваними мікроконтролерами – управлінням, для якого типовим є оперування з однорозрядними двійковими сигналами, велике число операцій введення-виведення і розгалужень програми.

Найбільш істотна особливість системи команд розглянутих мікроЕОМ – це можливість адресації окремих бітів в резидентній пам'яті даних. Крім того, як зазначалося, деякі регістри блоку регістрів спеціальних функцій також допускають адресацію окремих бітів. Карти адрес окремих бітів у резидентній пам'яті даних і в блоці регістрів спеціальних функцій.

## **Типи команд**

МікроЕОМ виконують 13 типів команд. Перший байт команди завжди містить код операції (КОП), а другий і третій (якщо вони наявні в команді) – адреси операндів або їх безпосередні значення.

## **Типи операндів**

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

Мікроконтролер має 128 програмно-керованих прапорців користувача. Є також можливість адресації окремих бітів блоку регістрів спеціальних функцій і портів. Для адресації бітів використовується пряма 8-бітова адреса (bit). Непряма адресація бітів неможлива. Карти адрес окремих бітів представлені на рисунку А1.

Чотирибітові операнди використовуються тільки при операціях обміну SWAP та XCHD.

Восьмибітовим операндом може бути осередок пам'яті програм (ПП) або даних (резидентної (РПД) або зовнішньої (ВПД)), константа (безпосередній операнд), регістри спеціальних функцій, а також порти введення / виведення. Порти і регістри спеціальних функцій адресуються тільки прямим способом. Байти пам'яті можуть адресуватися також і непрямо через адресні регістри R0, R1, DPTR і PC.

Двобайтові операнди – це константи і прямі адреси, для представлення яких використовуються другий і третій байти команди.

## **Групи команд**

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

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

| Адреса | (D <sub>7</sub> ) |    |    |    | (D <sub>0</sub> ) |    |    |    |
|--------|-------------------|----|----|----|-------------------|----|----|----|
| 7FH    |                   |    |    |    |                   |    |    |    |
| 2FH    | 7F                | 7E | 7D | 7C | 7B                | 7A | 79 | 78 |
| 2EH    | 77                | 76 | 75 | 74 | 73                | 72 | 71 | 70 |
| 2DH    | 6F                | 6E | 6D | 6C | 6B                | 6A | 69 | 68 |
| 2CH    | 67                | 66 | 65 | 64 | 63                | 62 | 61 | 60 |
| 2BH    | 5F                | 5E | 5D | 5C | 5B                | 5A | 59 | 58 |
| 2AH    | 57                | 56 | 55 | 54 | 53                | 52 | 51 | 50 |
| 29H    | 4F                | 4E | 4D | 4C | 4B                | 4A | 49 | 48 |
| 28H    | 47                | 46 | 45 | 44 | 43                | 42 | 41 | 40 |
| 27H    | 3F                | 3E | 3D | 3C | 3B                | 3A | 39 | 38 |
| 26H    | 37                | 36 | 35 | 34 | 33                | 32 | 31 | 30 |
| 25H    | 2F                | 2E | 2D | 2C | 2B                | 2A | 29 | 28 |
| 24H    | 27                | 26 | 25 | 24 | 23                | 22 | 21 | 20 |
| 23H    | 1F                | 1E | 1D | 1C | 1B                | 1A | 19 | 18 |
| 22H    | 17                | 16 | 15 | 14 | 13                | 12 | 11 | 10 |
| 21H    | 0F                | 0E | 0D | 0C | 0B                | 0A | 09 | 08 |
| 20H    | 07                | 06 | 05 | 04 | 03                | 02 | 01 | 00 |
| 1FH    | Банк 3            |    |    |    |                   |    |    |    |
| 18H    | Банк 2            |    |    |    |                   |    |    |    |
| 17H    |                   |    |    |    |                   |    |    |    |
| 10H    | Банк 1            |    |    |    |                   |    |    |    |
| 0FH    |                   |    |    |    |                   |    |    |    |
| 08H    | Банк 0            |    |    |    |                   |    |    |    |
| 07H    |                   |    |    |    |                   |    |    |    |
| 00H    |                   |    |    |    |                   |    |    |    |

*Рисунок А1 – Карта адресованих бітів у резидентній пам'яті даних*

Формат команд – одно-, дво- і трибайтові (більшість команд один або два байтові). Перший байт будь-якого типу і формату завжди містить код операції, другий і третій байти містять або адреси операндів, або безпосередні операнди.

Склад операндів включає в себе операнди чотирьох типів: біти, цифрові (4 розряди), байти і 16-бітові слова. Час виконання команд становить 1, 2 або 4 машинних цикли. При тактовій частоті 12 мГц тривалість машинного циклу становить 1 мкс.

Набір команд MCS-51 підтримує такі режими адресації.

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

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

*Реєстрова адресація (Register Instruction).* Дана адресація застосовується для доступу до реєстрів R0 + R7 вибраного банку. Команди реєстрової адресації містять у байті коду операції трибітового поля, що визначає номер реєстра. Вибір одного із чотирьох реєстрових банків здійснюється програмуванням бітів селектора банку (RS1, RS0) в PSW.

*Безпосередня адресація (Immediate constants).* Операнд міститься безпосередньо в полі команди услід за кодом операції і може займати один або два байти (data8, data16).

*Індексна адресація (Indexed Addressing).* Індексна адресація використовується при зверненні до пам'яті програм і тільки при читанні. У цьому режимі здійснюється перегляд таблиць у пам'яті програм. 16-бітовий реєстр (DPTR або PC) вказує базову адресу необхідної таблиці, а акумулятор вказує на точку входу в неї. Адреса елемента таблиці знаходиться складанням бази з індексом (вмістом акумулятора).



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

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

### **Позначення, які використовуються при описі команд**

Rn (n = 0, 1, ..., 7) – регістр загального призначення в обраному банку регістрів;

@ Ri (i = 0, 1) – регістр загального призначення в обраному банку регістрів, який використовується як регістр непрямої адреси;

ad – байт із РПД за прямою адресацією (у команді зазначається або адреса байта – 00-FF, або позначення одного з регістрів спеціальних функцій SFR, наприклад, TH0, P1, TCON, SBUF);

ads – байт-джерело із прямою адресацією;

add – байт-приймач із прямою адресацією;

ad11 – 11-розрядна абсолютна адреса переходу;

ad16 – 16-розрядна абсолютна адреса переходу;

rel – відносна адреса переходу;

# D – безпосередній операнд;

# D16 – безпосередній операнд (2 байти);

bit – біт із прямою адресацією;

A – акумулятор;

PC – лічильник команд;

DPTR – регістр-вказівник даних;

() – вміст комірки пам'яті або регістра.

### **Команди пересилання даних мікроконтролера MCS-51**

Ця група представлена 28 командами, їх короткий опис наведено в таблиці, де також зазначений тип команди (Т) відповідно до таблиці, її довжина в байтах (Б) і час виконання в машинних циклах (Ц) (таблиця А1).

За командою MOV виконується пересилання даних із другого операнда в перший. Ця команда не має доступу ні до зовнішньої пам'яті даних, ні до пам'яті програм. Для цих цілей призначені команди MOVX і MOVC відповідно. Перша з них забезпечує читання/запис байтів із зовнішньої пам'яті даних, друга – читання байтів з пам'яті програм.

За командою XCH виконується обмін байтами між акумулятором і осередком РПД, а за командою XCHD – обмін молодшої тетради акумулятора з молодшою тетрадою байта РПД.

Команди PUSH і POP призначені відповідно для запису даних у стек і їх читання зі стека. Розмір стека обмежений лише розміром резидентної пам'яті даних. У процесі ініціалізації мікроЕОМ після сигналу скидання або при включенні напруги живлення в SP заноситься код 07H. Це означає, що перший елемент стека буде розміщуватися в комірці пам'яті з адресою 08H.

Група команд пересилання даних мікроконтролера має таку особливість – у ній немає спеціальних команд для роботи зі спеціальними регістрами: PSW, таймером, портами вводу-виводу. Доступ до них, як і до інших регістрів спеціальних функцій, здійснюється завданням відповідної прямої адреси, тобто це команди звичайних пересилань, в яких замість адреси можна ставити назву відповідного регістра. Наприклад, читання PSW в акумулятор може бути виконано командою

MOV A, PSW,

яка перетворюється асемблером до вигляду

MOV A, 0D0h (E5 D0),

де E5 – код операції, а D0 – операнд (адреса PSW).

Крім того, слід зазначити, що в мікроЕОМ акумулятор має два різних імені залежно від способу адресації: A – при неявній адресації (наприклад, MOV A, R0) і ACC – при використанні прямої адреси.

Таблиця А.1 – Команди передачі даних

| Команда            | Функція  |
|--------------------|--|
| MOV A, Rn          | Пересилання в акумулятор з регістра (n = 0...7)                  |
| MOV A, ad          | Пересилання в акумулятор байта за прямою адресацією              |
| MOV A, @Ri         | Пересилання в акумулятор байта з РПД (i = 0,1)                   |
| MOV A, #D8         | Завантаження в акумулятор константи                              |
| MOV Rn, A          | Пересилання в регістр (n = 0...7) з акумулятора                  |
| MOV Rn, ad         | Пересилання в регістр байта за прямою адресацією                 |
| MOV Rn, #D8        | Завантаження в регістр (n = 0...7) константи                     |
| MOV ad, A          | Пересилання за прямою адресацією акумулятора                     |
| MOV ad, Rn         | Пересилання за прямою адресацією регістра                        |
| MOV add, ads       | Пересилання байта за прямою адресацією                           |
| MOV ad, @Ri        | Пересилання байта з РПД за прямою адресацією                     |
| MOV ad, #D8        | Пересилання за прямою адресацією константи                       |
| MOV @Ri, A         | Пересилання в РПД з акумулятора                                  |
| MOV @Ri, ad        | Пересилання в РПД байта за прямою адресацією                     |
| MOV @Ri, #D8       | Пересилання в РПД константи                                      |
| MOV DPTR, #D16     | Завантаження регістра-показчика даних                            |
| MOVC<br>A, @A+DPTR | Пересилання в акумулятор байта з ПП                              |
| MOVX A, @Ri        | Пересилання в акумулятор байта з ВПД                             |
| MOVX A, @DPTR      | Пересилання в акумулятор байта з розширеної ВПД                  |
| MOVX @Ri, A        | Пересилання у ВПД з акумулятора                                  |
| MOVX @DPTR, A      | Пересилання в розширену ВПД з акумулятора                        |
| PUSH ad            | Завантаження в стек  |
| POP ad             | Витягування зі стека   |
| XCH A, Rn          | Обмін акумулятора з регістром                                    |
| XCH A, ad          | Обмін акумулятора і байта з прямою адресацією                    |
| XCH A, @Ri         | Обмін акумулятора і байта з РПД                                  |
| XCHD A, @Ri        | Обмін молодшої тетради акумулятора з молодшою тетрадою байта РПД |
| SWAP A             | Обмін місцями тетрад в акумуляторі                               |

## Команди арифметичних операцій MCS-51

До цієї групи входять 24 команди, короткий опис яких наведено в таблиці А.2.

*Таблиця А.2 – Команди арифметичних операцій*

| Команда     | Функція  |
|-------------|--|
| ADD A, Rn   | Додавання акумулятора з регістром (n = 0...7)  |
| ADD A, ad   | Додавання акумулятора і байта з прямою адресацією  |
| ADD A, @Ri  | Додавання акумулятора і байта з РПД (i=0,1)  |
| ADD A, #D8  | Додавання акумулятора і константи  |
| ADDC A, Rn  | Додавання акумулятора і регістра із перенесенням   |
| ADDC A, ad  | Додавання акумулятора і байта з прямою адресацією із перенесенням                                |
| ADDC A, @Ri | Додавання акумулятора і байта із РПД із перенесенням   |
| ADDC A, #D8 | Додавання акумулятора з константою і перенесенням  |
| DA A        | Десяткова корекція акумулятора (після додавання операндів, представлених в упакованому BCD-коді) |
| SUBB A, Rn  | Віднімання від акумулятора регістра і позики   |
| SUBB A, ad  | Віднімання від акумулятора байта із прямою адресацією і позикою                                  |
| SUBB A, @Ri | Віднімання з акумулятора байта РПД і позики  |
| SUBB A, #D8 | Віднімання з акумулятора константи і позики  |
| INC A       | Інкремент акумулятора  |
| INC Rn      | Інкремент регістра   |
| INC ad      | Інкремент байта із прямою адресацією   |
| INC @Ri     | Інкремент байта в РПД  |
| INC DPTR    | Інкремент покажчика даних  |
| DEC A       | Декремент акумулятора  |
| DEC Rn      | Декремент регістра   |
| DEC ad      | Декремент байта із прямою адресацією   |
| DEC @Ri     | Декремент байта в РПД  |
| MUL AB      | Множення вмісту акумулятора на вміст регістра B  |
| DIV AB      | Ділення вмісту акумулятора на вміст регістра B   |

Отже, МікроЕОМ виконує досить широкий набір команд для організації обробки цілочислових даних, включаючи команди множення і ділення.

За результатом виконання команд ADD, ADDC, SUBB, MUL і DIV встановлюються прапорці в регістрі PSW.

Прапорець C встановлюється при перенесенні з розряду D7, тобто у випадку, якщо результат не вміщується у вісім розрядів; Прапорець AC встановлюється при перенесенні з розряду D3 в командах додавання і віднімання і служить для реалізації десяткової арифметики. Ця ознака використовується командою DAA.

Прапорець OV встановлюється при перенесенні з розряду D6, тобто у випадку, якщо результат не поміщається в сім розрядів і восьмий не може бути інтерпретовано як знаковий. Ця ознака служить для організації обробки чисел зі знаком.

Нарешті, Прапорець P встановлюється та скидається апаратно. Якщо число одиничних бітів в акумуляторі непарне, то  $P = 1$ , інакше  $P = 0$ .

### **Команди логічних операцій мікроконтролера MCS-51**

У цій групі 25 команди, короткий опис яких наведено в таблиці А.3. Ці команди дозволяють виконувати операції над байтами: логічне І ( $\wedge$ ), логічне АБО ( $\vee$ ), що виключає АБО ( $\oplus$ ), інверсію (NOT), скидання в нульове значення і зсув.

### **Команди операцій над бітами мікроконтролера MCS-51**

Група складається з 12 команд, короткий опис яких наведено в таблиці А.4. Ці команди дозволяють виконувати операції над окремими бітами: скидання, установку, інверсію біта, а також логічні І ( $\wedge$ ) і АБО ( $\vee$ ). Як "логічний" акумулятор, який бере участь у всіх операціях із двома операндами, виступає ознака перенесення C (розряд D7 PSW), як операнди можуть використовуватися 128 бітів спектра поліграфічної резидентної пам'яті даних і регістри спеціальних функцій, що допускають адресацію окремих бітів.

*Таблиця А.3 – Команди логічних операцій*

| Команда     | Функція  |
|-------------|--|
| ANL A, Rn   | Логічне І акумулятора і регістра                       |
| ANL A, ad   | Логічне І акумулятора і байта із прямою адресацією     |
| ANL A, @Ri  | Логічне І акумулятора і байта з РПД                    |
| ANL A, #D8  | Логічне І акумулятора і константи                      |
| ANL ad, A   | Логічне І байта із прямою адресацією й акумулятора     |
| ANL ad, #D8 | Логічне І байта із прямою адресацією і константи       |
| ORL A, Rn   | Логічне АБО акумулятора і регістра                     |
| ORL A, ad   | Логічне АБО акумулятора і байта із прямою адресацією   |
| ORL A, @Ri  | Логічне АБО акумулятора і байта із РПД                 |
| ORL A, #D8  | Логічне АБО акумулятора і константи                    |
| ORL ad, A   | Логічне АБО байта із прямою адресацією й акумулятора   |
| ORL ad, #D8 | Логічне АБО байта із прямою адресацією і константою    |
| XRL A, Rn   | Виключне АБО акумулятора і регістра                    |
| XRL A, ad   | Виключне АБО акумулятора і байта із прямою адресацією  |
| XRL A, @Ri  | Виключне АБО акумулятора і байта РПД                   |
| XRL A, #D8  | Виключне АБО акумулятора і константи                   |
| XRL ad, A   | Виключне АБО байта із прямою адресацією та акумулятора |
| XRL ad, #D8 | Виключне АБО байта із прямою адресацією і константи    |
| CLR A       | Скидання акумулятора                                   |
| CPL A       | Інверсія акумулятора                                   |
| RL A        | Циклічний зсув ліворуч                                 |
| RLC A       | Зсув ліворуч через перенесення                         |
| RR A        | Циклічний зсув праворуч                                |
| RRC A       | Зсув праворуч через перенесення                        |

*Таблиця А.4 – Команди операцій над бітами*

| Команда     | Функція                                 |
|-------------|---|
| CLR C       | Скидання перенесення                    |
| CPL C       | Інверсія перенесення                    |
| SETB C      | Установка перенесення                   |
| CLR bit     | Скидання біта                           |
| CPL bit     | Інверсія біта                           |
| SETB bit    | Установка біта                          |
| ANL C, bit  | Логічне І перенесення біта              |
| ANL C, /bit | Логічне І перенесення й інверсії біта   |
| ORL C, bit  | Логічне АБО перенесення біта            |
| ORL C, /bit | Логічне АБО перенесення й інверсії біта |
| MOV C, bit  | Пересилання біта в перенесення          |
| MOV bit, C  | Пересилання перенесення в біт           |

### **Команди передачі управління мікроконтролера MCS-51**

Група представлена командами безумовного і умовного переходів, командами виклику підпрограм і командами повернення спектр у поліграфічних підпрограм (таблиця А.5).

Команда безумовного переходу LJMP (L – long – довгий) здійснює перехід за абсолютною 16-бітною адресою, зазначеною в тілі команди, тобто команда забезпечує перехід у будь-яку точку пам'яті програм.

Дія команди AJMP (A – absolute – абсолютний) аналогічна команді LJMP, проте в тілі команди зазначені лише 11 молодших розрядів адреси. Тому перехід здійснюється в межах сторінки розміром 2 Кбайт, при цьому треба мати на увазі, що спочатку вміст лічильника команд збільшується на 2 і тільки потім замінюються 11 розрядів адреси.

На відміну від попередніх команд у команді SJMP (S – short – короткий) зазначений не абсолютна, а відносна адреси переходу. Величина зміщення REI розглядається як число зі знаком, а отже, перехід можливий у межах: - 128 –. +127 байт щодо адреси команди, наступної за командою SJMP.

Таблиця А.5 – Команди передачі управління

| Команда            | Функція  |
|--------------------|--|
| LJMP ad16          | Довгий перехід у межах всієї пам'яті програм                                     |
| AJMP ad11          | Абсолютний перехід у межах сторінки 2 Кб   |
| SJMP rel           | Короткий відносний перехід у межах сторінки 256 байтів                           |
| JMP @A + DPTR      | Непрямий перехід   |
| JC rel             | Перехід, якщо перенесення дорівнює одиниці                                       |
| JNC rel            | Перехід, якщо перенесення дорівнює нулю  |
| JZ rel             | Перехід, якщо акумулятор дорівнює нулю   |
| JNZ rel            | Перехід, якщо акумулятор не дорівнює нулю  |
| JB bit, rel        | Перехід, якщо біт дорівнює одиниці   |
| JBC bit, rel       | Перехід, якщо біт установлений, із наступним скиданням біта                      |
| JNB bit, rel       | Перехід, якщо біт дорівнює нулю  |
| DJNZ Rn, rel       | Декремент регістра і перехід, якщо не дорівнює нулю                              |
| DJNZ ad, rel       | Декремент байта із прямою адресацією і перехід, якщо не нуль                     |
| CJNE A, ad, rel    | Порівняння акумулятора з байтом із прямою адресацією і перехід, якщо не дорівнює |
| CJNE A, #D8, rel   | Порівняння акумулятора з константою і перехід, якщо не дорівнює                  |
| CJNE Rn, #D8, rel  | Порівняння регістра з константою і перехід, якщо не дорівнює                     |
| CJNE @Ri, #D8, rel | Порівняння байта в РПД із константою і перехід, якщо не дорівнює                 |
| LCALL ad16         | Довгий виклик підпрограми  |
| ACALL ad11         | Абсолютний виклик підпрограми у межах сторінки в 2 Кбайта                        |
| RET                | Повернення з підпрограми   |
| RETI               | Повернення з підпрограми обробки переривання                                     |
| NOP                | Порожня команда  |



Команда непрямого переходу `JMP @ A + DPTR` дозволяє обчислювати адреси переходу в процесі виконання самої програми.

Усі команди умовного переходу, розглянуті мікроЕОМ, містять коротку відносну адресу, тобто перехід може здійснюватися в межах 128 – 127 байтів від наступної команди.

Команда `DJNZ` призначена для організації програмних циклів. Регістр `Rn` або байт за адресою `ad`, зазначені в тілі команди, містять лічильник повторень циклу, а зсув `rel` – відносна адреса переходу до початку циклу. При виконанні команди вміст лічильника зменшується на 1 і перевіряється на 0. Якщо значення вмісту лічильника не дорівнює 0, то здійснюється перехід на початок циклу, в іншому випадку виконується наступна команда.

Команда `CJNE` зручна для реалізації процедур очікування зовнішніх подій. У тілі команди зазначені "координати" двох байтів і відносна адреса переходу `rel`.

Навчальне видання

**МЕТОДИЧНІ ВКАЗІВКИ**  
до лабораторних робіт  
із дисципліни «Мікропроцесорна техніка»  
для студентів напряму підготовки  
"Електротехніка та електротехнології"  
денної форми навчання

Відповідальний за випуск А. С. Опанасюк  
Редактор Н. В. Лисогуб  
Комп'ютерне верстання І. Є. Бражник

Підп. до друку 17.03.2017, поз.  
Формат 60x84/16. Ум. друк. арк. Обл.-вид.арк. Тираж 40 пр. Зам. №  
Собівартість видання грн к.

Видавець і виготовлювач  
Сумський державний університет,  
вул. Р.- Корсакова, 2, м. Суми, 40007  
Свідоцтво суб'єкта видавничої справи ДК №3062 від 17.12.2007.