МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ”

для студентів спеціальності
141 – Електроенергетика, електротехніка та електромеханіка
освітня програма «Електромеханічні системи автоматизації та електропривод»
усіх форм навчання

Частина ІІ

2022

Конспект лекцій з дисципліни «Основи мікропроцесорної техніки» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітня програма «Електромеханічні системи автоматизації та електропривод» усіх форм навчання. Частина ІІ. /Укл: О.С. Назарова, В.В. Осадчий – Запоріжжя: НУ «Запорізька політехніка», 2022. – 38 с.

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

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

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

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 05 від 16.12.2021 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 05 від 23.12.2021 р.

ЗМІСТ

ПЕРЕДМОВА

Конспект лекцій містить матеріали з вивчення дисципліни «Основи мікропроцесорної техніки (ОМПТ)» у відповідності до навчальних планів ОКР бакалаврів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електромеханічні системи автоматизації та електропривод».

Мета навчити студентів застосовувати основи мікропроцесорної техніки, функціональні можливості мікропроцесорних систем і призначення пов'язаних з ними об'єктів при виконанні інженерних завдань за фахом.

Завдання сформувати у студентів знання, вміння та навички, необхідні для розуміння питань щодо програмування МК-51, ADuC 841, здійснення перевірки розроблених програм за допомогою емулятора Franklin Software, програми «WDS Analog Devices».

Для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електромеханічні системи автоматизації та електропривод» усіх форм навчання.

1 КОМАНДИ ЛОГІЧНИХ ОПЕРАЦІЙ

У дискретній і цифровій схемотехніці широко використовуються функції заперечення, кон'юнкції, заперечення кон'юнкції, диз'юнкції, заперечення диз'юнкції й додавання за модулем два.

Для реалізації цих функцій розроблені й випускаються інтегральні мікросхеми. Їх часто називають базовими логічними елементами «НІ», «І», «І – НІ», «АБО», «АБО – НІ», а пристрої, синтезовані на їхній основі, називають комбінаційними автоматами або пристроями комбінаційного типу [1-3].

У таблиці 1.1 наведено узагальнені відомості про позначення і виконання логічних операцій, таблиці істинності, електричні та комбінаційні схеми [8,17] і команди.

Пріоритети дій в порядку спадання:

- дії в дужках;
- інверсія;
- множення, ділення;
- логічні дії (лог. множення, лог.додавання, що виключає АБО);
- Додавання, віднімання

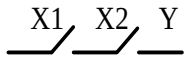
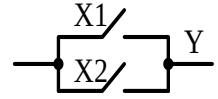
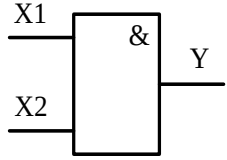
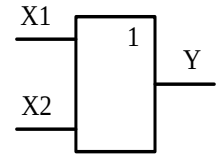
У разі рівного розподілу пріоритетів дії виконуються зліва на право.

Приклад 1.1.
Логічні операції.

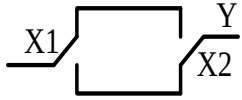
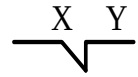
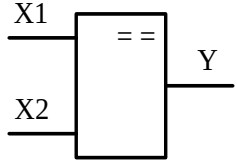
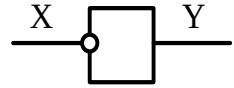
$$(20H) * \overline{(21H) - 10H} \forall 45H \rightarrow (23H)$$

```
Org 0h
Mov a, 21h          ;(21H)→(A)
Clr c               ; (C)=0
Subb a, #10h        ;(A)-10H-(C)→(A)
Cpl a               ; (Ā) → (A)
Mov b, 20h          ;(20H)→(B)
Mul ab              ;(A)*(B) →(B).(A)
Xrl a,#45H          ; (A)∀45H → (A)
mov 23h, A          ;(A) →(23H)
L1: JMP L1
End
```

Таблиця 1.1 – Узагальнені відомості про позначення і виконання логічних операцій і команд

<table>
<tr><td>Назва</td><td>Логічне множення, кон’юнкція,
«логічне І»</td><td>Логічне додавання,
диз’юнкція,
«логічне АБО»</td></tr>
<tr><td>Позначення</td><td>AND</td><td>OR</td></tr>
<tr><td>Графічне зображення</td><td>∧</td><td>∨</td></tr>
<tr><td>Таблиця істинності (правило)</td><td>
<table>
<tr><th>X1</th><th></th><th>X2</th><th></th><th>Y</th></tr>
<tr><td>0</td><td>∧</td><td>0</td><td>=</td><td>0</td></tr>
<tr><td>0</td><td>∧</td><td>1</td><td>=</td><td>0</td></tr>
<tr><td>1</td><td>∧</td><td>0</td><td>=</td><td>0</td></tr>
<tr><td>1</td><td>∧</td><td>1</td><td>=</td><td>1</td></tr>
</table>
</td><td>
<table>
<tr><th>X1</th><th></th><th>X2</th><th></th><th>Y</th></tr>
<tr><td>0</td><td>∨</td><td>0</td><td>=</td><td>0</td></tr>
<tr><td>0</td><td>∨</td><td>1</td><td>=</td><td>1</td></tr>
<tr><td>1</td><td>∨</td><td>0</td><td>=</td><td>1</td></tr>
<tr><td>1</td><td>∨</td><td>1</td><td>=</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Команда (мнемоніка)</td><td>ANL A, Rn; (A) ∧ (Rn) →(A)
ANL A, #d8; (A) ∧ d8→ (A)
ANL A, dir; (A) ∧ (dir) →(A)
ANL dir, A; (dir) ∧ (A)→(dir)
ANL dir, #d8; (dir) ∧ d8→(dir)
ANL A, @Ri; (A) ∧ ((Ri))→(A)
ANL C, bit;(C) ∧ (bit) →(C)
ANL C, /bit;(C) ∧ (bit) →(C)</td><td>ORL A, Rn; (A) ∨ (Rn) →(A)
ORL A, #d8; (A) ∨ d8→ (A)
ORL A, dir; (A) ∨ (dir) →(A)
ORL dir, A; (dir) ∨ (A)→(dir)
ORL dir, #d8; (dir) ∨ d8→(dir)
ORL A, @Ri; (A) ∨ ((Ri))→(A)
ORL C, bit;(C) ∨ (bit) → (C)
ORL C, /bit;(C) ∨ (bit) → (C)</td></tr>
<tr><td>Електрична схема</td><td> 
</td><td> 
</td></tr>
<tr><td>Комбінаційна схема</td><td> 
</td><td> 
</td></tr>
</table>

Продовження таблиці 1.1

<table>
<tr><td>Назва</td><td>Логічне виключення,
додавання по модулю 2,
«виключне АБО»</td><td>Логічне
заперечення,
інверсія,
«логічне НЕ»</td></tr>
<tr><td>Позначення</td><td>XOR</td><td>NOT</td></tr>
<tr><td>Графічне зображення</td><td>$\forall$</td><td>$\overline{X}$</td></tr>
<tr><td>Таблиця істинності (правило)</td><td>
<table>
<tr><th>X1</th><th></th><th>X2</th><th></th><th>Y</th></tr>
<tr><td>0</td><td>$\forall$</td><td>0</td><td>=</td><td>0</td></tr>
<tr><td>0</td><td>$\forall$</td><td>1</td><td>=</td><td>1</td></tr>
<tr><td>1</td><td>$\forall$</td><td>0</td><td>=</td><td>1</td></tr>
<tr><td>1</td><td>$\forall$</td><td>1</td><td>=</td><td>0</td></tr>
</table>
</td><td>
<table>
<tr><th>X</th><th></th><th>Y</th></tr>
<tr><td>$\overline{0}$</td><td>=</td><td>1</td></tr>
<tr><td>$\overline{1}$</td><td>=</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>Команда (мнемоніка)</td><td>XRL A, Rn; (A) $\forall$ (Rn) →(A)
XRL A, #d8; (A) $\forall$ d8→ (A)
XRL A, dir; (A) $\forall$ (dir) →(A)
XRL dir, A; (dir) $\forall$ (A)→(dir)
XRL dir, #d8; (dir) $\forall$ d8→(dir)
XRL A, @Ri; (A) $\forall$ ((Ri)) → (A)</td><td>CPL A; ($\overline{A}$) → (A)
CPL C; ($\overline{C}$) → (C)
CPL bit;
$\overline{(bit)}$ → (bit)</td></tr>
<tr><td>Електрична схема</td><td> 
</td><td> 
</td></tr>
<tr><td>Комбінаційна схема</td><td> 
</td><td> 
</td></tr>
</table>

2 КОМАНДИ ОПЕРАЦІЙ З БІТАМИ

Відмінною особливістю цієї групи команд є те, що вони оперують з однобітними операндами. В якості таких операндів виступають 128 програмних прапорів користувача, а також біти деяких регістрів спеціальних функцій [6,7].

Для адресації бітів використовується пряма 8-ми розрядна адреса (bit).

Для обміну інформацією з зовнішніми пристроями (кнопки, датчики, виконавчі механізми та ін.) в мікроконтролерах серії MSC-51 є чотири восьмирозрядних порти P0, P1, P2, P3 (всього 32 цифрових входи / виходи). Їх біти мають адреси, тобто можуть виступати в якості операндів бітових операцій [9,10].

Скидання в «нуль»

CLR C; (C) ← 0

CLR bit; (bit) ← 0

Приклад: Запалити світодіод, підключений до P0.2 (другого біту нульового порту).

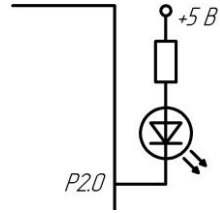


Рисунок 2.1 – Апаратна частина

Струм тече – світодіод горить.

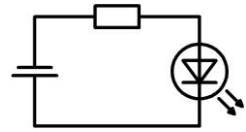


Рисунок 2.2 - Еквівалентна схема

Програмна частина.
CLR 82H ↔ CLR P0.2

Установка в «одиницю»
SETB C; (C) ← 1
SETB bit; (bit) ← 1

Приклад: Погасити світодіод, підключений до P2.0 (нульового біту другого порту).

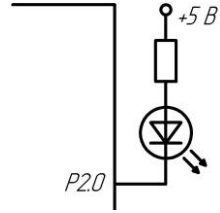


Рисунок 2.3 – Апаратна частина

Струм не тече – світлодіод не горить.

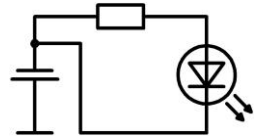


Рисунок 2.4 – Еквівалентна схема

Програмна частина.
SETB 0A0H ↔ SETB P2.0

Переміщення (копіювання)
MOV C, bit; (bit) → (C)
MOV bit, C; (C) → (bit)

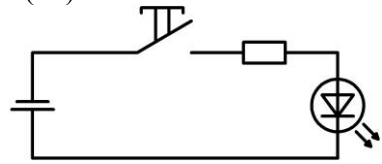


Рисунок 2.5 – Еквівалентна схема

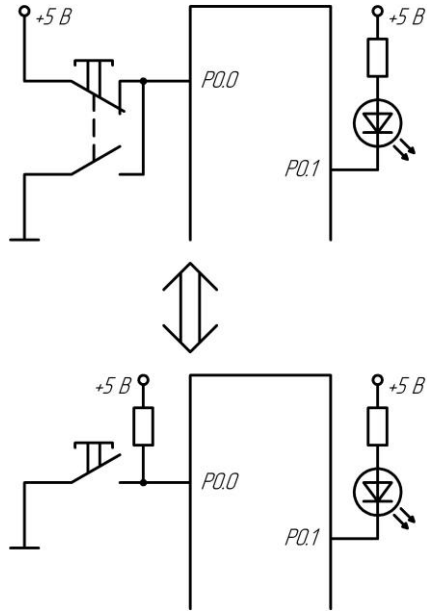


Рисунок 2.6 – Апаратна частина

Програмна частина.

```
SETB   P0.0; налаштування біта до вводу
L1: MOV C, P0.0
      MOV P0.1, C
      JMP  L1
```

Інвертування

CPL C; (C) → (C)

CPL bit; (bit) → (bit)

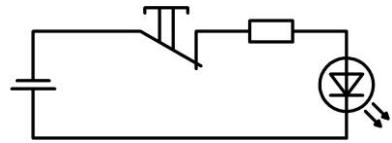


Рисунок 2.7 – Еквівалентна схема

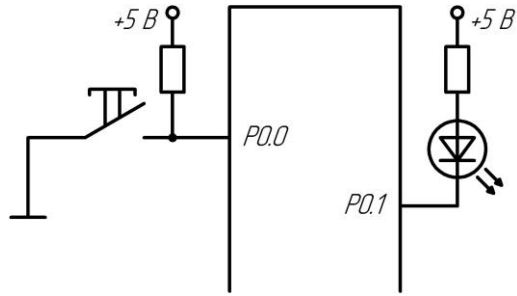


Рисунок 2.8 – Апаратна частина

Програмна частина.

```
      SETB P0.0
L1:  MOV C,  P0.0
      CPL  C
      MOV P0.1, C
      JMP  L1
```

Логічне множення (логічне «І»)

ANL C, bit; (C) /\ (bit) → (C)

ANL C, / bit; (C) /\ (bit) → (C)

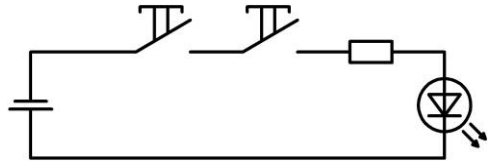


Рисунок 2.9 – Еквівалентна схема

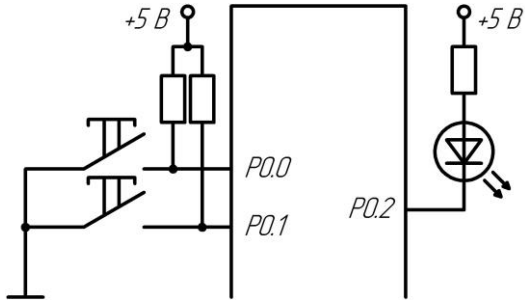


Рисунок 2.10 – Апаратна частина

Програмна частина.

```
    SETB P0.0
    SETB P0.1
L1:MOV C, P0.0
    CPL C
    ANL C, P0.1
    CPL  C
    MOV P0.2, C
    JMP  L1
```

Логічне додавання (логічне «АБО»)

ORL C, bit; (C) V (bit) → (C)

ORL C, /bit; (C) V (bit) → (C)

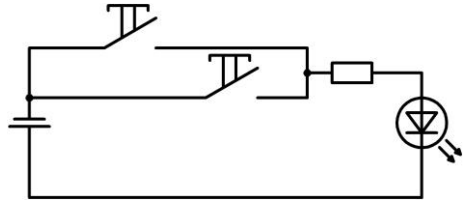


Рисунок 2.11 – Еквівалентна схема

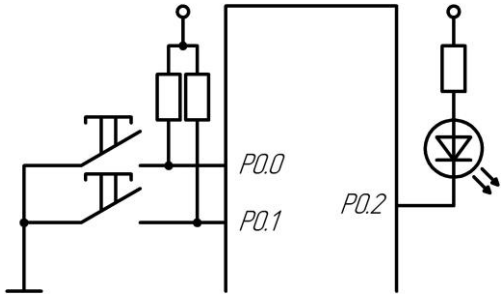


Рисунок 2.12 – Апаратна частина

Програмна частина.

```
   SETB  P0.0
   SETB  P0.1
L1:MOV C, P0.0
   CPL C
```

```
ORL C, P0.1
CPL C
MOV P0.2, C
JMP L1
```

Приклад 2.1.

Згідно заданої функції алгебри логіки скласти:

- комбінаційну схему;
- електричну схему;
- таблицю істинності для всіх можливих комбінацій бітів порта P2;
- написати програму.

Задано функію алгебри логіки:

$\overline{\overline{P2.1 \wedge P2.7} \vee P2.3} \rightarrow P0.0$.

Складаємо за цією функцією комбінаційну схему.

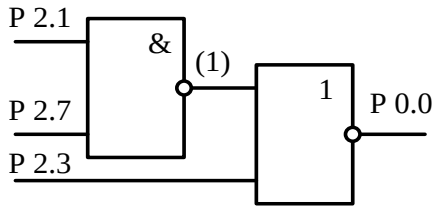


Рисунок 2.13 – Комбінаційна схема, що відповідає заданій функції алгебри логіки.

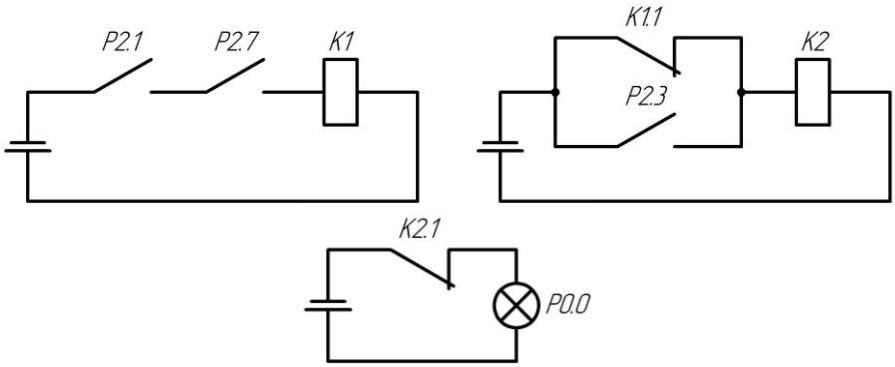


Рисунок 2.14 – Електрична схема, що відповідає заданій функції алгебри логіки.

Таблиця 2.1 - Таблиця істинності для всіх можливих комбінацій бітів порта P2.

P2.1	P2.7	P2.3	(1)	P0.0
0	0	0	0(1)	1(0)
0	0	1	0(1)	1(0)
0	1	0	0(1)	1(0)
0	1	1	0(1)	1(0)
1	0	0	0(1)	1(0)
1	0	1	0(1)	1(0)
1	1	0	1(0)	0(1)
1	1	1	1(0)	1(0)

Програмна реалізація завдання.

```
ORG 0H

L1:   MOV C,P2.1
      ANL C,P2.7
      CPL C
      ORL C, P2.3
      CPL C
      MOV P0.0,C
      JMP L1
      END
```

При перевірці працездатності програми, у емуляторі треба відкрити додаткове обладнання, для цього зайти у меню View / Hardware / Port0, а потім так само відкрити Port2 (рис. 2.15).

Імітація зміни стану певного біту порта відбувається шляхом натискання лівої клавіші миші на віконечку біля цього біту порта, при цьому з'являється меню, у якому Ground змінює стан біта з 1 на 0 (рис. 2.16).

Таким чином можна за складеними схемами і теблицею істинності перевірити правильність виконання завдання для всіх можливих комбінацій бітів, які використані у завданні.

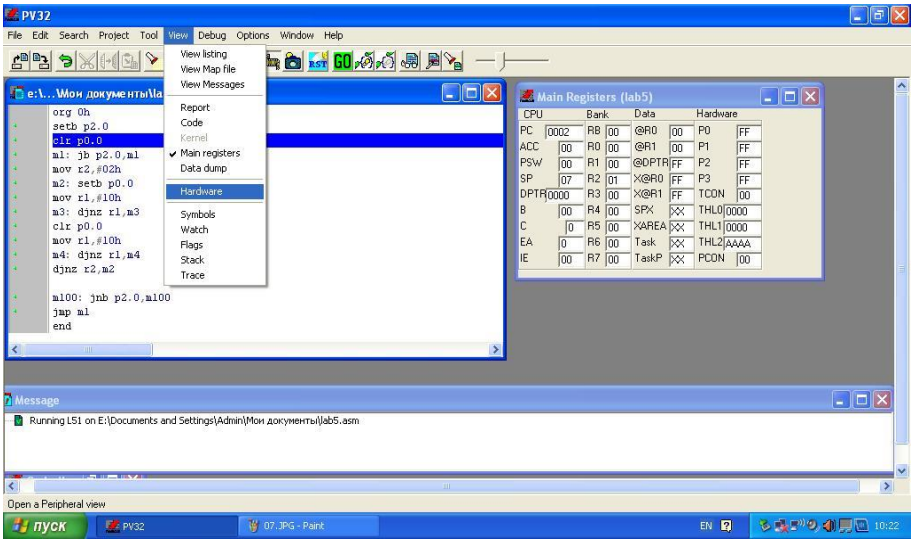


Рисунок 2.15 – Скрін екрану при відкритті додаткового обладнання у вигляді портів P0, P2.

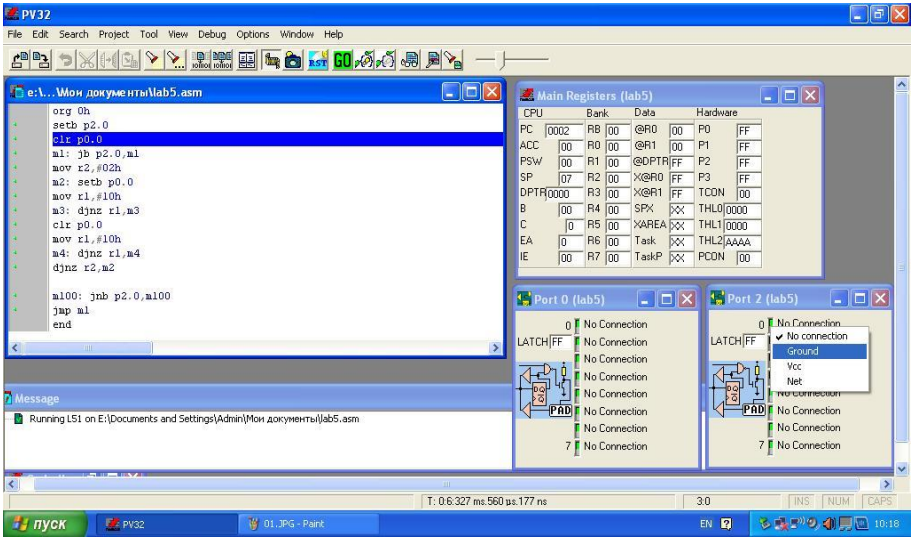


Рисунок 2.16 – Скрін екрану при перевірці працездатності програми

3 РОЗГАЛУДЖЕНІ СТРУКТУРИ. ЦИКЛИ

До даної групи належать команди переходу, які забезпечують умовне і безумовне розгалуження, виклик і повернення з підпрограми, а також команда порожній операції NOP. У більшості цих команд використовується пряма адресація, адреса переходу цілком або частину міститься в самій команді [1,3,5].

Можна виділити три різновиди командного розгалуження по розрядності зазначених вище адреси переходу.

3.1 Команди безумовного переходу

Довгий перехід - перехід по всьому адресному простору пам'яті програм, тобто в команді міститься повний розрядний (16-бітний) адреса переходу (ad 16) - 3 байта.

Трьохбайтні команди довгого переходу містять в Мнемокоді літеру L (Long).

Всього існує дві такі команди:

LJMP - довгий перехід

LCALL - довгий виклик підпрограми.

На практиці рідко виникає необхідність переходу в межах всього адресного простору і частіше використовуються укорочені команди переходу, що займає менше місця в пам'яті.

LJMP ad16; (PC) ← ad16 (перехід на ad16),

де PC - лічильник команд, в якому зберігається адреса наступної команди програми, яка буде виконуватися процесором, ad16 - повний 16-бітову адресу переходу.

Абсолютний перехід - перехід в межах однієї сторінки пам'яті програм розміром 2048 байт. Такі команди містять тільки 11 молодших біт адреси переходу (ad11). Команди абсолютного переходу мають формат 2 байта.

Початкова літера мнемокода - A (Absolute). При виконанні команди в розрахунковій адресі наступної по порядку команди ((PC) = (PC) + 2)

11 молодших біт замінюються на ad11 з тіла команди абсолютного переходу.

AJMP ad11; (PC) ← (PC) + 2, (PC0-10) ← ad11 (перехід на ad11), де ad11 - 11 молодших біт адреси переходу, що витягають із тіла команди абсолютного переходу.

Відносний перехід (короткий) дозволяє передати управління в межах від - 128 до + 127 адресного простору.

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

SJMP rel; (PC) ← (PC) + 2, (PC) ← (PC) + rel (перехід на rel),
де rel - коротка відносна адреса переходу в межах - 128 - +127 байт.

Всі команди умовного переходу використовують відносний перехід. **Відносна адреса переходу (rel)** міститься в другому байті команди.

NB! При програмуванні в середовищі Franklin Software досить просто вказати команду **JMP** і мітку переходу, при трансляції програми буде підставлена команда оптимальна по розрядності адреси переходу.

Непрямий перехід.
JMP @A + DPTR; (PC) ← (A) + (DPTR)
Передає управління за непрямою адресою. Ця команда зручна тим, що надає можливість організації переходу обчислюється за адресою програмою і невідомому при написанні вихідного тексту програми.

3.2 Команди умовного переходу

Розвинена система умовних переходів надає можливість здійснювати розгалуження за наступними умовами.
Перехід на мітку (rel), якщо виконується одна з умов:

1. Якщо акумулятор містить нуль;

JZ rel; якщо (A) = 0, то PC = PC + rel

2. Якщо вміст акумулятора не дорівнює нулю;

JNZ rel; якщо (A) ≠ 0, то PC = PC + rel

3. Якщо біт C (позики / перенесення) дорівнює одиниці;

JC rel; якщо (C) = 1, то PC = PC + rel

4. Якщо біт C (позики / перенесення) дорівнює нулю;

JNC rel; якщо (C) = 0, то PC = PC + rel

5. Якщо адресований біт дорівнює одиниці;

JB bit, rel; якщо (bit) = 1, то PC = PC + rel

6. Якщо адресований біт не дорівнює одиниці, тобто дорівнює нулю;

JNB bit, rel; якщо (bit) = 0, то PC = PC + rel

7. Якщо адресований біт дорівнює одиниці, скидання адресованого біта в 0;

JBC bit, rel; якщо (bit) = 1, то PC = PC + rel і (bit) = 0

3.3 Організація циклів

Для організації програмних циклів зручно користуватися такими командами.

1. Декремент регістра і перехід, якщо він не дорівнює нулю

DJNZ Rn, rel; Rn-1→ Rn, якщо (Rn) ≠ 0, то перехід на rel

2. Декремент комірки ВПД і перехід, якщо її вміст не = нулю

DJNZ dir, rel; (ad) -1→ (ad), якщо (ad) ≠ 0, то перехід на rel

3. Порівняння акумулятора з коміркою ВПД і перехід, якщо вони не = один одному

CJNE A, dir, rel; якщо (A) ≠ (ad), то перехід на rel

4. Порівняння акумулятора з константою і перехід, якщо вони не дорівнюють один одному

CJNE A, #d, rel; якщо (A) ≠ #d, то перехід на rel

5. Порівняння регістра з константою і перехід, якщо вони не дорівнюють один одному

CJNE Rn, #d, rel; якщо (Rn) ≠ #d, то перехід на rel

6. Порівняння побічно адресованої комірки ВПД з константою і перехід, якщо вони не дорівнюють один одному

CJNE @Ri, #d, rel; якщо ((Ri)) ≠ #d, то перехід на rel.

Наприклад, команда CJNE ефективно використовується в процедурах очікування якої-небудь події.

WAIT: CJNE A, P0, WAIT

Команда буде виконуватися до тих пір, поки на лініях порту 0 встановиться інформація, що збігається з вмістом акумулятора.

Приклад 3.1.

Написати програму для вирішення наступного математичного завдання.

Знайти кількість чисел, що знаходяться в межах: 05 <X <10H з масиву цілих чисел в комірках пам'яті (20H) ... (29H). Результат помістити в комірку (2AH).

```
        ORG 0
        MOV R0 , #20H     ; 20H → (R0) – базова адреса масива;
        MOV R1 , #0AH     ; 0AH → (R1)– лічильник кількості циклів;
        MOV R2 , #0       ; 0H →  (R2) – лічильник кількості чисел;
L0:     MOV A , @R0       ; ((R0)) → (A)
        CLR C             ; (C)=0
        SUBB A , #05H     ; (A) – 05H – (C)→ (A)
        JZ L1             ; якщо (A)=0 (X=05), то перейти на мітку L1
        JC L1             ; якщо (C)=1 (X<05), то перейти на мітку L1
        MOV A , @R0       ; ((R0)) → A
        CLR C             ; (C)=0
        SUBB A ,#10H      ; (A) – 10H – (C) → (A)
        JZ L1             ; якщо (A)=0 (X=10H), то перехід на L1
        JNC L1            ; якщо (C)=0 (X>10H) , то перехід на L1
        INC R2            ; (R2) + 1 → (R2)
L1:     INC R0            ; (R0) + 1 → (R0)
        DJNZ R1, L0       ; (R1)-1 → (R1); якщо (R1) ≠0 , то перехід
на  мітку L0
        MOV 2AH, R2       ; (R2) → (2AH)
L3:     JMP L3            ; перехід на L3
        END
```

Приклад 3.2.

Написати програму для формування послідовності коротких та довгих імпульсів на вказаному виході порту 0 при низькому рівні сигналу вказаного біта порту 2 згідно індівідуального завдання («*» - короткий імпульс; «-» - довгий імпульс).

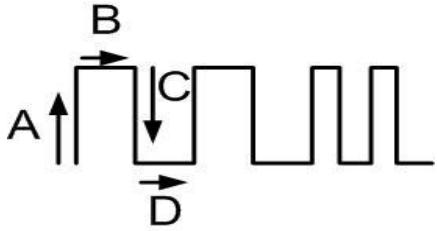


Рисунок 3.1 – Умовне зображення послідовності імпульсів

Приклад фрагменту програми.

```
Org 0h
Setb P2.0
CLR P0.0
M1:     JB P2.0, M1     ;очікування низького рівня
        Mov R2, #02h    ;задання кількості імпульсів
-----------------------------------------
M2:     Setb P0.0       ;A
-----------------------------------------
        Mov R1, #10h
M3:     Djnz R1, M3     ;B
-----------------------------------------
        CLR P0.0        ;C
-----------------------------------------
        Mov R1, #10h
M4:     Djnz R1, M4     ;D
-----------------------------------------
  Djnz R2, M2

І так далі…

M100:   JNB P2.0, M100      ;очікування високого рівня
Jmp M1
End
```

4 КОМАНДИ НЕПРЯМОЇ АДРЕСАЦІЇ

Непряма реєстрова адресація використовується для звертання до комірок внутрішньої пам'яті даних, причому в якості адреси комірки, в якій знаходиться операнд, використовується вміст одного з індексних регістрів R0, R1 [1,3].

Ознака непрямої реєстрової адресації в команді - @.

Дії з коміркою, адреса якої знаходиться в індексному регістрі.

4.1 Команди премещенія даних з непрямою реєстровою адресацією

1. Пересилання в РПД константи
MOV @Ri , #data; ((Ri))←#d
2. Пересилка байта з РПД у комірку ВПД
MOV direct , @Ri ; (dir)←((Ri))
3. Пересилання в РПД вмісту комірки ВПД
MOV @Ri , direct; ((Ri))←(dir)
4. Пересилання в акумулятор байта з РПД (i=0,1)
MOV A, @Ri ; (A)←((Ri))
5. Пересилання в РПД вмісту акумулятора
MOV @Ri , A; ((Ri))←(A)
6. Обмін акумулятора з непрямо адресованою коміркою ВПД
XCH A , @Ri; (A)↔((Ri))

4.2 Арифметичні команди з непрямою реєстровою адресацією

1. Інкремент непрямоадресованої комірки ВПД
INC @Ri; (Ri) ← (Ri)+1
2. Декремент непрямоадресованої комірки ВПД
DEC @Ri; (Ri) ← (Ri)-1
3. Додавання акумулятора і непрямоадресованої комірки ВПД
ADD A , @Ri; (A) ← (A)+((Ri))
4. Додавання акумулятора і непрямоадресованої комірки ВПД з урахуванням переносу
ADDC A , @Ri; (A) ← (A)+((Ri))+(C)

5. Віднімання від акумулятора непрямоадресованої комірки ВПД і позики

SUBB A , @Ri; (A) ← (A)-(C)-((Ri))

4.3 Логічні команди з непрямою реєстрової адресацією

1. Логічне АБО акумулятора і непрямоадресованої комірки ВПД

ORL A , @Ri; (A)←(A) ((Ri))

2. Логічне І акумулятора і непрямоадресованої комірки ВПД

ANL A , @Ri; (A)←(A) ((Ri))

3. Що виключає АБО акумулятора і непрямоадресованої комірки ВПД

XRL A , @Ri; (A)←(A) ((Ri))

4.4 Команди розгалуження з непрямою реєстровою адресацією

Порівняння непрямоадресованої комірки ВПД з константою і перехід, якщо вони не дорівнюють одне одному

CJNE @Ri , #data, rel ; перехід якщо ((Ri)) ≠#data

(PC)←(PC)+3, якщо ((Ri))≠#d, то (PC)← (PC)+rel,

якщо ((Ri))<#d, то (C)←1, інакше (C)←0

Приклад 4.1.

Знайти суму елементів масиву, розташованих в комірках з адресами 20h ... 27h.

$$\sum\left[(20h)...(27h)\right]\rightarrow(30h)$$

<u>Без непрямої реєстрової адресації</u>

```
MOV A, 20H
ADD A, 21H
ADD A, 22H
...
...
ADD A, 27H
MOV 30H, A
```

Використовуючи непряму реєстрову адресацію.

```
MOV R0, # 20H; адреса початку масиву
MOV R1, # 08H; кількість елементів масиву
CLR  A
L1:
  ADD  A, @R0
  INC  R0
  DJNZ  R1, L1

  MOV  30H, A
```

Приклад 4.2.

Знайти максимум з елементів масиву, розташованих в комірках з адресами 20h ... 27h. Результат помістити в комірку 40h.

```
MOV R0, # 20H
MOV R1, # 8H
MOV A, 20H
L1: MOV 30H, @ R0
CJNE A, 30H, L2; якщо (A) ≠ (30H), то перехід на L2,
                 при цьому якщо (A) <(30H), то (C) = 1,
                 в іншому випадку (C) = 0.

L2: JNC L3; якщо (C) = 0, то перехід на L3
MOV A, 30H; перенесення в (A) більшого числа

L3: INC R0
DJNZ R1, L1
MOV 40H, A
M1: JMP M1
```

5 ПІДПРОГРАМИ

Для звернення до підпрограми використовуються команди виклику [1,3]

LCALL

ACALL

Ці команди зберігають в стеку адресацію повернення в наступну команду або в основну програму. Для повернення з підпрограми необхідно виконати команду RET, яка бере з стека адресу повернення. RETI відрізняється від команди RET тим, що дозволяє переривання обслугованого рівня переривання.

Виклик підпрограми:

LCALL ad16	;виклик підпрограми з адресою (ad16)
	(PC) = PC + 3 SP = SP + 1
	(SP) ← PC_{0-7} SP = SP + 1
	(SP) ← PC_{15-8} PC = ad16
ACALL ad11	;виклик підпрограми з адресою (ad16)
	PC = PC =3 SP = SP + 1
	(SP) ← PC_{0-7} SP = SP + 1
	(SP) ← PC_{15-8} PC_{0-10} ← ad11
RET	;PC_{15-8} ← (SP) SP – 1 → SP
	PC_{0-7} ← (SP) SP – 1 → SP
RETI	;PC_{15-8} ← (SP) SP – 1 → SP
	PC_{0-7} ← (SP) SP – 1 → SP
	Дозволяє переривання по даному рівню

NOP - команда порожньої операції.

NB! При програмуванні в середовищі Franklin Software досить просто вказати команду **CALL** і назву підпрограми, при трансляції програми буде підставлена команда оптимальна по розрядності адреси переходу.

ПЕРЕЛІК ПОСИЛАНЬ

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

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

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

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

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

6. Колонтаєвський, Ю.П. Конспект лекцій з дисципліни «Мікропроцесорна техніка» (для студентів, які навчаються за напрямом 6.050701 – Електротехніка та електротехнології всіх форм навчання) / Ю. П. Колонтаєвський – Харків : ХНУМГ ім. О. М. Бекетова, 2016. – 78 с.

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

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

9. Analog Devices – Режим доступу: https://www.analog.com/en/index.html.

10. Nazarova, O. Computer Modeling of Multi-Mass Electromechanical Systems. The Third International Workshop on Computer Modeling and Intelligent Systems (CMIS-2020), Vol. 2608, pp. 489-498.

11. Osadchyy, V. Laboratory Stand for Investigation of Liquid Level Microprocessor Control Systems / V. Osadchyy, O. Nazarova // 2020 IEEE Problems of Automated Electrodrive. Theory and Practice (PAEP),

Kremenchuk, Ukraine, 2020. - pp. 1-4. doi: 10.1109/PAEP49887.2020.9240868.

12. Nazarova, O. Research on the Influence of the Position of the Electric Vehicles Mass Center on Their Characteristics / O. Nazarova, V. Osadchyy, V. Brylystyi // 2020 IEEE Problems of Automated Electrodrive. Theory and Practice (PAEP), Kremenchuk, Ukraine, 2020. - pp. 1-4, doi: 10.1109/PAEP49887.2020.9240824.

13. Nazarova, O. Influence of Supply Voltage on the Accuracy of Two-Speed Elevator Positioning / O. Nazarova, V. Osadchyy, S. Shulzhenko // 2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES), 2021. - pp. 1-4, doi: 10.1109/MEES52427.2021.9598664.

14. Osadchyy, V. Laboratory Stand for Research of Energy Characteristics of Electric Vehicle Drives / V. Osadchyy, O. Nazarova, V. Brylystyi // 2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES), 2021. - pp. 1-4, doi: 10.1109/MEES52427.2021.9598661.

15. Osadchyy, V. The Research of a Two-Mass System with a PID Controller, Considering the Control Object Identification / V. Osadchyy, O. Nazarova, M. Olieinikov // 2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES), 2021. - pp. 1-5, doi: 10.1109/MEES52427.2021.9598542.

16. Брилистий, В.В. Вимірювання крутного моменту для дослідження енергетичних характеристик приводів електромобілей / В.В. Брилистий, О.С. Назарова, В.В. Осадчий // Електротехніка та електроенергетика, 2021. - №4. - С. 36–44. https://doi.org/10.15588/1607-6761-2021-4-4

17.Кулинич, Е.М. Лабораторний стенд з бездротовим інтерфейсом для дослідження електроприводу постійного струму / Е.М. Кулинич, О.С. Назарова, Д.В. Гончаров, С.Г. Чернишев, В.В. Піскун // Електроенергетика та електротехніка, 2020. - №3. – С.24-36. DOI: https://doi.org/10.15588/1607-6761-2020-3-3.

Додаток А

Перелік команд мікроконтролера Intel 8051

Таблиця А.1 - Група команд пересилання даних

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

Продовження таблиці А.1

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

Таблиця А.2 - Команди арифметичних операцій

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

Продовження таблиці А.2

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

Продовження таблиці А.2

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

Таблиця А.3 - Команди логічних операцій

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

Продовження таблиці А.3

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

Продовження таблиці А.3

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

Таблиця А.4 - Команди роботи з бітами

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

Продовження таблиці А.4

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

Таблиця А.5 - Команди передачі керування

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

Продовження таблиці А.5

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

Продовження таблиці А.5

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

Продовження таблиці А.5

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

Додаток Б

Представлення даних у двійковій, шістнадцятковій та десятковій системі числення

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

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