

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 «Електротехнічні системи
електроспоживання»
усіх форм навчання

Частина 2

2021

Конспект лекцій з дисципліни «Основи мікропроцесорної техніки» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітня програма «Електротехнічні системи електроспоживання» усіх форм навчання. Частина 2. /Укл: О.С. Назарова, В.В. Осадчий – Запоріжжя: НУ «Запорізька політехніка», 2021. – 50 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 № 07 від 02.12.2020 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 05 від 18.02.2021 р.

ЗМІСТ

Передмова	4
1 Команди логічних операцій.....	5
2 Команди операцій з бітами	8
3 Розгалуджені структури. Цикли.....	16
3.1 Команди безумовного переходу	16
3.2 Команди умовного переходу	17
3.3 Організація циклів	18
4 Команди непрямої адресації	21
4.1 Команди переміщення даних з непрямою реєстровою адресацією	21
4.2 Арифметичні команди з непрямою реєстровою адресацією	21
4.3 Логічні команди з непрямою реєстровою адресацією	22
4.4 Команди розгалуження з непрямою реєстровою адресацією	22
5 Підпрограми	24
6 Паралельне введення-виведення інформації.....	25
7 Програмування мікроконтролера ADuC 841	27
8 Ознайомлення зі стендом мікропроцесорних засобів автоматизації фірми SIEMENS та вивчення його функціональних можливостей	31
Перелік посилань.....	37
Додаток А Перелік команд мікроконтролера Intel 8051..	39
Додаток Б Представлення даних у двійковій, шістнадцятковій та десятковій системі числення	50

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 «Електротехнічні системи електроспоживання».

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»; ознайомитися з можливостями мікропроцесорних засобів автоматизації фірми SIEMENS.

Для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електротехнічні системи електроспоживання» 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.

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

Org 0h

Mov a, 21h ;(21H)→(A)

Clr c ; (C)=0

Subb a, #10h ;(A)-10H-(C)→(A)

Cpl a ; $(\bar{A}) \rightarrow (A)$

Mov b, 20h ;(20H)→(B)

Mul ab ;(A)*(B) →(B).(A)

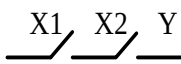
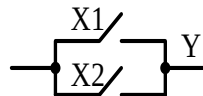
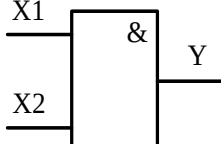
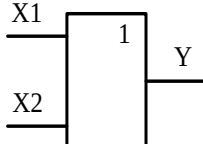
Xrl a,#45H ;(A) $\forall$ 45H → (A)

mov 23h, A ;(A) →(23H)

L1: JMP L1

End

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

Назва	Логічне множення, кон'юнкція, «логічне І»	Логічне додавання, диз'юнкція, «логічне АБО»																																																		
Позначення	AND	OR																																																		
Графічне зображення	∧	∨																																																		
Таблиця істинності (правило)	<table><tr><td>X1</td><td></td><td>X2</td><td></td><td>Y</td></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>	X1		X2		Y	0	∧	0	=	0	0	∧	1	=	0	1	∧	0	=	0	1	∧	1	=	1	<table><tr><td>X1</td><td></td><td>X2</td><td></td><td>Y</td></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>	X1		X2		Y	0	∨	0	=	0	0	∨	1	=	1	1	∨	0	=	1	1	∨	1	=	1
X1		X2		Y																																																
0	∧	0	=	0																																																
0	∧	1	=	0																																																
1	∧	0	=	0																																																
1	∧	1	=	1																																																
X1		X2		Y																																																
0	∨	0	=	0																																																
0	∨	1	=	1																																																
1	∨	0	=	1																																																
1	∨	1	=	1																																																
Команда (мнемоніка)	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)	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)																																																		
Електрична схема																																																				
Комбінаційна схема																																																				

Продовження таблиці 1.1

Назва	Логічне виключення, додавання по модулю 2, «виключне АБО»	Логічне заперечення, інверсія, «логічне НЕ»																																		
Позна- чення	XOR	NOT																																		
Графі- чне зобра- ження	∇	$\overline{X}$																																		
Таблиця істинно- сті (прави- ло)	<table><tr><td>X1</td><td></td><td>X2</td><td></td><td>Y</td></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>0</td></tr></table>	X1		X2		Y	0	∇	0	=	0	0	∇	1	=	1	1	∇	0	=	1	1	∇	1	=	0	<table><tr><td>X</td><td></td><td>Y</td></tr><tr><td>$\overline{0}$</td><td>=</td><td>1</td></tr><tr><td>$\overline{1}$</td><td>=</td><td>0</td></tr></table>	X		Y	$\overline{0}$	=	1	$\overline{1}$	=	0
X1		X2		Y																																
0	∇	0	=	0																																
0	∇	1	=	1																																
1	∇	0	=	1																																
1	∇	1	=	0																																
X		Y																																		
$\overline{0}$	=	1																																		
$\overline{1}$	=	0																																		
Коман- да (мнемо- ніка)	XRL A, Rn; (A) ∇ (Rn) $\rightarrow$ (A) XRL A, #d8; (A) ∇ d8 $\rightarrow$ (A) XRL A, dir; (A) ∇ (dir) $\rightarrow$ (A) XRL dir, A; (dir) ∇ (A) $\rightarrow$ (dir) XRL dir, #d8; (dir) ∇ d8 $\rightarrow$ (dir) XRL A, @Ri; (A) ∇ ((Ri)) $\rightarrow$ (A)	CPL A; ($\overline{A}$) $\rightarrow$ (A) CPL C; ($\overline{C}$) $\rightarrow$ (C) CPL bit; $\overline{(\text{bit})} \rightarrow (\text{bit})$																																		
Елек- трична схема																																				
Комбі- націй- на схема																																				

2 КОМАНДИ ОПЕРАЦІЙ З БІТАМИ

Відмінною особливістю цієї групи команд є те, що вони оперують з одиницями 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) $\leftarrow$ 0

CLR bit; (bit) $\leftarrow$ 0

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

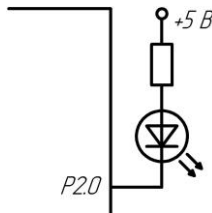


Рисунок 2.1 – Апаратна частина
Струм тече – світлодіод горить.

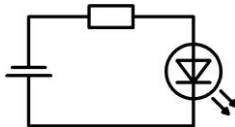


Рисунок 2.2 - Еквівалентна схема

Програмна частина.
CLR 82H $\leftrightarrow$ CLR P0.2

Установка в «одиницю»

SETB C; (C) $\leftarrow$ 1
SETB bit; (bit) $\leftarrow$ 1

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

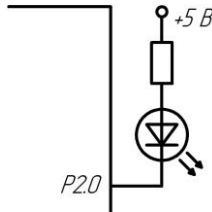


Рисунок 2.3 – Апаратна частина

Струм не тече – світлодіод не горить.

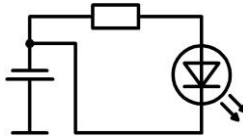


Рисунок 2.4 – Еквівалентна схема

Програмна частина.
SETB 0A0H $\leftrightarrow$ SETB P2.0

Переміщення (копіювання)

MOV C, bit; (bit) $\rightarrow$ (C)
MOV bit, C; (C) $\rightarrow$ (bit)

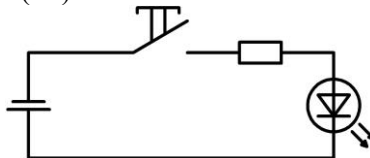


Рисунок 2.5 – Еквівалентна схема

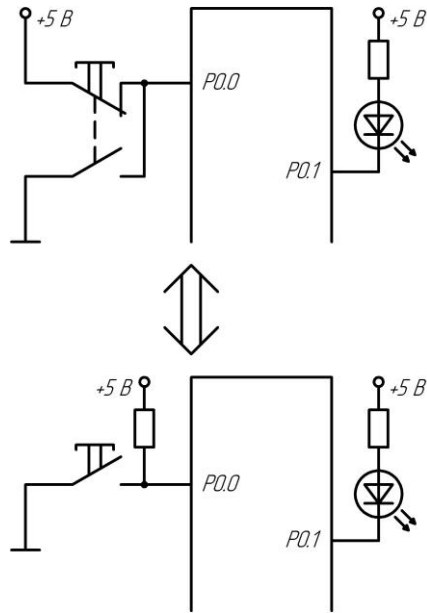


Рисунок 2.6 – Апаратна частина

Програмна частина.

SETB P0.0; налаштування біта до вводу

L1: MOV C, P0.0

MOV P0.1, C

JMP L1

Інвертування

CPL C; $(C) \rightarrow (\bar{C})$

CPL bit; $(\text{bit}) \rightarrow (\bar{\text{bit}})$

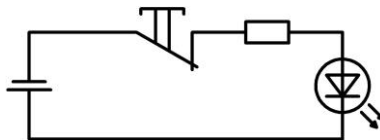


Рисунок 2.7 – Еквівалентна схема

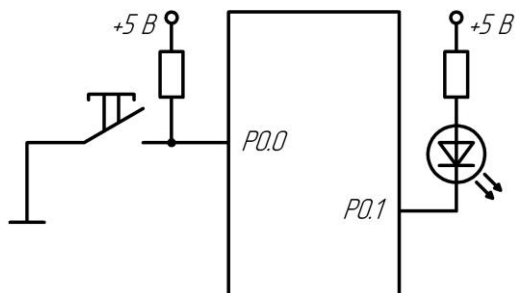


Рисунок 2.8 – Апаратна частина

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; $(C) \wedge (\text{bit}) \rightarrow (C)$

ANL C, / bit; $(C) \wedge (\text{bit}) \rightarrow (C)$

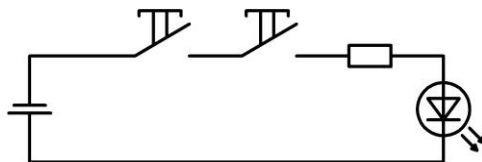


Рисунок 2.9 – Еквівалентна схема

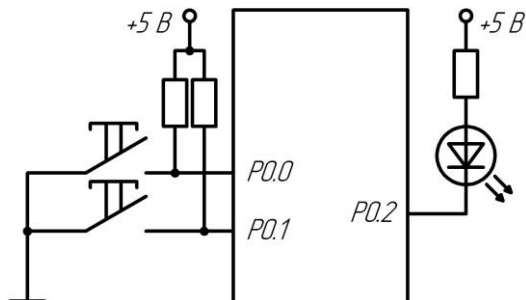


Рисунок 2.10 – Апаратна частина

Програмна частина.

SETB PO.0

SETB PO.1

L1:MOV C, PO.0

CPL C

ANL C, IPO.1

CPL C

MOV PO.2, C

JMP L1

Логічне додавання (логічне «АБО»)

ORL C, bit; (C) V (bit) $\rightarrow$ (C)

ORL C, /bit; (C) V (bit) $\rightarrow$ (C)

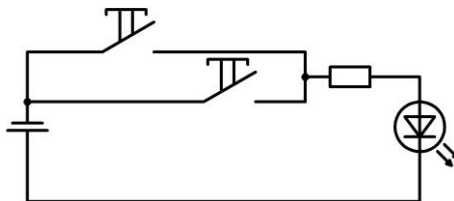


Рисунок 2.11 – Еквівалентна схема

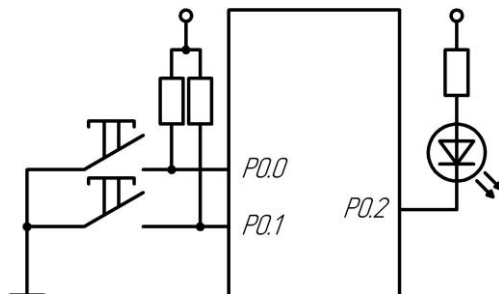


Рисунок 2.12 – Апаратна частина

Програмна частина.

SETB PO.0

SETB PO.1

L1:MOV C, PO.0

CPL C

ORL C, I PO.1
 CPL C
 MOV PO2, 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 функцію алгебри логіки:

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

Складаємо за цією функцією комбінаційну схему.

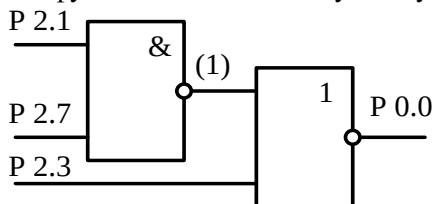


Рисунок 2.13 – Комбінаційна схема, що відповідає заданій функції алгебри логіки.

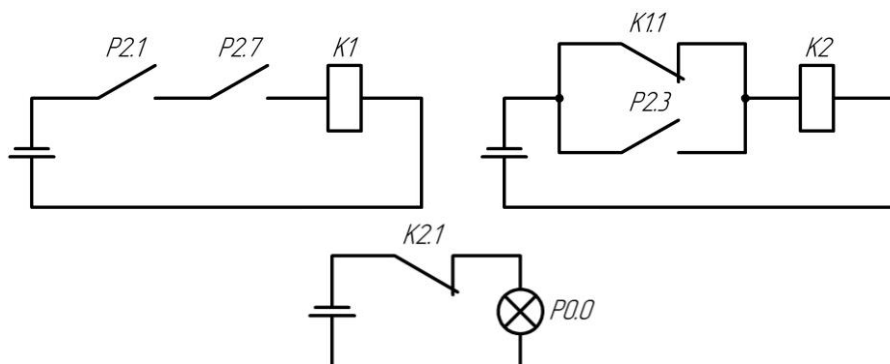


Рисунок 2.14 – Електрична схема, що відповідає заданій функції алгебри логіки.

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.

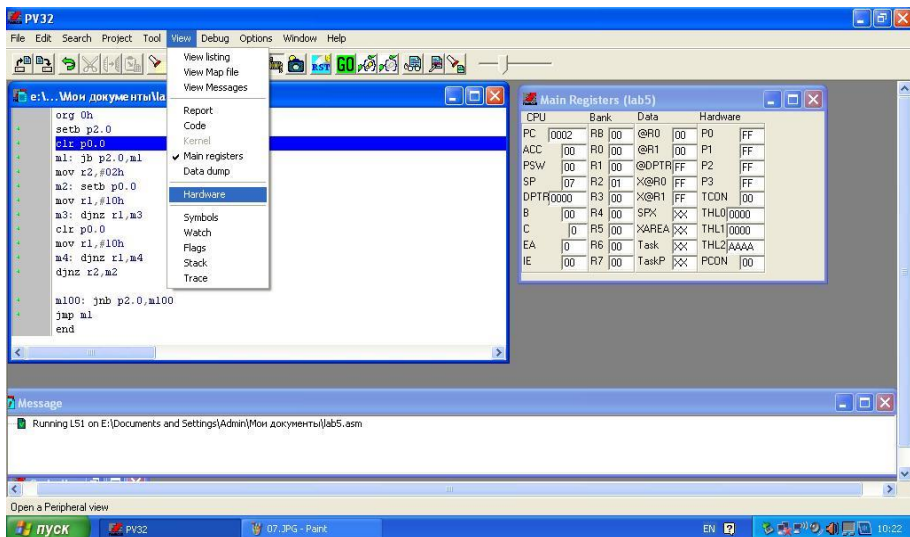


Рисунок 2.15 – Скрін екрану при відкритті додаткового обладнання у вигляді портів P0, P2.

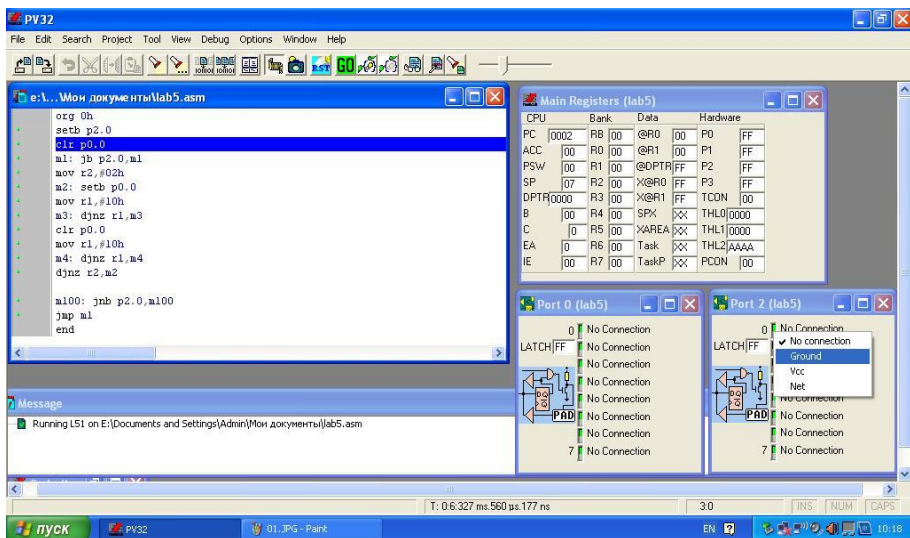


Рисунок 2.16 – Скрін екрану при перевірці працездатності програми

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) $\leftarrow$ (PC) + 2, (PC0-10) $\leftarrow$ 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) $\leftarrow$ (PC) + 2, (PC) $\leftarrow$ (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) $\leftarrow$ (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 $(A) \neq 0$, 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; $Rn-1 \rightarrow Rn$, якщо $(Rn) \neq 0$, то перехід на rel
2. Декремент комірки ВПД і перехід, якщо її вміст не = нулю
DJNZ dir, rel; $(ad) - 1 \rightarrow (ad)$, якщо $(ad) \neq 0$, то перехід на rel
3. Порівняння акумулятора з коміркою ВПД і перехід, якщо вони не = один одному
CJNE A, dir, rel; якщо $(A) \neq (ad)$, то перехід на rel
4. Порівняння акумулятора з константою і перехід, якщо вони не дорівнюють один одному
CJNE A, #d, rel; якщо $(A) \neq \#d$, то перехід на rel
5. Порівняння регістра з константою і перехід, якщо вони не дорівнюють один одному
CJNE Rn, #d, rel; якщо $(Rn) \neq \#d$, 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 $((Ri)) \neq \#d$, 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 індивідуального завдання («*» - короткий імпульс; «-» - довгий імпульс).

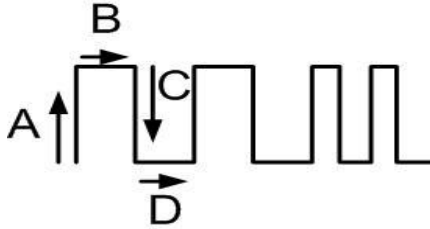


Рисунок 3.1 – Умовне зображення послідовності імпульсів

Приклад фрагменту програми.

Org 0h

Setb P2.0

CLR P0.0

M1: JB P2.0, M1 ;очікування низького рівня
Mov R2, #02h ;завдання 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

I 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 переміщення 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) $\leftarrow$ (A)-(C)-((Ri))

4.3 Логічні команди з непрямою реєстрової адресацією

1. Логічне АБО акумулятора і непрямоадресованої комірки ВПД
 ORL A , @Ri; (A) $\leftarrow$ (A) ((Ri))

2. Логічне І акумулятора і непрямоадресованої комірки ВПД
 ANL A , @Ri; (A) $\leftarrow$ (A) ((Ri))

3. Що виключає АБО акумулятора і непрямоадресованої комірки ВПД

XRL A , @Ri; (A) $\leftarrow$ (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)) $\neq$ #data
 (PC) $\leftarrow$ (PC)+3, якщо ((Ri)) $\neq$ #d, то (PC) $\leftarrow$ (PC)+rel,
 якщо ((Ri))<#d, то (C) $\leftarrow$ 1, інакше (C) $\leftarrow$ 0

Приклад 4.1.

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

$$\sum[(20h) \dots (27h)] \rightarrow (30h)$$

Без непрямої реєстрової адресації

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 комітках 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 $(A) \neq (30H)$, 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)

$(PC) = PC + 3$ $SP = SP + 1$

$(SP) \leftarrow PC_{0-7}$ $SP = SP + 1$

$(SP) \leftarrow PC_{15-8}$ $PC = ad16$

ACALL ad11 ;виклик підпрограми з адресою (ad16)

$PC = PC + 3$ $SP = SP + 1$

$(SP) \leftarrow PC_{0-7}$ $SP = SP + 1$

$(SP) \leftarrow PC_{15-8}$ $PC_{0-10} \leftarrow ad11$

RET ; $PC_{15-8} \leftarrow (SP)$ $SP - 1 \rightarrow SP$

$PC_{0-7} \leftarrow (SP)$ $SP - 1 \rightarrow SP$

RETI ; $PC_{15-8} \leftarrow (SP)$ $SP - 1 \rightarrow SP$

$PC_{0-7} \leftarrow (SP)$ $SP - 1 \rightarrow SP$

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.

6 ПАРАЛЕЛЬНЕ ВВЕДЕННЯ-ВИВЕДЕННЯ ІНФОРМАЦІЇ

Семисегментний індикатор (ССИ) - це один з простіших індикаторів, який використовується для виведення інформації з мікроконтролера (МК). Він підключається до паралельного порту таким чином, що кожному сегменту відповідає певний біт порту (рисунок 6.1).

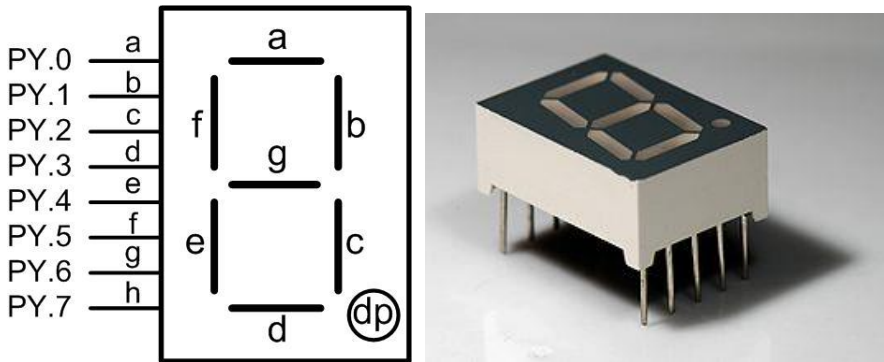


Рисунок 6.1 - Семисегментний індикатор

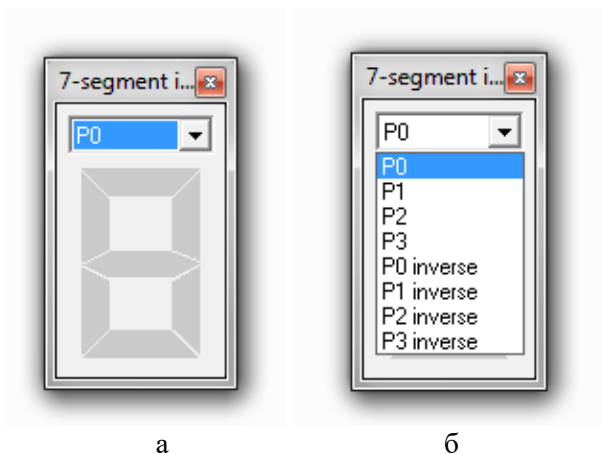
Наприклад, для того, щоб на семисегментному індикаторі з'явилась цифра «5», треба в один із чотирьох портів МК вивести двійкове число «01101101b» (таблиця 6.1).

Таблиця 6.1 – Відповідність сегментів індикатора для виведення цифри «5»

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

Для сполучення Proview Franklin Software з емулятором семисегментного індикатора треба запустити файл 7segment_1.exe.

Після цього відкриється вікно, зовнішній вигляд якого показано на рисунку 6.2, а. Зі списку, що відкривається вибирається потрібний порт (рисунок 6.2, б).



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

Рисунок 6.2 – Вікно емулятора CCI

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

7 ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРА ADuC 841

Використання контролера ADuC 841 обумовлює його доступність, набір виконуваних дій, можливість узгодження з лабораторним і виробничим обладнанням [11,19], порівняно невисока вартість [14,18].

Порядок запису розробленої програми в пульт.

1. Підключити пульт до USB.
2. Запустити програму «WDS Analog Devices» (рисунок 7.1).

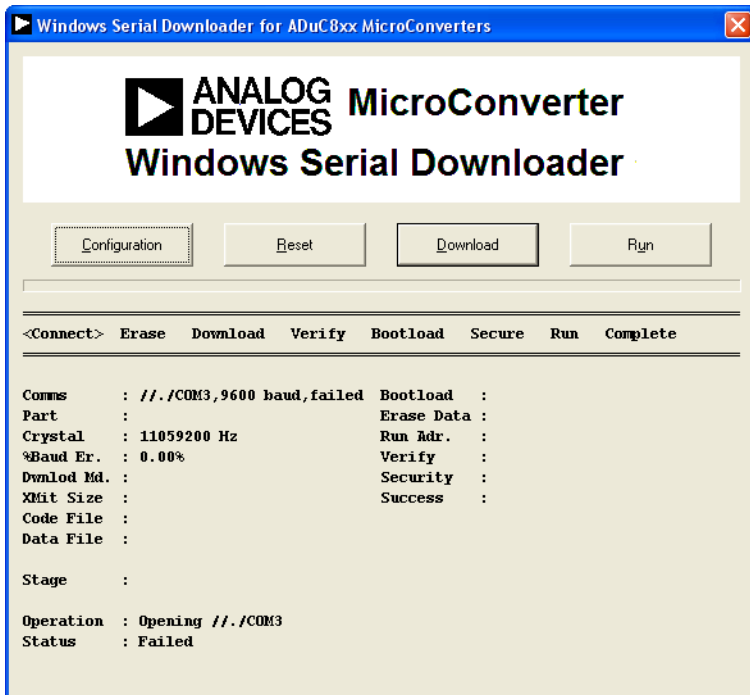


Рисунок 7.1 – Вікно програми «WDS Analog Devices».

3. Натиснути кнопку «Configuration» цього вікна, яка відкриває нове вікно «Configuration» (рисунок 7.2)

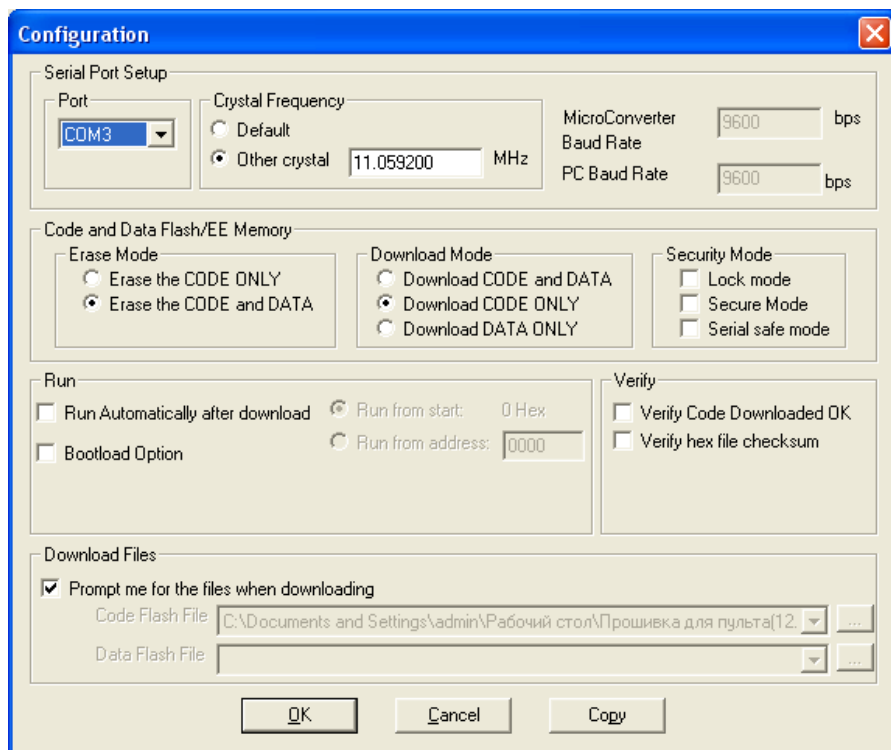


Рисунок 7.2 – Вікно «Configuration».

4. В області «Crystal Frequency» вибрати опцію «Other crustal» з частотою 11,0592 MHz, далі натиснути кнопку «OK».

5. Для встановлення у режим програмування на задній стороні пульта (рисунок 7.3) треба переключити повзунковий перемикач у положення «Р» і натиснути кнопку скидання «R», після чого пульт перейде у режим програмування.

6. Далі у вікні програми WDS треба натиснути кнопку скидання «Reset», після чого «Status» повинен змінитися на «Reset OK! ADuC841 device detected» (рисунок 7.4)



Рисунок 7.3 – Задня сторона пульта.

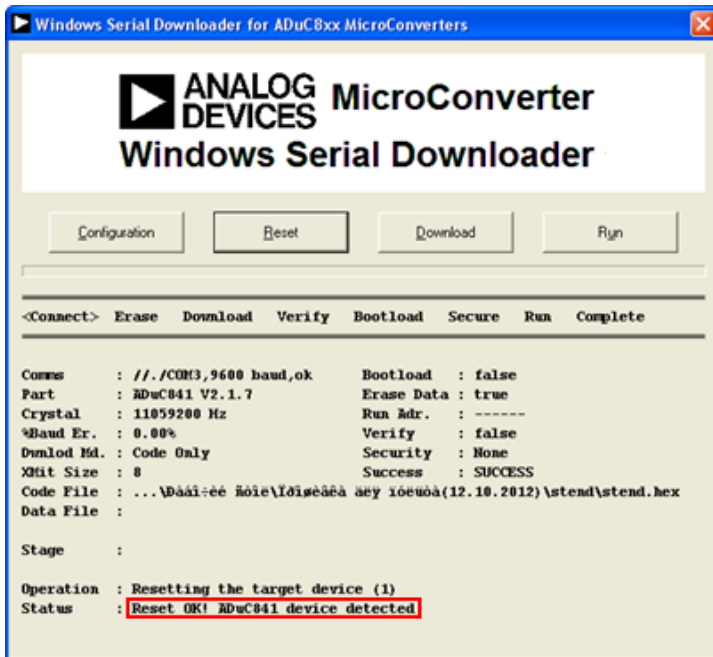


Рисунок 7.4 - Вікно програми WDS після виконання скидання.

7. Якщо з'явиться повідомлення «Status – Failed to connect to the ADuC8xx after 2 attempts», необхідно повторити пункт 6.

8. Далі у вікні програми WDS натиснути кнопку «Download» після чого відкриється вікно для вибору прошивки, вибрати потрібний файл і натиснути кнопку «Открыть», після цього почнеться завантаження програми у пульт.

9. Після завершення завантаження у «Status» відобразиться параметр «Success».

10. Далі для запуску програми в пульті необхідно натиснути на кнопку «Run» у вікні програми WDS або перемикнути повзунковий перемикач на задній панелі у положення «W» і натиснути кнопку скидання «R».

При розробці та удосконаленні систем автоматичного керування широко використовують найрізноманітніші мікроЕОМ [4], мікропроцесори, мікроконтролери для підвищення енергоефективності виробничих процесів та удосконалення алгоритмів керування [12,13,21].

8 ОЗНАЙОМЛЕННЯ ЗІ СТЕНДОМ МІКРОПРОЦЕСОРНИХ ЗАСОБІВ АВТОМАТИЗАЦІЇ ФІРМИ SIEMENS ТА ВИВЧЕННЯ ЙОГО ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ

При ознайомленні зі стендом автоматизації здобуваються навички програмування LOGO! для автоматичної роботи конвеєра лінії приготування фарби [15,16].

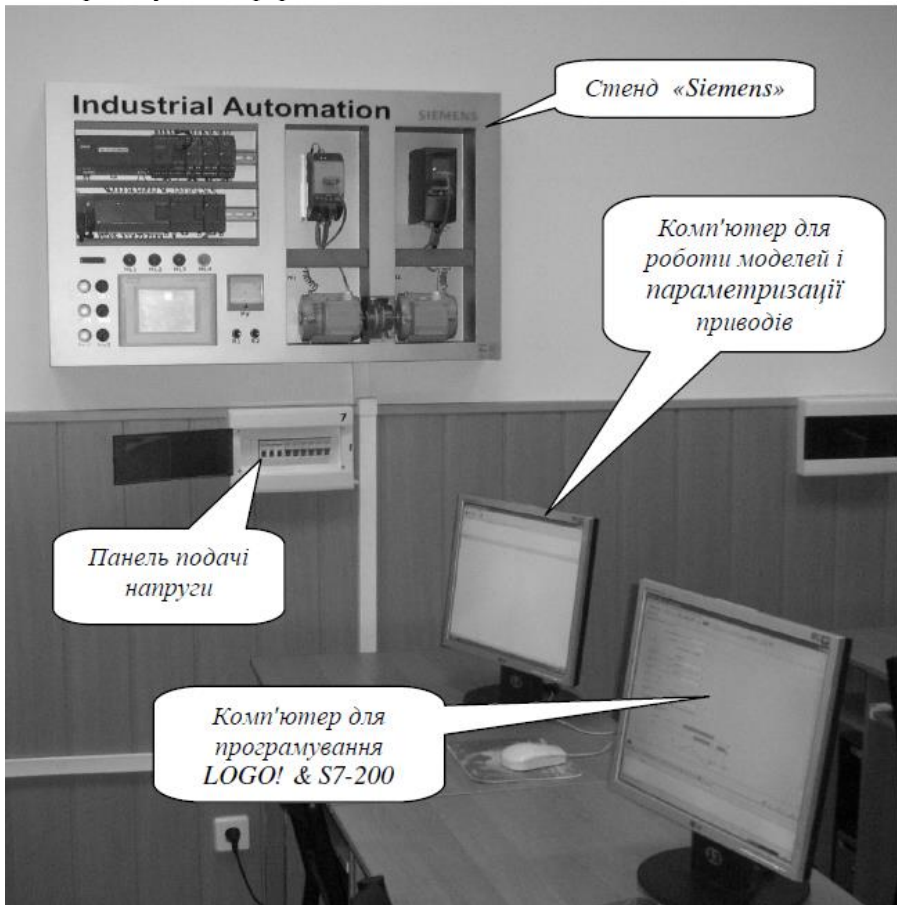


Рисунок 8.1 – Загальний вигляд робочого місця

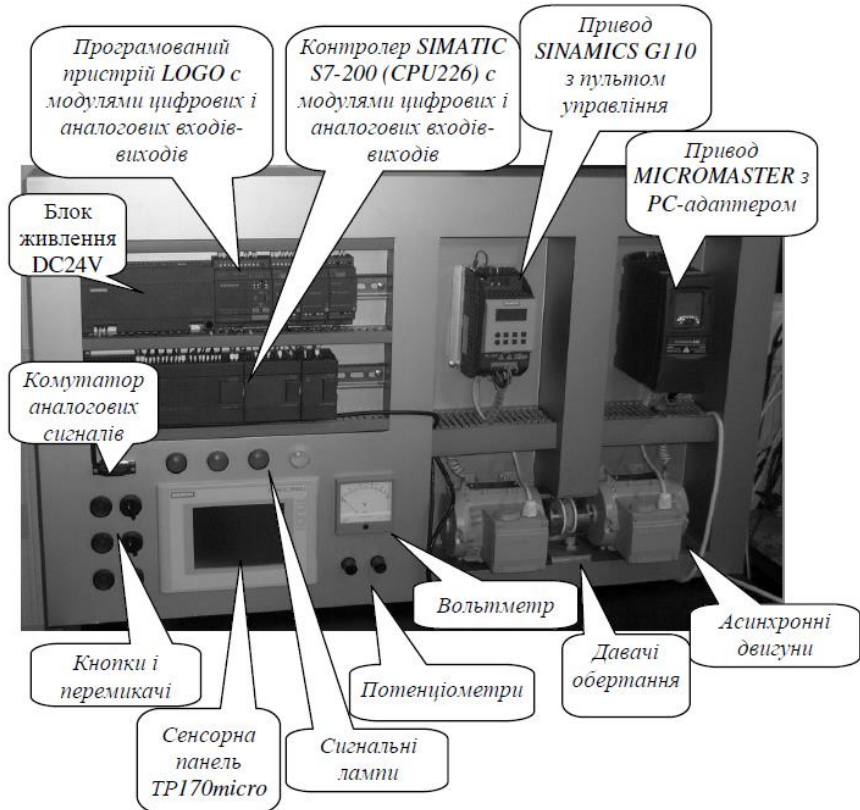


Рисунок 8.2 – Опис стенда фірми SIEMENS

Можливість довільної конфігурації з'єднань обладнання стенду між собою і з моделлю об'єкта керування. До стенду приєднані два комп'ютери. Перший комп'ютер містить програмні засоби розробки програм для LOGO! - LogoSoft Comfort і контролера S7-200 (CPU226) - Step7-micro/Win [20]. Цей комп'ютер сполучений з LOGO! кабелем програмування LOGO! через COM-порт, а з контролером S7- 200 (CPU226) з допомогою PC / PPI адаптера через USB порт. Другий комп'ютер містить програму STARTER для параметризації і онлайн роботи приводів SINAMICS G110 і Micromaster 440 [15,16].

Зв'язок з відповідними приводами проводиться кабелями RS232 через COM і USB порти. Також на цьому комп'ютері встановлено пакет математичного моделювання MATLAB. Він служить для написання, коригування та запуску математичної моделі об'єктів керування. Модель керування використовує в якості вхідних координат реальні кути повороту і частоту обертання валу двигуна, а також стану цифрових виходів пристроїв стенду. У свою чергу модель станом змодельованого об'єкта може змінювати стан цифрових і аналогових входів об'єкта. На рисунку 8.3 показаний вигляд екрану з візуалізацією об'єкту керування, розробленого за допомогою пакет математичного моделювання MATLAB [15,16].

Для програмування програмованого пристрою LOGO! використовується програма LOGOSoft V1.3 фірми SIEMENS. Ця програма дозволяє запрограмувати пристрій, а також подивитися в режимі он-лайн правильність роботи програми. Комп'ютер приєднується до LOGO! на стенді через кабель програмування LOGO!

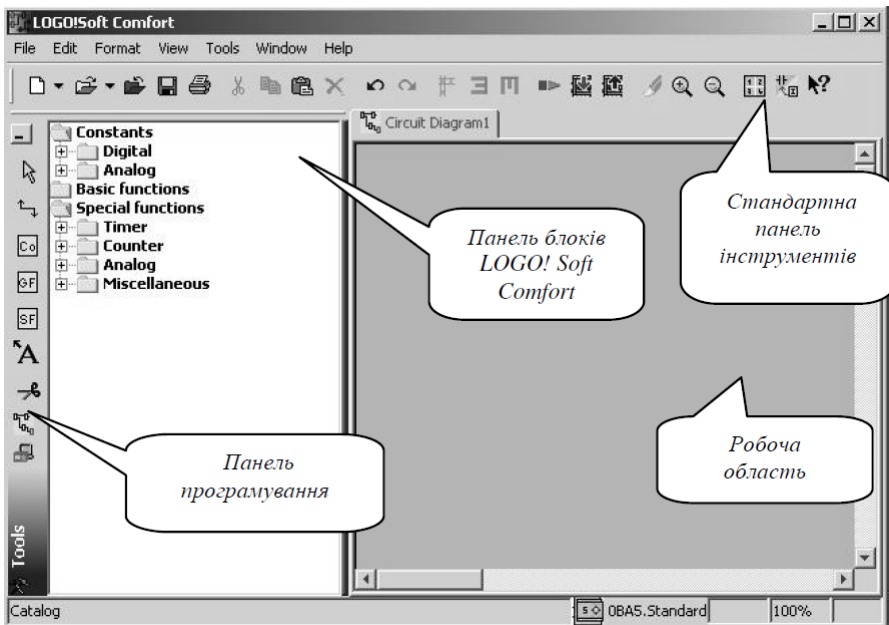


Рисунок 8.3 – Робоче вікно LOGO! Soft Comfort

Всі зміни комутацій на стенді не вимагають зміни існуючої схеми з'єднань стенду. Всі зміни настроюються програмно шляхом коригування шаблонної таблиці комутацій (таблиця Excel) відповідно до вимоги лабораторних робіт. У принципі є можливість з'єднати будь-який цифровий вихід будь-якого пристрою стенду (кнопки, перемикачі, цифрові виходи контролера і LOGO!, Цифрові виходи приводів) з будь-яким цифровим входом пристроїв стенду (лампочки, цифрові входи контролера і LOGO!, Цифрові входи приводів).

Запускаємо програму AccessPort (рис. 8.4). Потім відкриваємо порт, в панелі інструментів: Tools / Port Switch. Для передачі вмісту результату таблиці комутації в комутатор, треба зайти в панелі інструментів: Tools / Transfer file. У вікні натиснути кнопку Select file і вибираємо файл комутації (D: DA_stendconnect_1_DoZ_LOGO_70913_Коммутація_Дозіровка з красками_LOGO.txt.), Після натиснути кнопку "Send", при цьому вміст буфера передається по послідовному інтерфейсу в мікропроцесорний комутатор стенду і проводить його перекомутацію. Нова комутація, буде автоматично виконана за переданими даними мікропроцесорним комутатором стенду [15,16].

Якщо посилка пройшла успішно, то комутатор відповідає знаком точки, яка з'являється в полі буфера прийому.

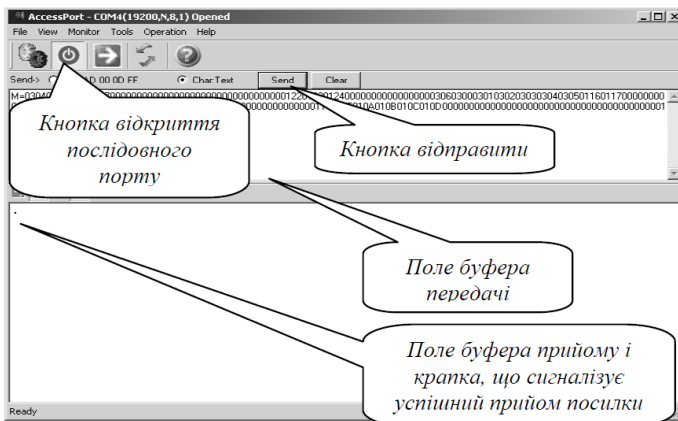


Рисунок 8.4 – Вікно програми Access Port

На рис. 8.5 показані необхідні настройки порту, через який відбувається зв'язок з комутатором стенду.

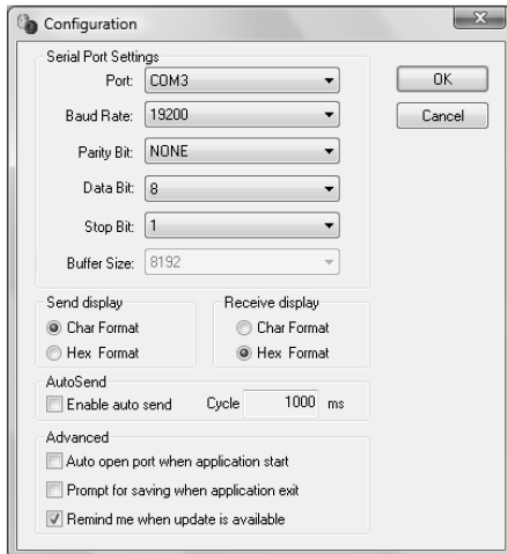


Рисунок 8.5 – Налаштування порту Access Port

Після налаштування AccessPort необхідно запустити модель дозування фарби (Simulink model). Адреса моделі: (D:\ DA_stend\models\ 2_LiquidDozing\ LiquidDozing.mdl) [15,16].

Для програмування програмованого пристрою LOGO! використовується програма LOGOSoft V5.x фірми SIEMENS. Ця програма дозволяє запрограмувати пристрій, а також подивитися в режимі онлайн правильність роботи програми.

Програмування в LOGO!

При написанні програми керування необхідно керуватися принциповими схемами наведеними в «Альбомі схем» (далі Альбом) [15,16]. Нижче наведені покрокові інструкції з написання програми керування транспортером за принципом «від простого до складного».

Першим кроком прив'яжемо роботу приводу конвеєра до органів керування (наприклад по натисканню кнопки). Нехай поки натиснута кнопка SB1 працює конвеєр, коли кнопка відпущена конвеєр не

працює. Реалізація цієї програми представлена на рис. 8.6 (згідно альбому SB1 підключена до входу I1). Набрати цю програму в LOGO! Soft Comfort.



Рисунок 8.6 – Блок-схема

Необхідно обов'язково підписувати всі кнопки, датчики, виходи (рис.8.7).

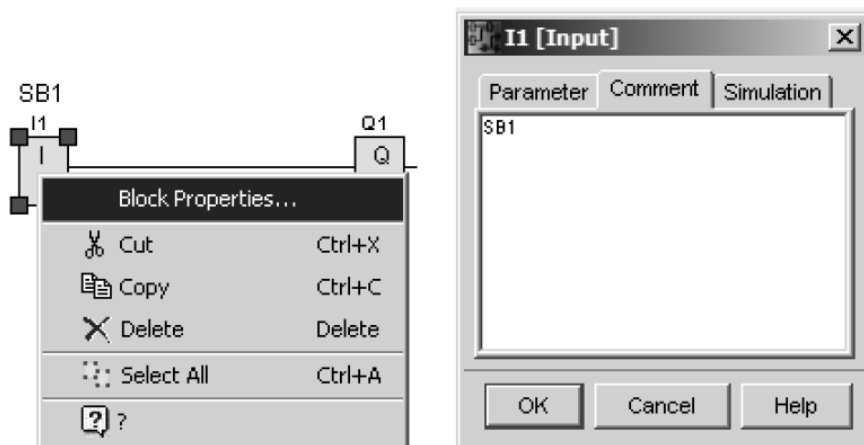


Рисунок 8.7 – Вікно коментарів

Після написання програми необхідно перевірити її роботу, для цього потрібно: записати програму в модуль LOGO!:

Вибрати в панелі програмування кнопку онлайн тест .

Якщо LOGO! знаходиться в STOP, натиснути кнопку старт , LOGO! перейде в режим онлайн роботи [15,16].

Натискаючи кнопку , відстежуємо стан кнопок, датчиків, перемикачів. Запустити модель LiquidDozing.mdl, вивести на екран візуалізації моделі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мікропроцесорна техніка: навч. посібник / В.В. Ткачов, Г.Грулер, Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. – 188 с.
2. Белов, А. В. Самоучитель по микропроцессорной технике. – СПб.: Наука и техника, 2003. – 224 с.
3. Сташин В.В. Проектирование цифровых устройств на одно кристалльных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Молногонцева – М.: Энергоатомиздат, 1990. – 224 с.
4. Карташов, Б. А. Системы автоматического регулирования с микроЭВМ / Б.А. Карташов, Е.А. Шабаев – Зерноград: АЧГАА, 2008. – 50 с.
5. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы / Под ред. В.Б. Смолова – М.: Радио и связь, 1981. – 328 с.
6. Липовецкий Г.П. и др. Однокристалльные микроЭВМ семейств МК48, МК51. – М., 1992. – 344 с.
7. Ишматов, З. Ш. Микропроцессорное управление электроприводами и технологическими объектами. Полиномиальные методы. – Екатеринбург, УГТУ-УПИ, 2007. – 278 с.
8. Магда, Ю. С. Микроконтроллеры серии 8051: практический поход. – М. ДМК Пресс, 2008. – 228 с.
9. Горюнов А. Г., Ливенцов С.Н. Архитектура микроконтроллера INTEL 8051: Учебное пособие. – Томск, Изд-во ТПУ, 2005. – 86 с.
10. Веприк, В.Н. и др. Микроконтроллеры семейства MCS-51: Учебное пособие / В.Н. Веприк, В.А. Афанасьев, А.И. Дружинин, А.А. Земсков, А.Р. Исаев, О.В. – Новосибирск, 1997. – 62 с.
11. Назарова О. С. Ідентифікація кутової швидкості при завадах в оптичній системі енкодера / О.С. Назарова, В. В. Осадчий, І. А. Мелешко, М. О. Олейніков // Вісник НТУ «ХПІ» - Харків, 2019. – С.65-69. <http://doi.org/10.20998/2079-8024.2019.16.12>
12. Sadovoi O., Nazarova O., Bondarenko V., Pirozhok A., Hutsol T., Nurek T., Glowacki Sz. Modeling and research of electromechanical systems of cold rolling mills. Monograph. – Krakow: Traicon, 2020. – 138 p.
13. Nazarova Olena. Computer Modeling of Multi-Mass Electromechanical Systems. The Third International Workshop on

Computer Modeling and Intelligent Systems (CMIS-2020), Vol. 2608, pp. 489-498. CEUR-WS.org/Vol-2608/paper36.pdf

14. Osadchyy V. and Nazarova O., "Laboratory Stand for Investigation of Liquid Level Microprocessor Control Systems," 2020 IEEE Problems of Automated Electrodrive. Theory and Practice (PAEP), Kremenchuk, Ukraine, 2020, pp. 1-4, doi: 10.1109/PAEP49887.2020.9240868

15. Методичні вказівки з виконання лабораторних робіт №1, 2, 3 "Ознайомлення зі стендом засобів автоматизації та приводи фірми SIEMENS" з дисципліни «Дискретна автоматика» для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА денної форми навчання. / Укладачі: Кулинич Е.М., Осадчий В.В. - Запоріжжя: ЗНТУ, 2017. - 70с.

16. Методичні вказівки з виконання лабораторних робіт №4, 5, 6 «Автоматизація на основі LOGO! фірми SIEMENS» з дисципліни «Дискретна автоматика» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка денної форми навчання. / Укладачі: Кулинич Е.М., Осадчий В.В. – Запоріжжя: ЗНТУ, 2017. – 46 с.

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

18. Коваленко, М. А. Автономний експериментальний стенд для випробування уніполярного крокового двигуна на базі мікроконтролера / М. А. Коваленко, Д.С. Мацюк // Електротехніка та електроенергетика. – 2015. – № 2. – С. 15-20.

19. Analog Devices – Режим доступу: <https://www.analog.com/en/index.html>.

20. Системи автоматизації для задоволення будь-яких вимог – Режим доступу: <https://new.siemens.com/ua/uk/produkty/avtomatyzatsiya-promyslovosti/systemy-avtomatyzatsiyi.html>

21. Дрешпак Н.С. Системи контролю енергоефективності виробничих процесів та шляхи їх удосконалення / Н.С. Дрешпак // Електротехніка та електроенергетика, № 1, 2020. - С. 40-48. DOI 10.15588/1607-6761-2020-1-5

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

Продовження таблиці А.2

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

Продовження таблиці А.2

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

Таблиця А.3 - Команди логічних операцій

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

Продовження таблиці А.3

Логічне АБО акумулятора і константи	ORL A,#d	2	2	1	$(A) \leftarrow (A) \vee \#d$
Логічне АБО комірки ВПД і акумулятора	ORL dir,A	3	2	1	$(dir) \leftarrow (dir) \vee (A)$
Логічне АБО комірки ВПД і константи	ORL dir,#d	7	3	2	$(dir) \leftarrow (dir) \vee \#d$
Що виключає АБО акумулятора і регістра	XRL A,Rn	1	1	1	$(A) \leftarrow (A) \vee (Rn)$
Що виключає АБО акумулятора і комірки ВПД	XRL A, dir	3	2	1	$(A) \leftarrow (A) \vee (dir)$
Що виключає АБО акумулятора і непрямоадресованої комірки ВПД	XRL A,@Ri	1	1	1	$(A) \leftarrow (A) \vee ((Ri))$
Що виключає АБО акумулятора і константи	XRL A,#d	2	2	1	$(A) \leftarrow (A) \vee \#d$
Що виключає АБО комірки ВПД і акумулятора	XRL dir,A	3	2	1	$(dir) \leftarrow (dir) \vee (A)$
Що виключає АБО комірки ВПД і константи	XRL dir,#d	7	3	2	$(dir) \leftarrow (dir) \vee \#d$
Очищення акумулятора	CLR A	1	1	1	$(A) \leftarrow 0$
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	$(\text{bit}) \leftarrow 0$
Установка переносу	SETB C	1	1	1	$(C) \leftarrow 1$
Установка біта	SETB bit	4	2	1	$(\text{bit}) \leftarrow 1$
Інверсія переносу	CPL C	1	1	1	$(C) \leftarrow (\bar{C})$
Інверсія біта	CPL bit	4	2	1	$(\text{bit}) \leftarrow (\bar{\text{bit}})$
Логічне І біта і прапору переносу	ANL C, bit	4	2	2	$(C) \leftarrow (C) \wedge (\text{bit})$
Логічне І інверсії біта і переносу	ANL C, /bit	4	2	2	$(C) \leftarrow (C) \wedge (\bar{\text{bit}})$

Продовження таблиці А.4

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

Таблиця А.5 - Команди передачі керування

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

Продовження таблиці А.5

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

Продовження таблиці А.5

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

Продовження таблиці А.5

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

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