НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 2

з курсу

Проектування мікропроцесорних систем керування технологічними процесами

Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Робота з підпрограмами

Тернопіль 2023

Методичні вказівки для виконання лабораторної роботи № 2 «Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Робота з підпрограмами» з курсу «Проектування мікропроцесорних систем керування технологічними процесами»/Укл.: Медвідь В.Р., Пісьціо В.П. - Тернопіль ТНТУ, 2023 - 9 с.

Розглянуто і затверджено на засіданні кафедри автоматизації технологічних процесів і виробництв (протокол № 1 від 30.08.2023 року)

Лабораторна робота № 2

Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Робота з підпрограмами.

1. Команди MCS51

Система команд мікроконтролера MCS51 містить 111 базових команд, які зручно розділити **за функціональною ознакою на п'ять груп: команди передачі даних, арифметичних операцій, логічних операцій, передачі управління і операцій з бітами.**

Більшість команд мають формат один або два байти і виконуються за один або два машинних циклу. При тактовій частоті 12 **МГц тривалість машинного циклу складає 1 мкс**.

Склад операндів MCS51 включає в себе **операнди чотирьох типів: біти, 4-бітні цифри, байти і 16-бітні слова.** Є також можливість адресації окремих бітів блоку регістрів спеціальних функцій (РСФ) і портів. **Для адресації бітів використовується пряма 8-бітна адреса (bit)**.

Чотирибітні операнди використовуються тільки під час операції обміну (команди SWAP і XCHD).

Восьмибітним операндом може бути комірка пам'яті програм або даних (резидентної або зовнішньої), константа (безпосередній операнд), регістри спеціальних функцій (РСФ), а також порти вводу/виводу.

Порти і РСФ адресуються тільки прямим способом. Байти пам'яті можуть адресуватися також і непрямим чином через **адресні регістри (R0, R1, DPTR і PC).**

Двобайтні операнди - це константи і прямі адреси, для подання яких використовуються другий і третій байти команди.

2. Група команд передачі управління

До даної групи команд відносяться команди, що забезпечують умовне і безумовне розгалуження, виклик підпрограм і повернення з них, а також команда порожньої операції NOP (табл. 1). У більшості команд використовується пряма адресація, тобто адреса переходу цілком (або його частина) міститься в самій команді передачі управління.

Таблиця 1. Команди передачі управління

Назва команди	Мнемокод	Операція
Довгий перехід в повному об'ємі ПП	LJMP ad16	(PC)← ad16
Абсолютний перехід всередині сторінки в 2 Кб	AJMP ad11	(PC) ← (PC) + 2, (PC_{0-10}) ← ad11
Короткий відносний перехід всередині сторінки в 256 байт	SJMP rel	(PC) ← (PC) + 2, (PC) ← (PC) + rel
Непрямий відносний перехід	JMP @A+DPTR	(PC)← (A) + (DPTR)
Перехід, якщо акумулятор рівний нулю	JZ rel	(PC)←(PC)+2, якщо (A)=0, то (PC)←(PC)+rel
Перехід, якщо акумулятор не рівний нулю	JNZ rel	(PC)←(PC)+2, якщо (A)≠0, то (PC)←(PC)+rel
Перехід, якщо флаг перенесення рівний одиниці	JC rel	(PC)←(PC)+2, якщо (C)=1, то (PC)←(PC)+rel
Перехід, якщо флаг перенесення рівний нулю	JNC rel	(PC)←(PC)+2, якщо (C)=0, то (PC)←(PC)+rel
Перехід, якщо біт рівний одиниці	JB bit, rel	(PC)←(PC)+3, якщо (b)=l, то (PC)←(PC)+rel
Перехід, якщо біт рівний нулю	JNB bit, rel	(PC)←(PC)+3, якщо (b)=0, то (PC)←(PC)+rel
Перехід, якщо біт встановлений в 1, з наступним скиданням в 0 біту	JBC bit, rel	(PC) ← (PC) + 3, якщо (b)=1, то (b) ← 0 и (PC)← (PC) + rel
Декремент регістра і перехід, якщо не нуль	DJNZ Rn, rel	(PC) ← (PC) + 2, (Rn) ← (Rn) - 1, якщо (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресованого байту і перехід, якщо не нуль	DJNZ ad, rel	(PC) ← (PC) + 2, (ad) ← (ad) - 1, якщо (ad) ≠ 0, то (PC) ← (PC) + rel
Порівняння акумулятора з прямоадресованим байтом і перехід, якщо не рівно	CJNE A, ad, rel	(PC) ← (PC) + 3, якщо (A) ≠ (ad), то (PC) ← (PC) + rel, якщо (A) < (ad), то (C) ← 1, інакше (C) ← 0
Порівняння акумулятора з константою і перехід, якщо не рівно	CJNE A, #d, rel	(PC) ← (PC) + 3, якщо (A) ≠ #d, то (PC) ← (PC) + rel, якщо (A) < #d, то (C) ← 1, інакше (C) ← 0

Назва команди	Мнемокод	Операція
Порівняння регістра з константою і перехід, якщо не рівно	CJNE Rn, #d, rel	(PC) ← (PC) + 3, якщо (Rn) ≠ #d, то (PC) ← (PC) + rel, якщо (Rn) < #d, то (C) ← 1, інакше (C) ← 0
Порівняння байту в РПД з константою і перехід, якщо не рівно	CJNE @Ri, d, rel	(PC) ← (PC) + 3, якщо ((Ri)) ≠ #d, то (PC) ← (PC) + rel, якщо ((Ri)) < #d, то (C) ← 1, інакше (C) ←0
Довгий виклик підпрограми	LCALL adl6	(PC) ← (PC) + 3, (SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC) ← ad16
Абсолютний виклик підпрограми в межах сторінки в 2 Кб	ACALL ad11	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← ($PC_{0...7}$), (SP) ← (SP) + 1, ((SP)) ← ($PC_{8...15}$), (PC_{0-10}) ← ad11
Повернення з підпрограми	RET	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Повернення з підпрограми обробки прерывания	RETI	($PC_{8...15}$) ← ((SP)), (SP) ← (SP) - 1, ($PC_{0...7}$) ← ((SP)), (SP) ← (SP) – 1
Порожня операція	NOP	(PC) ← (PC) + 1

Можна виділити три різновиди команд розгалуження по розрядності вказаної адреси переходу.

Довгий перехід. Перехід по всьому адресному простору ПП. У команді міститься повна 16-бітова адреса переходу (ad 16). Трибайтні команди довгого переходу містять в мнемокоді літеру L (Long). Всього існує дві такі команди: LJMP - довгий перехід і LCALL – довгий виклик підпрограми. На практиці нечасто виникає необхідність переходу в межах всього адресного простору і частіше використовуються укорочені команди переходу, що займає менше місця в пам'яті.

Абсолютний перехід. Перехід в межах однієї сторінки пам'яті програм розміром 2048 байт. Такі команди містять тільки 11 молодші біти адреси переходу (ad 11). Команди абсолютного переходу мають формат 2 байти. Початкова літера мнемокоду - A (Absolute). При виконанні команди в адресі наступної по порядку команди ((PC) = (PC) + 2) 11 молодших бітів замінюються на ad11 з тіла команди абсолютного переходу.

Відносний перехід. Короткий відносний перехід дозволяє передати управління в межах -128 127 байт ів адреси наступної команди (команди, наступної за командою відносного переходу). Існує одна команда безумовного короткого переходу SJMP (Short). Всі команди умовного переходу використовують даний метод адресації. Відносна адреса переходу (rel) міститься в другому байті команди.

Непрямий перехід. Команда JMP @ A+DPTR дозволяє передавати управління за непрямою адресою. Ця команда зручна тим, що надає можливість організації переходу за адресою, обчислюваною самою програмою і невідомою при написанні вихідного тексту програми.

Умовні переходи. Розвинена система умовних переходів надає можливість здійснювати розгалуження за наступними умовами: акумулятор містить нуль (JZ); вміст акумулятора не дорівнює нулю (JNZ); перенесення дорівнює одиниці (JC); перенесення дорівнює нулю (JNC); адресований біт дорівнює одиниці (JB); адресований біт дорівнює нулю (JNB).

Для організації програмних циклів зручно користуватися командою DJNZ. В якості лічильника циклів може використовуватися не тільки регістр, а й прямоадресований байт (наприклад, комірка РПД).

Всі команди цієї групи, за винятком CJNE і JBC, не впливають на флажки. Команда CJNE встановлює флажок C, якщо перший операнд виявляється меншим за другий. Команда JBC скидає флажок C у випадку переходу.

Работа з підпрограмами

Зазвичай, у вигляді підпрограм записуються багаторазово використовувані фрагменти програм, наприклад, підпрограма формування часової затримки, підпрограми реалізації спеціальних функцій, підпрограма обслуговування клавіатури, підпрограми виведення інформації і т.п.

Для звернення до підпрограм використовують команди виклику підпрограм (LCALL, ACALL), які **відносяться до групи команд передачі даних**, які на відміну від команд переходу (LJMP, AJMP) зберігають в стеку адресу повернення в основну програму.

Для повернення з підпрограми необхідно виконати команду RET. Команда RETI відрізняється від команди RET тим, що дозволяє переривання обслугованого рівня. При необхідності тимчасового зберігання результатів виконання основної програми використовується стек.

Для звернення до стеку використовують команди PUSH (запис в стек) і POP (витяг з стеку), які відносяться до групи команд передачі даних. Межа заповнення стеку визначається вмістом покажчика стеку SP (**при включенні мікроконтролера вміст SP автоматично встановлюється в 07**).

3. Завдання на самостійну підготовку

1. Дослідити команди умовних і безумовних переходів.
2. Дослідити команди входу і виходу з підпрограми.

ЗАВДАННЯ

1. Завантажити в EdSim51 та виконати команди з Прикладу 1 в покроковому режимі роботи симулятора. Звернути увагу на зміну вмісту регістрів після виконання команд програми.

Приклад 1

Інкрементувати вміст комірок РПД за адресами від 10 до 18:

```
        MOV   R0,#10        ; завантаження в R0 початкової адреси
        MOV   R3,#9         ; завантаження в R3 числа комірок пам'яті кількістю
                            ; (18-10+1)
LOOP:   INC   @R0           ; інкремент комірок РПД
        INC   R0            ; просування покажчика адреси
        DJNZ  R3,LOOP       ; декремент R3 і повтор, поки R3 не дорівнює нулю
```

Зробити і внести у звіт копію екрану з виконаною програмою з Прикладу 1.

2. Завантажити в EdSim51, виконати та дослідити програму опитування вмісту двох молодших розрядів порту P1 (D1,D0) і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в комірках 41H, 55H, 6AH і 5FH.

Перед початком асемблювання програми (натисканням кнопки ***Assm*** симулятора) необхідно записати в поле регістра P1 (на рис. 1 виділене червоним овалом) потрібне значення двох молодших розрядів (наприклад, 01), клацнувши на виділеному полі курсором**:**

Приклад 2

```
M1:
    MOV A,#00H
    CJNE A,P1,M2
    AJMP 41H
    ORG 41H
    AJMP M1
M2:
```

```
INC A
CJNE A,P1,M3
AJMP 55H
```

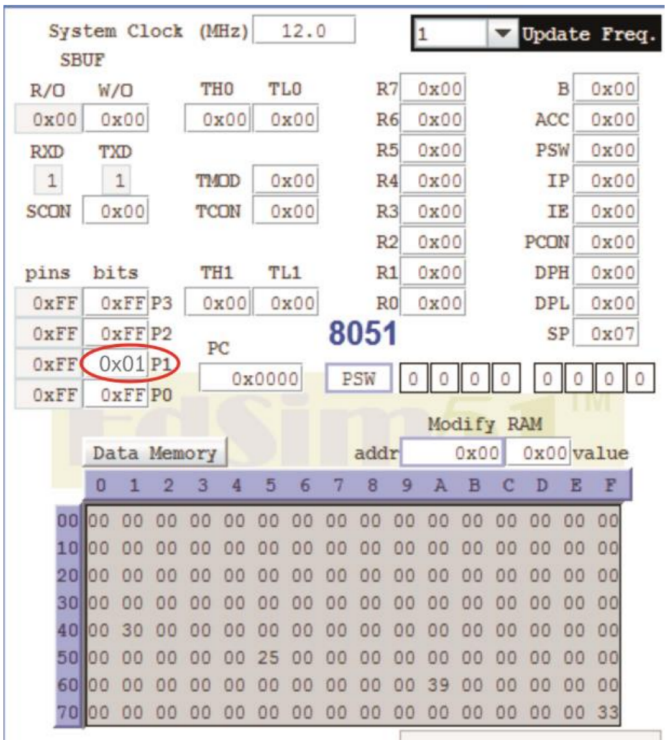

Рис. 1. Завантаження потрібного значення бітів молодших розрядів в порт P1

```
        ORG 55H
        AJMP M1
M3:
        INC A
        CJNE A,P1,M4
        AJMP 6AH
        ORG 6AH
        AJMP M1
M4:
        INC A
        CJNE A,P1,M1
        AJMP 5FH
        ORG 5FH
        AJMP M1
```

Після виконання програми записати коментарі про призначення кожної з команд за Прикладом 1. Зробити і внести у звіт копію екрану з виконаною програмою з Прикладу 2.

3. Написати і дослідити роботу команд з використанням програмної моделі відповідно до заданого варіанту:

Варіанти індивідуальних завдань

№	Зміст індивідуального завдання
1	Завантажити і виконати послідовність команд в покроковому режимі: MOV P1, #3FH; MOV R1, #C3H; MOV A,R1; PUSH P1; PUSH 01; PUSH PSW; MOV P1, #1H; MOV 01H, #02H; POP PSW; POP 01; POP P1, AJMP 01F2H.
2	Записати і виконати з використанням програмної моделі в покроковому режимі команди: ACALL 05; LJMP 20; NOP; NOP; NOP; RET; ORG 20; MOV R1,#02H; MOV A,#30H; MOV

№	Зміст індивідуального завдання
	R2,#10H.
3	Розробити програму опитування вмісту двох старших розрядів порту P2 (D7,D6) і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в осередках 25H(00), 35H(10), 4FH(01) і 10FH(11).
4	Розробити програму опитування вмісту двох молодших розрядів порту P2 (D1,D0) і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в осередках 20H(00), 30H(01), 40H(10) і 50FH(11).
5	Записати і виконати з використанням програмної моделі в покроковому режимі команди: ACALL 07; LJMP 30; NOP; NOP; NOP; NOP; NOP; RET; ORG 30; MOV R0,#02H; MOV A,#1FH; MOV R2,A.
6	Записати і виконати з використанням програмної моделі в покроковому режимі команди: MOV A,#00H; MOV R1,#10H; MOV 35,#15H; JZ 03; NOP; NOP; NOP; MOV A,#05H; MOV R7,#05H; JNZ 03; NOP; NOP; NOP; MOV A,#10H.
7	Записати і виконати з використанням програмної моделі в покроковому режимі команди: MOV A,#00H; MOV R0,#1FH; MOV B,#12H; JZ 02; NOP; NOP; MOV A,#18H; MOV R3,#16H; JNZ 02; NOP; NOP; MOV 35,#R3.
8	Завантажити і виконати послідовність команд в покроковому режимі: MOV P2, #10H; MOV R0, #25H; MOV A,R0; PUSH ACC; PUSH 01; PUSH PSW; MOV P2, #1H; MOV B, #02H; POP PSW; POP 01; POP P1, ACALL 0010H.
9	Розробити програму опитування вмісту двох старших розрядів порту P1 (D7,D6) і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в осередках 22H(00), 32H(01), 42H(10) і 52H(11).
10	Завантажити і виконати послідовність команд в покроковому режимі: MOV P2, #3FH; MOV R2, #C3H; MOV A,R1; PUSH P2; PUSH ACC; PUSH PSW; MOV P1, #1H; MOV 01H, #02H; POP PSW; POP ACC; POP P2, AJMP 0101H.
11	Завантажити і виконати послідовність команд в покроковому режимі: MOV P3, #10H; MOV R5, #C3H; MOV A,R5; PUSH P3; PUSH R5; PUSH PSW; MOV P1, #1H; MOV 01H, #02H; POP PSW; POP R5; POP P3, AJMP 16E2H.
12	Розробити програму опитування вмісту двох молодших розрядів порту P3 (D1,D0) і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в осередках 33H(00), 49H(01), 60H(10) і 72H(11).

4. Записати в звіт зміни в вікнах регістрів мікроконтролера при виконанні програми завдання вибраного варіанту **для перших десяти команд** відповідно до табл. 2.

Таблиця 2 - Результати виконання команд

№	Команда	Виконувана операція	Вміст використовуваних регістрів та комірок пам'яті до і після виконання		Пояснення
			До	**Після**	
1	MOV A,R0	Пересилання байту даних з регістру R0 в акумулятор A	A/00 PC/00 PC/01	A/F2 PSW/00 PSW/01	
2	…	…	…	…	…
…	…	…	…	…	…
5	…	…	…	…	…

5. Додати у звіт копію екрану з виконаною програмою на програмному симуляторі відповідно до обраного варіанту.

6. Послідовність виконання роботи

6.1. Вивчити команди пересилання. Вивчення кожної команди проводити наступним чином:

6.1.1. Відкрити інтерфейс симулятора, двічі клацнувши клавішею миші на архівованому файлі «EdSim51.jar". Відкриється інтерфейс програмного симулятора, зображений на рис.2.

Середнє поле симулятора, що називається **"Панель коду Асемблера"**, в верхній частині містить кнопки **"*Reset*", "*Assm*", "*Run*", "*Load*", "*Save*", "*Copy*", "*Past*"**.

Панель коду використовується для:

- **набору команд** програми з клавіатури. Для цього курсор встановлюється в верхній частині панелі і вводиться програма по одній команді в рядку (при потребі, з міткою та коментарем) (див. рис.2). У цьому випадку перша команда після асемблювання запишеться в пам'ять програм за адресою 00h. В іншому випадку, директивою **ORG** перед першою командою необхідно задати початкову адресу команди. Наприклад, запис **ORG 30h** завантажить перший байт команди програми в пам'ять програм за адресою 30h;
- **завантаження** вже існуючої програми. Для цього необхідно на панелі вгорі натиснути кнопку **"*Load*"** і вказати шлях до потрібного файлу;
- **запису** набраного файлу. Для цього потрібно натиснути кнопку **"*Save*"** і вказати шлях для збереження файлу.

6.1.2. Перед виконанням програми необхідно натиснути кнопку **"*Assm*"** панелі для асемблювання програми. Після цього, якщо команда записана не вірно, в рядку під верхнім рядом кнопок панелі (на рис.1 виділений сірим кольором) з'явиться повідомлення про помилку, а **колір рядка зміниться на червоний**. Червоним кольором буде виділена також не вірно написана команда.

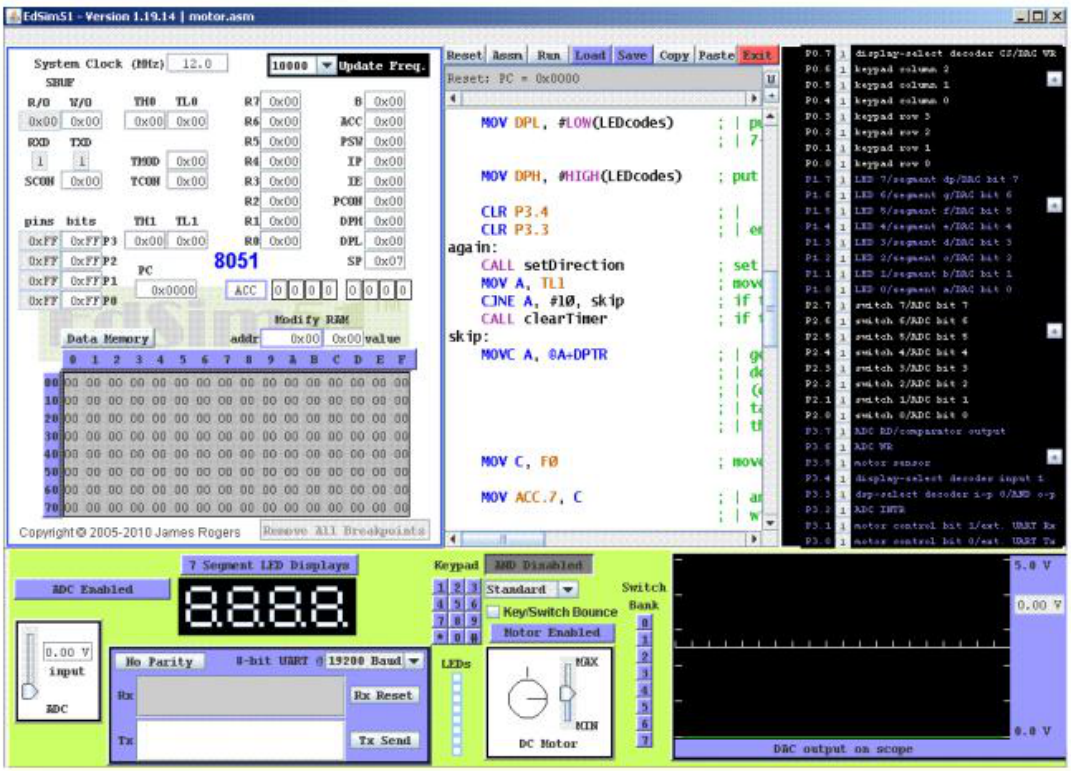

Рис. 2. Інтерфейс програмного симулятора EdSim51

Якщо помилки відсутні, зліва від команд набраної програми з'являться адреси, і сама програма буде готова до виконання. Після асемблювання кнопка **"*Assm*"** зміниться на кнопку **"*Step*"**. Таким чином, є можливим виконувати програму покомандно **в кроковому режимі**, натискаючи кнопку **"*Step*"**після виконання кожної команди, або **в автоматичному режимі**, коли виконується вся програма, натиснувши один раз кнопку **"*Run*"**. В останньому випадку програму слід закінчувати директивою ***"End".***

При написанні програми можна користуватися для копіювання її фрагментів та вставки в будь-якому місці "Панелі коду Асемблера" кнопками **"*Copy*"** *та* **"*Past*"**.

Щоб зупинити виконання програми і скинути в початковий стан регістри мікроконтролера симулятора, необхідно натиснути кнопку **"*Reset*".**

***Примітка.**

1. Якщо ви хочете виконати якусь з команд пересилання, наприклад, з регістра в регістр, необхідно попередньо в регістр, з якого буде здійснене пересилання, командою MOV попередньо записати якесь значення операнду (адресу чи константу).
2. Програма, що виконується, буде записана в пам'ять програм, вміст якої можна побачити, натиснувши на кнопку ***"Data memory"*** в нижній частині **"Панелі пам'яті даних та програмної пам'яті"**, що знаходиться зліва від **"Панелі коду Асемблера"**. Після натискання кнопка ***"Data memory"*** зміниться на кнопку ***"Code memory"***, тобто буде висвічуватися в полі пам'яті вміст пам'яті програм.
3. Область пам'яті, що відводиться під стек, буде відображатися в полі ***"Data memory", починаючи з адреси 07h***.

7. Контрольні запитання

1. Пояснити роботу команди PUSH.
2. Пояснити роботу команди POP.
3. Пояснити роботу команди CALL.
4. Пояснити роботу команди RET.
5. Пояснити роботу команд умовного переходу на підпрограми.

Рекомендована література

1. Проектування мікропроцесорних систем керування : навчальний посібник, перевидання / Медвідь В.Р., Пісьціо В.П., Козбур І.Р. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 360 с.

2. Handbook of Microcontrollers/Predko Michael. NYc. McGraw-Hill. 1998. 861 p.

3. Бойко В. І., Гуржій А. М., Жуйков В. Я. та ін.Схемотехніка электронних схем: У 3 кн. Кн.3 Мікропроцесори та мікроконтролери: підручник. 2-ге вид., допов. і переробл. К.: Вища шк., 2004. 399 с.

4 Мілих В. І., Шавьолкін О. О. Електротехніка, електроніка та мікропроцесорна техніка: підручник; за ред. В. І. Мілих. 2-е вид. К.: Каравела, 2008. 688 с.