НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет
імені Івана Пулюя

Кафедра автоматизації
технологічних
процесів та виробництв

Методичні вказівки для виконання лабораторної роботи №2 “Програмування мікроконтролера I8051 з використанням програмної моделіEdSim51. Робота з підпрограмами” з курсу “Розробка систем керування на основі ОМЕОМ”

Тернопіль 2016

Методичні вказівки для виконання лабораторної роботи №2 «Програмування мікроконтролера I8051 з використанням програмної моделі EdSim51. Робота з підпрограмами» з курсу «Розробка систем керування на основі ОМЕОМ».

Методичні вказівки розглянуті і схвалені кафедрою «Автоматизація технологічних процесів і виробництв», протокол № 4 від 21.11.2016 р.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.

асистент Пісьціо В.П.

Лабораторна робота №2

Програмування мікроконтролера I8051 з використанням програмної моделі EdSim51. Робота з підпрограмами

1. Команди МК51

Система команд мікроконтролера МК51 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 МК51 включає в себе **операнди чотирьох типів: біти, 4-бітові цифри, байти і 16-бітові слова.** Є також можливість адресації окремих бітів блоку регістрів спеціальних функцій (РСФ) і портів. **Для адресації бітів використовується пряма 8-бітова 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 (RO, Rl, 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 повном уоб'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 в середині 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, если (A)=0, то (PC)←(PC)+rel
Перехід, якщо акумулятор не рівний нулю	JNZ rel	(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel
Перехід, якщо флаг перенесення рівний одиниці	JC rel	(PC)←(PC)+2, если (C)=1, то (PC)←(PC)+rel
Перехід, якщо флаг перенесення рівний нулю	JNC rel	(PC)←(PC)+2, если (C)=0, то (PC)←(PC)+rel
Перехід, якщо біт рівний одиниці	JB bit, rel	(PC)←(PC)+3, если (b)=l, то (PC)←(PC)+rel
Перехід, якщо біт рівний нулю	JNB bit, rel	(PC)←(PC)+3, если (b)=0, то (PC)←(PC)+rel
Перехід, якщо біт встановлений в 1, з	JBC bit, rel	(PC) ← (PC) + 3, если (b)=1,

Назва команди	Мнемокод	Операція
наступним скиданням в 0 біту		то (b) ← 0 и (PC)← (PC) + rel
Декремент регістра і перехід, якщо не нуль	DJNZ Rn, rel	(PC) ← (PC) + 2, (Rn) ← (Rn) - 1, если (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресованого байту і перехід, якщо не нуль	DJNZ ad, rel	(PC) ← (PC) + 2, (ad) ← (ad) - 1, если (ad) ≠ 0, то (PC) ← (PC) + rel
Порівняння акумулятора з прямоадресованим байтом і перехід, якщо не рівно	CJNE A, ad, rel	(PC) ← (PC) + 3, если (A) ≠ (ad), то (PC) ← (PC) + rel, если (A) < (ad), то (C) ← 1, иначе (C) ← 0
Порівняння акумулятора з константою і перехід, якщо не рівно	CJNE A, #d, rel	(PC) ← (PC) + 3, если (A) ≠ #d, то (PC) ← (PC) + rel, если (A) < #d, то (C) ← 1, иначе (C) ← 0
Порівняння регістра з константою і перехід, якщо не рівно	CJNE Rn, #d, rel	(PC) ← (PC) + 3, если (Rn) ≠ #d, то (PC) ← (PC) + rel, если (Rn) < #d, то (C) ← 1, иначе (C) ← 0
Порівняння байту в РПД з константою і перехід, якщо не рівно	CJNE @Ri, d, rel	(PC) ← (PC) + 3, если ((Ri)) ≠ #d, то (PC) ← (PC) + rel, если ((Ri)) < #d, то (C) ← 1, иначе (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 переривання	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 вказуваної адреси переходу.

Довгий перехід. Перехід по всьому адресному простору ПП. У команді міститься повна 16-бітова адреса переходу (ad 16). Трибайтові команди довгого переходу містять в мнемокоді літеру L (Long). Всього існує дві такі команди: LJMP - довгий перехід і 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.

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 флаги. Команда CJNE встановлює флаг C, якщо перший операнд виявляється меншим за другий. Команда JBC скидає флаг C у випадку переходу.

Работа с підпрограмами

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

Для звернення до підпрограм необхідно використовувати команди виклику підпрограм (LCALL, ACALL), які відносяться до групи команд передачі даних. Ці команди 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 використовуються 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

1. Дослідити команди запису в стек і витягу із стеку: PUSH P1; PUSH 01; PUSH PSW; POP PSW; POP 01; POP P1.
2. Дослідити команди входу і виходу з підпрограми.
3. Написати і дослідити роботу підпрограми зведення в квадрат цілого числа.
4. Написати і дослідити програму опитування вмісту двох молодших розрядів порту P1 і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в комірках 21H, 25H, 3AH і 2FH.

4. Послідовність виконання роботи

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

4.1.1. Відкрити інтерфейс емулятора, двічі клацнувши клавішею миші на архівованому файлі «EdSim51.jar". Відкриється інтерфейс програмного емулятора, зображений на рис.1.

Середнє поле емулятора, 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.1). 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.

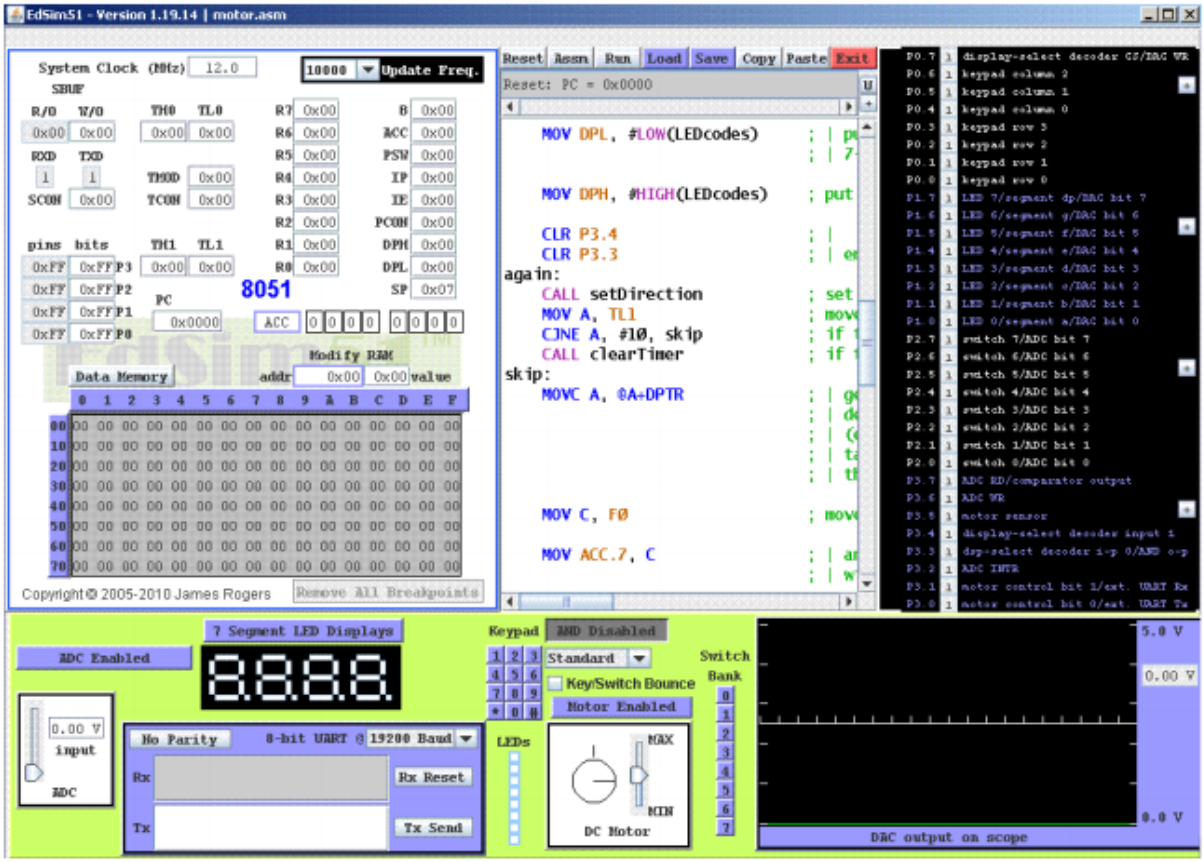

Рис.1 Інтерфейс програмного емулятора EdSim51

4.1.2. 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.

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 командою ***"Stop".***

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 емулятора, необхідно натиснути кнопку **"*Reset*".**

4.1.3. Записати в звіт зміни в вікнах регістрів мікроконтролера відповідно до табл. 2.

4. Послідовність виконання роботи

1. Завантажити послідовність команд: MOV P1, # 3FH; MOV R1, # C3H; MOV A, R1; PUSH P1; PUSH 01; PUSH PSW; SETB RS1; MOV P1, # 1H; MOV 01H, # 02H; POP PSW; POP 01; POP P1.

2. Виконати послідовність команд в покроковому режимі.

3. Записати в звіт згідно з табл. 2 виконувані команди і вміст змінюваних регістрів і комірок пам'яті.

4. Завантажити, починаючи з адреси 0000, послідовність команд: ACALL 05; LJMP 20; NOP; NOP; NOP; RET.

5. Виконати програму, що складається з даної послідовності команд в покроковому режимі.

6. Записати в звіт згідно з табл. 2 виконувані команди і вміст змінюваних регістрів і комірок пам'яті.

7. Розробити і ввести підпрограму зведення в квадрат числа з використанням команди MUL AB. Для звернення до підпрограми використовувати команду ACALL.

8. Переконатися в працездатності підпрограми.

9. Записати в звіт згідно з табл. 2 виконувані команди і вміст змінюваних регістрів. Визначити час виконання підпрограми.

10. Розробити програму опитування вмісту двох молодших розрядів порту P1 і переходу в залежності від їх стану до однієї з чотирьох підпрограм, початкові адреси яких знаходяться в осередках 21H, 25H, 3AH і 2FH

11. Записуючи в порт різні кодові комбінації в використовуваних розрядах, перевірити працездатність програми.

Таблиця 2. Результати вконання команд

№	Команда	Код	Виконувана операція	Вміст регистрів і пам'яті до і після виконання		Пояснення
				До	Після	
1	MOV A,R0	E8	Пересилання байту даниих з регістру R0 в акумулятор A	A/00 R0/F2 0000/F2	A/F2 R0/F2 0000/F2	
2	…	…	…	…	…	…

***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***.

5. 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 підпрограми зведення в квадрат.
6. Пояснити роботу програми умовного вибору підпрограми.