МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Мікропроцесорні пристрої керування та обробки інформації

Методичні вказівки до комп'ютерного практикуму

Частина 1

Київ – 2007

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Мікропроцесорні пристрої керування та обробки інформації

Методичні вказівки до комп'ютерного практикуму

Частина 1

для студентів спеціальностей 7.0908.03 «Електронні системи»,
7.0908.01 «Мікроелектроніка і напівпровідникові прилади»,
7.0908.04 «Фізична та біомедична електроніка»,
7.0912.01 «Акустичні засоби та системи»,
7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати»
усіх форм навчання

Затверджено Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2007

Мікропроцесорні пристрої керування та обробки інформації: Метод. вказівки до комп'ютерного практикуму для студ. спец. 7.0908.03 "Електронні системи", 7.0908.01 «Мікроелектроніка і напівпровідникові прилади», 7.0908.04 «Фізична та біомедична електроніка», 7.0912.01 «Акустичні засоби та системи», 7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати» усіх форм навчання. / Уклад. В.А. Тодоренко, Л.М. Батрак. - К.: НТУУ "КПІ", 2007. – 44 с.

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № 4 від 21.12.2006 р.)

Навчальне видання

Мікропроцесорні пристрої керування та обробки інформації

Методичні вказівки до комп'ютерного практикуму

для студентів спеціальностей 7.0908.03 "Електронні системи",
7.0908.01 «Мікроелектроніка і напівпровідникові прилади»,
7.0908.04 «Фізична та біомедична електроніка»,
7.0912.01 «Акустичні засоби та системи»,
7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати»
усіх форм навчання

Укладачі: *Тодоренко Віктор Агафонович,* канд. техн. наук, доц.
Батрак Лариса Миколаївна

Відповідальний редактор: *В. Я. Жуйков,* д-р техн. наук, проф.

Рецензент: *С. Р. Михайлов*, канд. техн. наук, доц.

ЗМІСТ

ВСТУП

В умовах сучасного розвитку однією з основних базових дисциплін у вивченні електроніки є „Мікропроцесорні пристрої керування та обробки інформації”.

Вивчення цієї дисципліни, як і будь-якої іншої, не можливе без застосування теоретичного матеріалу на практиці.

Метою комп’ютерного практикуму з курсу „Мікропроцесорні пристрої керування та обробки інформації” є закріплення теоретичних знань з архітектури, системи команд програмування мовою Асемблеру для мікроконтролерів сімейства MCS-51 та набуття практичних навичок з цього виду програмування.

Готуючись до кожного практикуму, необхідно вивчити теоретичні відомості, ознайомитися з програмним забезпеченням та підготувати відповідні розділи звіту.

Робота містить методичні вказівки до п'яти практикумів (мету практикуму, варіанти завдань, порядок виконання практикуму, зміст звіту, контрольні питання, теоретичні відомості).

В процесі виконання практикумів робіт студенти повинні навчитися програмувати мікроконтролер у ручному режимі, програмувати та налагоджувати найпростіші програми за допомогою фірмового інтегрованого програмного забезпечення середовища виробника, застосовувати принципи модульного, структурного програмування.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №1

ДОСЛІДЖЕННЯ ПРОГРАМНОГО СЕРЕДОВИЩА РОЗРОБКИ ТА НАЛАГОДЖЕННЯ ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Мета практикуму:

- ознайомитися з основними прийомами написання програм, із використанням мови асемблеру мікропроцесорів сімейства MCS-51;
- ознайомитися з програмним середовищем μVision, призначеним для розробки та відпрацювання прикладних програм.

2. Програма практикуму:

2.1. У програмному середовищі μVision написати тестову програму, орієнтовний текст якої наведено нижче. Провести асемблювання програми, відкоригувати помилки.

2.2. З використанням програмного симулятора, виконати програму. Прослідкувати за зміною даних в регістрах SFR та RAM мікропроцесора.

3. Завдання до практикуму

В таблиці 1.1 наведені вихідні дані для виконання практикуму. Результати виконання слід занести до таблиці 1.2

Таблиця 1.1

1.	org 10h
	mov A, 100
	mov B, @35h
	setd PSW.7
	addc A, B
	jo v1
	mov 30, A
	jmp v1
	v1: or A, #FF
	v1: nop
	end
		6.	org 10h
			mov A,100
			mov 20, AA
			add A 20
			set PSW.0
			ja v1
			mov R9, A
			jmp v1
			v1: mov R0, 20
			v3: nop
			end
				11.	org 10h
					mov PSW, 100
					mov A, @R5
					and A,D
					jc v1
					clr C, A
					jmp v1
					v1: movx R0, A
					rl B
					v3: nop
					end

Продовження табл. 1.1

2.	org 20h
	mov PSW, 96h
	mov A, @R2
	or A,#FF
	jb m2
	mov B,A
	jmp v2
	m2: mov R8 A
	v1: swap B
	end
		7.	org 20
			mov R8, 20
			mov A,R2
			xor A,#FF
			jb PSW, m2
			mov B,A
			jmp v1
			m2: mov PSW, 20h
			v1: rr R0
			end
				12.	org 20h
					mov PSW, 78h
					mov C,R2
					anl PSW,#DF
					jp m4
					mov R8,C
					jmp v1
					m2: mov @R7, A
					v1: nop
					end

3.	org 30
	mov B, 100
	mov C, 88h
	orl B, C
	jnb m1
	mov A, B
	mov R9, 01
	jmp m2
	m2: mov 10H, GF
	end
		8.	org 30h
			mov B,100
			mov R0, @88h
			sub PSW,C
			jnb m1
			mov A,B
			mov R9, 01
			jmp m2
			m2: xch 2H,GF
			end
				13.	org 30h
					mov A, 100
					mov @Ri, #88
					orl PSW, A
					djnz m1
					mov A, B
					mov R9, 01
					jmp m2
					m2: xchd 10H,GF
					end

4.	org 40h
	mov A,56h
	mov B,78h
	xrl A, B
	djnz m1
	mov 20, PSW
	set 00, 0ffh
	jmp m2
	m1: inc A, D
	m2: end
		9.	org 40h
			mov A, 64
			mov R6, 35h
			addc A, R6h
			jnc v1
			mov D, A
			set A.3
			jmp v1
			v1: clr R6.4
			v1: end
				14.	org 40
					mov A, 64
					mov R6, 35h
					addc A, R6h
					jnc v1
					mov D, A
					set A.3
					jmp v1
					v1: div R6.4
					v1: end

5.	org 50h
	mov R0,f5h
	mov A,8eh
	and R0,A
	jb m1
	rlc A
	jmp m2
	m1: mov R3, A
	m2: dec B,0ffh
	end
		10.	Org 50h
			mov A ff
			mov R3 e8
			anl A,R3
			jnb m2
			cpl B
			jmp v1
			m2: mov R8 A
			v1: set HSW.0
			end
				15.	org 50h
					mov A ff
					mov R8 8
					cjne A,R3
					cpl m2
					mov B,A
					jmp v1
					m2: mov Ri, A
					v1: clr PSW.8
					end

Таблиця 1.2.

Адреса ПЗУ	Команда	Вміст регістрів Вміст комірок пам'яті						PSW						
		1	2	3	4	5	6	CY	A	F	RS1	RS0	OV	P

4. Зміст звіту:

- Титульний лист з відомостями про назву практикуму і склад бригади;
- Текст програми з коментарями.

5. Теоретичні відомості

Нині існує досить багато пакетів програм, що дозволяють вести розробку та програмну симуляцію прикладних програм для мікропроцесорів сімейства MCS-51.

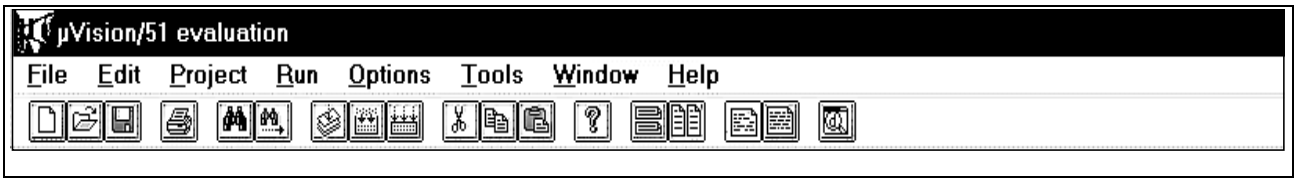
Серед програмних середовищ вирізняється продукція Keil Software, що розробляється до мікропроцесорів сімейств MCS-51, MCS-251, MCS-166. Програмні середовища цього виробника орієнтовані на використання MS Windows. До складу інтегрованого середовища (MCS-51) входять: менеджер проектів, текстовий редактор, компілятори з мов асемблера, C та PLM, засоби для інтерактивної корекції помилок, програмний симулятор, внутрішньосхемний емулятор процесора, емулятор ROM, та засоби для визначення ефективності використання мікропроцесора. Поєднання в одному програмному середовищі трьох компіляторів допускає написання фрагментів прикладних програм як на мовах високого так і низького рівня.

В програмне середовище Keil Software для розробки прикладних програм мікропроцесорів сімейств MCS-51 входять дві програми:

- компілятор μ-VISION/51;
- програмний симулятор dScope-51.

Запуск програми μ-VISION/51 відбувається в середовищі Windows, з вікна програм Keil PK51-Eval.

Після запуску на екрані монітору з`являється типове вікно Windows з рядком заголовку вікна, кнопками системного меню, згортання, мінімізації та розгортання. В цьому вікні розташовані рядок меню програми та лінійка кнопок керування основними операціями програми μ-VISION/51.

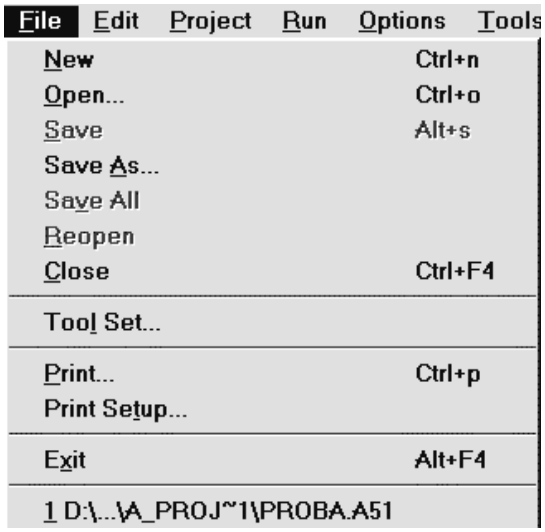


Лінійка кнопок дозволяє прискорити виконання основних команд в програмі μ-VISION/51:

- створення нового файлу - New file;
- відкриття нового файлу - Open;
- запису файлу - Save;
- друкування на принтері тексту файлу - Print;
- пошуку фрагменту тексту - Find;
- повторення пошуку того самого фрагменту тексту - Repeat Find;
- компіляції активного файлу - Compile;
- компіляції та запису файлів проекту, що змінювалися - Update;
- компіляції та запису всіх файлів проекту - Build All;
- видалення в буфер обміну позначеного фрагменту тексту - Cut;
- копіювання позначеного тексту в буфер обміну без видалення - Copy;
- копіювання на позицію курсору тексту з буферу обміну - Paste;
- виклик довідкової системи - Help;
- горизонтальне розділення робочого поля на вікна - Tile Horisontally;
- вертикальне розділення робочого поля на вікна - Tile Vertically;
- ввімкнення\вимкнення кольорової палітри - Color Syntax;

- ввімкнення\вимкнення показу критеріїв пошуку в діалоговому вікні - Show occurences;
- підключення програмного симулятора dScope-51 - Debug.

Команда меню **File** викликає список команд, що використовуються для відкриття, запису або друкування файлів. В цьому списку існують команди, що дозволяють створювати новий файл (New), відкривати існуючий (Open) або повторно загружати в комп'ютер відкритий файл (Reopen), якщо в ньому були будь які зміни. Можливий також прискорений доступ до відкривання нещодавно використаних файлів, інформація про які надається в кінці списку команд.



Для запису файлів використовуються команди запису відкритого файлу без можливості модифікації його імені та місця знаходження (Save), та з можливістю таких змін (Save As). Існує також команда групового запису всіх відкритих файлів (Save All).

Закриття робочого файлу відбувається за умови звертання до команди Close.

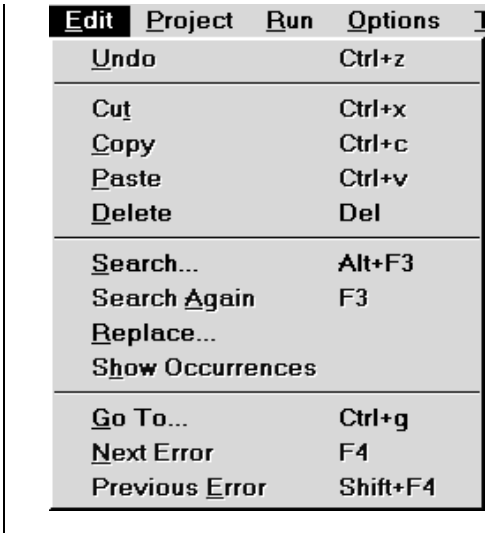
Для друкування тексту відкритого файлу, використовуються команди керування режимами роботи принтера (Print, Print Setup).

Команда Tool Set використовується для підключення різноманітних типів компіляторів, що використовуються для асемблеювання програм.

Для виходу з програми μ-VISION/51 використовується команда Exit.

Команда меню **Edit** викликає список команд, що використовуються для редагування тексту файлу. Серед них виділяють команди операцій з використанням буферу обміну- видалення позначеного тексту (Cut), копіювання позна-

ченого тексту без його видалення (Copy), вставки тексту з буферу на позицію маркеру.



Для знищення позначеного тексту використовується команда Delete.

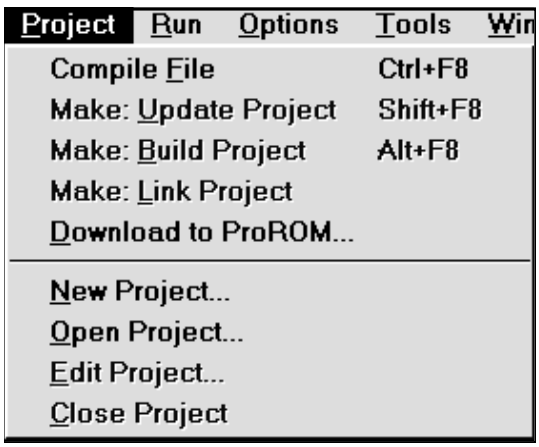
В редакторі існує можливість відміни замін в тексті , що були зроблені з моменту останнього запису файлу (Undo).

Редактор програми μ-VISION/51 дозволяє проводити як пошук фрагментів тексту (Search та Search Again), так і їх заміну (Replace). Для прискорення пошуку таких фрагментів по всьому тексту програми, може використовуватися команда Show Occurences. Ця команда викликає кольорову підсвітку по всьому тексту програми фрагменту тексту, що задається в вікні Search.

Команда Go To дозволяє задати номер лінії тексту та здійснити прискорений перехід в цю точку програми.

Під час роботи над помилками можуть бути корисними дві команди Next Error та Previous Error, які переміщують маркер в активному вікні програми що редагується до наступної, або попередньої помилки.

Команда меню **Project** викликає список команд, що використовуються для відкриття, редагування, запису та закриття файлів проектів; побудови або поновлення проектів та підключення інших файлів проектів. В разі необхідності створення програми, що поєднує декілька файлів, використовується методика роботи з проектами.

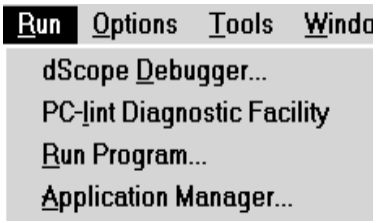


Створення нового проекту, надання

йому імені та місця розташування можливе з використанням команди New Project.

Команда Open Project служить для відкриття вже існуючого проекту. Вона відкриває діалогове вікно, де вказується ім`я файлу проекту. Команда Edit Project відкриває діалогове вікно, яке дозволяє змінити специфікації файлів, підключених до проекту. Для закриття проектів використовується команда Close Project.

Для проведення компіляції окремих файлів, розташованих в активному вікні програми, використовується команда Compile File. Команда Update Project дозволяє провести компіляцію лише тих файлів проекту, що набули змін. Команда Build Project використовується для компіляції всіх файлів, що входять до проекту, та поновлення файлу проекту.



Команда меню **Run** викликає список команд для запуску різноманітних програм, що дозволяють виконувати програму активного вікна.

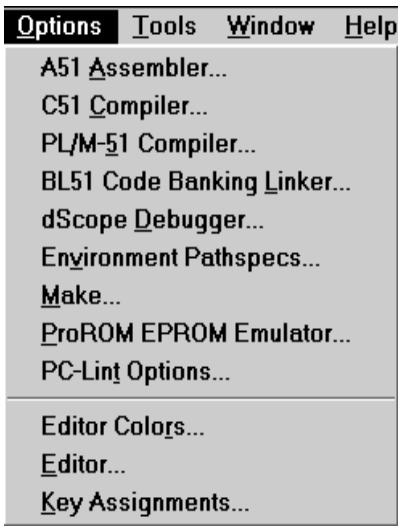
Можливі наступні варіанти виконання прикладної програми:

* за допомогою програмного симулятора;
* за допомогою зовнішнього внутрішньосхемного емулятора;
* за допомогою емулятора пам`яті програм, або програматора, якщо мікропроцесорна система використовує зовнішній постійний запам`ятовуючий пристрій для зберігання кодів команд програми;
* за допомогою ресурсів персонального комп’ютера.

Вибір команди dScope Debugger забезпечує виконання програми в середовищі програмного симулятора dScope-51, що входить до пакету програм. За допомогою команди PC-lint Diagnostic Facility можна проаналізувати можливість виконання програм, укладених на мові Сі, із використанням ресурсів персонального комп’ютера. Команда меню Application Manager дозволяє підключити до

Run-меню програми керування зовнішнім внутрішньосхемним емулятором, програматором. Ці програми активізуються в разі їхньому виклику за допомогою команди Run Program.

Команда меню **Options** викликає список команд, що використовуються для настройки режимів роботи складових частин програми μ-VISION/51 - крос-асемблера A51, компіляторів C51 та PL/M-51, лінковщика BL51, програмного симулятора dScope-51, EPROM - емулятора та редактора.



Команда Environment Pathspecs викликає діалогове вікно, що дозволяє встановити шляхи для розташування основних робочих та тимчасових файлів проектів.

Команда Make дозволяє встановити пакетну послідовність програм по обробці даних, після виконання компіляції одиночного файлу, або перебудови проекту. Ця команда також дозволяє підключити після завершення програмної обробки даних різноманітне зовнішнє обладнання - емулятор процесора, емулятор EPROM, програматор.

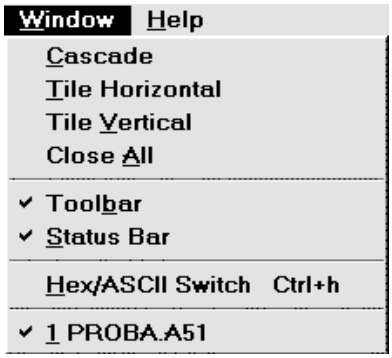
Команда PC-lint Option дозволяє встановити назву та місце розташування файлу конфігурації та місце розташування тих файлів прикладних програм, що мають виконуватись. В файлі конфігурації описаний набір опцій, який дозволяє використати персональний комп'ютер для виконання прикладної програми написаної на мові Сі.

Команди Editor, Editor Color та Key Assigment призначені для встановлення конфігурації редактора, кольорової палітри підсвіток та кнопкових комбінації для прискорення виконання основних операції в середовищі редактора.

оманда меню **Tools** викликає дві команди, що використовуються для роботи з програмами написаними на мові Сі. Команда Check C Braces виконує пе-

ревірку парності дужок. Команда Insert Template дозволяє використати зразки деяких операторів мови, таких як *for, while, do, switch*.

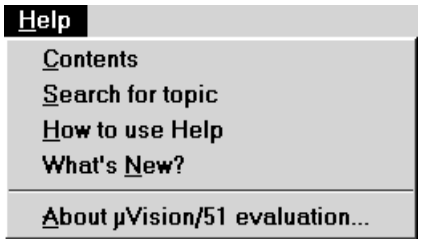
Команда меню **Window** використовується для настройки віконного середовище програми. Вона дозволяє розташувати вікна з відкритими файлами трьома способами: каскадного (Cascade), поділити простір між вікнами на рівні частини горизонтально (Tile Horisontal), або вертикально (Tile Vertical). Можливе використання списку відкритих файлів, для переходу від одного вікна до іншого.



В цьому меню розташовані також команди Toolbar та Status Bar, які дозволяють підключати лінійку кнопкових перемикачів прискореного доступу до певних команд та лінійку статусу, де є інформація про стан програми.

Команда Hex/ASCII Switch дозволяє змінювати формат кодування тексту програми.

Команда меню **Help** дозволяє отримати інформацію про порядок роботи з програмою (Contents), провести пошук по ключовим словам для отримання довідки (Search for topic).



В розділі How to use Help надана інформацію про порядок використання довідкової системи.

Команда What`s New дозволяє отримати інформацію про зміни, що відбулися в програмі µ-VISION/51 по відношенню до попередніх версій.

В розділі About µ-VISION/51 evaluation наведено загальні відомості про програму.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2

ДОСЛІДЖЕННЯ АРИФМЕТИЧНИХ ТА ЛОГІЧНИХ ОПЕРАЦІЙ

1. Мета практикуму:

Ознайомитися з основними арифметичними та логічними операціями мікропроцесорів сімейства MCS-51.

2. Програма практикуму:

2.1. В програмному середовищі μ-VISION/51 написати, асемблювати та відладити програму, яка наведена в таблиці 2.1;

2.2. З використанням програмного симулятора dScope-51 крок за кроком виконати програму. Прослідкувати за зміною даних у відповідних регістрах SFR.

3. Завдання до практикуму

В таблиці 2.1 наведені вихідні дані для виконання практикуму.

Таблиця 2.1.

Номер варіанту	Завдання	Номер варіанту	Завдання
1	Z = (3760 - x) *4 + T ∧ y	**11**	Z = 1024 + 3X -Y- 1⊕ T
2	Z = (4200 – y:8 –1) ⊕ X	**12**	Z = (8650 + 2X) :84∨ T + y
3	Z = (7650+Y) *4 – T ∧ X	**13**	Z = (7888 - Y/4) - (T ⊕ X)
4	Z = (7856-y/4) *3 + T	**14**	Z = (1256 ⊕ X) + T * 4+ y
5	Z = (5640 + 2y) * 4 ∨ T	**15**	Z = (2300 -y – x) *4∧ T
6	Z = (6543 + 6y ∧ x) *2 + T	**16**	Z = (7848 - Y/4) + (T ⊕ X)
7	Z = (3421 ∧ x) *2 - T	**17**	Z = (5430 -y – x) *8 ∧ T
8	Z = (2004 ∧ y) *2 – T ∨ X	**18**	Z = (2556 ⊕ X) – T * 8 + y
9	Z = (4000 – 4X) ∨ Y *5	**19**	Z = (8970 + 4X) :8 ∨ T-y
10	Z = (4200 ⊕ X) + T * 8 - y	**20**	Z = 1076 - 3X +Y- 1⊕ T

4. Зміст звіту:

- Титульний лист з відомостями про назву практикуму і склад бригади;
- Текст програми з коментарями.

5. Контрольні запитання:

- Назвіть основні групи команд арифметичних та логічних операцій. Які операції виконуються лише з байтовими числами?
- Які методи адресації існують в мові асемблеру? Наведіть приклади?
- Назвіть основні групи команд арифметичних та логічних операцій.
- Які операції виконуються лише з бітами?

6. Загальні відомості

Система команд мікропроцесорів сімейства MCS-51 містить 111 базових команд, що реалізують 33 різноманітних функцій системи. Для опису команд використовуються 42 мнемонічних позначення (абревіатури).

Команди оперують з одно-, чотири-, вісьми- та 16-бітними словами.

Більшість команд мають формат у 1 - 2 байта і як правило виконуються за 1 - 2 машинних цикли.

По функціональній ознаці систему команд поділяють на 5 груп: **арифметичні операції, логічні операції, команди передачі даних, операцій із бітами, команди передачі керування**.

Структура команди мови асемблера звичайно включає мнемонічне позначення реалізованої функції, за яким слідують операнди, з посиланням на застосовані методи адресації та типи даних. При цьому мнемонічні позначення функцій не змінюються при зміні типів операндів.

Мова асемблера мікропроцесорів MCS-51 допускає наступні методи адресації: **безпосередню, пряму, регістрову, непряму регістрову, індексну, символічну**.

Безпосередня адресація дозволяє занести на адресу призначення константу, що безпосередньо вказана в команді, наприклад:

MOV A, #100; в акумулятор записується десяткове число 100.

Пряма адресація використовується для звертання до SFR-області та внутрішніх регістрів RAM. Можлива як бітова так і байтова адресація.

Допускається пряма байтова адресація до внутрішніх регістрів RAM з номерами 0 - 127, наприклад,

MOV A, 25H ; в акумулятор записується вміст регістра з адресою 25H.

Для бітової адресації регістрів внутрішньої RAM доступні регістри з номерами 20H - 2FH. Біти цих регістрів послідовно нумеровані та мають адреси 0 - 127. Сукупність цих біт складає бітову RAM - область мікропроцесора. В командах з бітовою прямою адресацією можливе використання як прямих адрес бітової так і байтової RAM-областей.

Наприклад, тотожними є наступні команди,

SETB 0; установлення нульового біта бітової RAM - області в "1"

SETB 20H.1; установлення нульового біта регістра 20H байтової RAM - області в "1", тому що в обох випадках адресується один і той же біт внутрішньої RAM-області.

Прямі адреси регістрів SFR-області мають номера 128 - 255, які не пересікаються з адресами регістрів RAM. Старший біт байта коду прямої адреси селектує RAM чи SFR області мікропроцесора.

Для бітової адресації придатні наступні регістри SFR: P0, P1, P2, P3, TCON, SCON, IE, IP, PSW, ACC, B.

Регістрова адресація використовується для звертання до обраного банку регістрів загального призначення R0 - R7, регістрів A, B, AB, DPTR, бітового акумулятора C. Її використання дозволяє отримати двобайтовий код команди, еквівалентний по результатам дії трибайтовому коду, що утворюється при використанні прямої адресації. Економія виникає за рахунок того, що в коді операції розміщено також адресу одного з операндів - регістра. Наприклад, передачу даних з регістру RAM за номером 44H в регістр 0H (або R0) можна реалізувати наступними способами:

MOV 0H, 44H; пряма адресація - запис команди займає 3 байти в ROM

MOV R0, 44H; регістрова адресація - запис команди займає 2 байти.

Такий метод адресації дозволяє більш економно використовувати пам`ять програм.

Непряма регістрова адресація використовується для звертання:

- до регістрів внутрішньої RAM з використанням регістрів покажчиків R0, R1 обраного банку регістрів загального призначення. В мікропроцесору всього 8 регістрів такого типу і тому їх необхідно використовувати досить обачливо. Регістр покажчик вказує на номер регістру RAM, що адресується, наприклад,

```
MOV R0, #2; в регістр покажчик R0 записана адреса регістру до якого
                ;буде записане число
MOV A, #10; запис числа в акумулятор
MOV @R0, A; передача числа з акумулятора на вказану адресу.
```

- до регістрів зовнішньої пам`яті даних. Можливе використання як байтового так і двобайтового покажчиків. Якщо використовується байтовий покажчик (регістри R0, R1), то обирається комірка зовнішньої пам`яті даних з сторінки - блоку в 256 байт. Номер сторінки попередньо вказується за допомогою порту P2, наприклад,

```
MOV P2, #3; в порт P2 записано номер сторінки звертання
MOV RO, #2; запис в покажчик адреси регістру зовнішньої пам`яті даних,
MOV A, #10; запис в акумулятор числа, що буде передаться
MOVX @R0, A; передача числа з акумулятора на вказану адресу.
```

Якщо використовується двобайтовий покажчик (регістр DPTR), то можлива адресація до будь якого регістру зовнішньої пам`яті даних ємності 64К. Програма звертання в цьому випадку має наступний вигляд:

```
MOV DPH, #0A9H; запис в регістр покажчик DPTR адреси комірки
MOV DPL, #04H; пам`яті #0A904H зовнішньої пам`яті даних
MOVX A,@DPTR; передача байта даних в акумулятор.
```

Індексна адресація, або непряма регістрова по сумі базового та індексного регістрів, використовується для читання даних з пам`яті програм. Дозволяє використовувати табличні дані, що записані в пам`яті програм. Адреса будь якої комірки пам`яті може бути встановлена за сумою базової адреси, що задається двобайтовими регістрами покажчиками DPTR або PC, та адреси зміщення, що задається акумулятором ACC:

MOVC A, @A+PC; передача в акумулятор з адреси @A+PC

MOVC A, @ A+DPTR; передача в акумулятор з адреси @ A+ DPTR.

Символічна адресація широко використовується під час написання програм для найменування як регістрів так і окремих біт SFR регістрів користувача. Переважна більшість крос-асемблерних програм допускає вживання зарезервованих найменувань регістрів і біт SFR. Для встановлення оригінальних імен користувачем, можливе використання директив асемблера:

TIME_1DATA34H; регістру RAM з номером 34H присвоєне ім`я TIME_1

ON BIT 20H.1; першому біту регістру 20H присвоєне ім`я ON.

В цьому випадку команди передачі даних мають наступний вигляд:

MOV TIME_1, A ; передача даних з акумулятора в регістр 34H

SETB ON ; установка першого біту регістра 20H в “1“.

Використання символічної адресації значно спрощує розробку програм та їх узгодження з мікроконтролерними системами на етапі конструювання.

Внаслідок виконання деяких з команд процесор може модифікувати прапорці - переносу (C), додаткового переносу (AC), переповнення (OV), парності (P). Перелік таких команд наведено нижче в таблиці 2.2.

Для опису команд використані наступні позначення:

s - операнд джерело даних; **dest** - операнд призначення; **bit** - пряма адреса біту; **rel** - відносна адреса переходу; **#data** - восьмибітові безпосередні дані; **#data16** - шістнадцятибітові безпосередні дані; **byte** - восьмибітовий операнд; **Ri** - індексні регістри, i = 1, 2; **Rn** - регістри загального призначення, n = 0 - 7; **ACC**

- акумулятор; **DPTR** - шістнадцятибітовий регістр покажчик; **PC** - шістнадцятибітовий програмний рахівник; SP - регістр покажчик стеку.

Таблиця 2.2

Команди	**Прапорці**			**Команди**	**Прапорці**		
	C	**OV**	**AC**		**C**	**OV**	**AC**
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C, /bit	X		
DIV	0	X		ORL C, bit	X		
DA	X			ORL C, /bit	X		
RRC	X			MOV C, /bit	X		
RLC	X			CJNE	X		
SETB C	1						

Команди передачі даних

Можлива передача одно-, чотири- та вісьмибітових даних. Всі команди цієї групи не модифікують прапори, за винятком команд завантаження регістрів PSW та ACC. Структура інформаційних зв`язків в мікропроцесорах MCS-51 має такий вигляд як на рисунку 2.1.

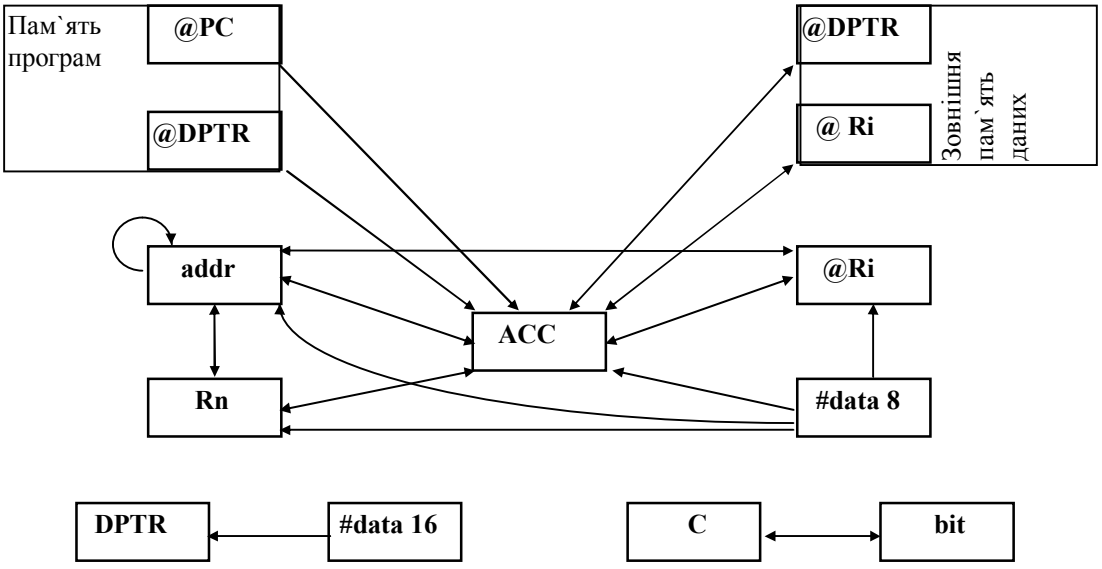


Рис. 2.1.

Звертання до акумулятора можливе як за неявної (A) так і прямої (ACC) адресації. Неявне звертання дозволяє використовувати вкорочені на 1 байт коди команд, але в цьому випадку неможливе звертання до окремих біт акумулятора.

В структурній схемі не зображені інформаційні зв`язки чотирибітового обміну. Існує дві операції, які проводяться з акумулятором, або акумулятором та будь яким регістром RAM:

SWAP A - обмін тетрадами в акумуляторі;

XCHD A, @Ri - обмін молодшої тетради акумулятора та будь якого регістру RAM, що адресується методом непрямої регістрової адресації.

Такі команди часто використовують для маніпуляцій з BCD - упакованими числами.

В таблицях наведено списки команд передачі даних мікропроцесорів сімейства MCS-51, з посиланням на можливість реалізації команд для прямої (1), непрямої регістрової (2), регістрової (3) та безпосередньої (4) адресації. Також наведено число циклів (N) мікропроцесора для реалізації цих команд.

Інструкції обміну даними з RAM та SFR областями процесора

Таблиця 2.3.

Команда	**Операція**	**Види адресації**				**N**
		1	2	3	4	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>, A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit				X	2
PUSH <src>	INC SP: MOV“@SP”,<src>	X				2
POP <dest>	MOV <dest>,“@SP”: DEC SP	X				2
XCH A,<byte>	A та <byte> обмін даними	X	X	X		1
XCHD A,@Ri	A та @Ri обмін молодшими тетрадами		X			1

Фрагмент програми з використанням команд обміну даними, які наведені у таблиці 2.3:

MOV A,#34 ; запис в акумулятор числа 34

MOV A, R2 ; запис в R2 вмісту акумулятора

MOV @R1, #2; запис в регістр РПД числа 2

XCH A, @R1; обмін інформації акумулятора з регістром РПД

PUSH PSW ; збереження вмісту регістру прапорів у стеку

Інструкції обміну даними з зовнішньою RAM

Таблиця 2.4.

Довжина адреси	**Команда**	**Операція**	**N**
8 біт	MOVX A, @Ri	Читання зовнішньої RAM @Ri	**2**
8 біт	MOVX @Ri, A	Запис у зовнішню RAM @Ri	**2**
16 біт	MOVX A,@DPTR	Зчитування зовнішньої RAM	**2**
16 біт	MOVX @DPTR, A	Запис у зовнішню RAM @ DPTR	**2**

Фрагменти програм з використанням команд приведених у таблиці:

MOV,#24H ;запис в регістр РПД числа 24

MOVX A, @R1 ; передача в акумулятор байта з РПД

Інструкції читання даних з пам`яті програм ROM

Таблиця 2.5.

Позначення команди	**Операція**	**N**
MOVC A,@A+DPTR	Читання за адресою @(A + DPTR)	**2**
MOVC A,@A+PC	Читання за адресою @(A + PC)	**2**

Фрагменти програм з використанням команд приведених у таблиці:

MOV DPTR, #32H ;завантаження вказівника данних

MOVC A, @A + DPTR; пересилка в акумулятор байта з ROM

Інструкції арифметичних операцій

В арифметико-логічному пристрої виконуються дії з цілими числами без знаку. В двоопернадних командах, акумулятор завжди використовується як перший операнд. В нього, після реалізації операції, записується результат дії (ADD, ADDC, SUBB). Операції додавання та віднімання з врахуванням знаку реалізувати з використанням прапора переповнення.

Якщо в програмі використовуються елементи BCD - арифметики (над двійково - десятковими упакованими числами), вживають прапорець додаткового переносу та команду десяткової корекції результату DA. Слід зауважити, що така команда не дозволяє перетворити формат чисел з шістнадцятирічного в BCD, а лише коригує результат операції додавання що проводиться з BCD числами. Арифметичні команди наведені в таблиці 2.6.

Таблиця 2.6.

Команда	**Операція**	**Види адресації**				**N**
		1	2	3	4	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Лише акумулятор				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Лише DPTR				2
DEC A	A = A – 1	Лише акумулятор				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	Лише ACC та B				4
DIV AB	A = Int[A/B] B = Mod[A/B]	Лише ACC та B				4
DA A	Десяткова корекція	Лише акумулятор				1

Розглянемо фрагменти програм з використанням арифметичних.

1. Операція додавання

MOV A,#34H ; запис в акумулятор числа 34

MOV R1,#2H; запис в регістр числа 2

ADD A, R1; додавання 34H + 2H. В акумулятор запис результату - 36H.

2. Операція віднімання

MOV A,#34H ; запис в акумулятор числа 34

MOV R1,#2H; запис в регістр R1 числа 2

SUBB A, R1; віднімання 34H – 2H - C. В акумулятоі результат - 32H.

3. Множення

MOV A,#34H ; запис в акумулятор числа 34

MOV B,#2H ; запис в регістр числа 2

MUL AB ; множення 2H*34H. В акумулятор запис результату: 68H.

4. Ділення

MOV A,#34 ; запис в акумулятор числа 34

MOV B,#2 ; запис в регістр числа 2

DIV AB ; ділення 34H + 2H. В акумулятор запис результату: 1A.

Інструкції логічних операцій

В перелік входить 25 команд, що реалізують логічні операції “І“, “АБО“, “НІ“, “ВИКЛЮЧАЮЧЕ АБО“ над вмістом акумулятора та іншим операндом. Результат операції в акумуляторі. Деякі з логічних операцій виконуються лише над акумулятором. Команди логічних операцій наведені в таблиці 2.7.

Таблиця 2.7.

Команда	**Операція**	**Види адресації**				**N**
		1	2	3	4	
ANL A,<byte>	A = A AND <byte>	X	X	X	X	1
ANL <byte>, A	<byte> = <byte> AND A	X				1
ANL <byte>,#data	<byte> = <byte> AND #data	X				2
ORL A,<byte>	A = A OR <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> OR A	X				1
ORL <byte>,#data	<byte> = <byte> OR #data	X				2
XRL A,<byte>	A = A.XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> XOR A	X				1
XRL <byte>,#data	<byte> = <byte> XOR #data	X				2
CRL A	A = 00H	Лише акумулятор				1
CPL A	A = NOT A	Лише акумулятор				1

Розглянемо фрагменти програм з використанням логічних команд.

1. Логічне „І”

MOV R1, #01010101B; запис числа в регістр

MOV A, #10101010B ; запис числа в акумулятор

ANL A, R1 ; логічне „і” акумулятора і регістра. Результат 00000000B.

2. Логічне „виключаюче або”

MOV R1, #01010101B ; запис числа в регістр

MOV A, #10101010B ; запис числа в акумулятор

XRL A, R1 ; логічне „ виключаюче або”. Результат 11111111B.

Інструкції операцій зсуву

В інструкціях зсуву вміст акумулятора циклічно зсувається вправо (RR A,) або вліво (RL A,) на 1 біт. В командах з використанням біту переносу (RRC A, RLC A), біт зсуву спочатку записується в прапорець переносу C, а його вміст завантажується в байт. Команди зсуву наведені в таблиці 2.8.

Таблиця 2.8.

Команда	**Операція**	**Види адресації**				**N**
		1	2	3	4	
RL A	Зсув ACC вліво на 1 біт	Лише акумулятор				1
RLC A	Зсув ACC вліво через біт переносу	Лише акумулятор				1
RR A	Зсув ACC вправо на 1 біт	Лише акумулятор				1
RRC A	Зсув ACC вправо через біт переносу	Лише акумулятор				1
SWAP A	Обмін тетрадами	Лише акумулятор				1

Розглянемо фрагменти програм з використанням команд зсуву:

MOV A, #00001111B ; запис числа в акумулятор

SWAP A ; обмін тетрадами в акумуляторі. Результ 11110000B.

RL A; зсув вліво. Результат 1110001.

Команди операцій з бітами

Мікропроцесори сімейства MCS-51 мають булевий (однобітний) процесор. Можлива адресація до 128 біт вбудованої RAM та такої ж кількості біт SFR- областей. Можлива незалежна адресація до всіх лінії портів і кожна з ліній може бути подана як окремий однобітний порт.

Повний перелік операцій з бітами включає основні інструкції байтового процесора: MOV, SET, CLR, CPL, OR, AND, JB, JNB, JBC.

Проте у всіх командах використовується лише пряма адресація. Біт переносу використовується як однобітний акумулятор булевого процесора. Бітові інструкції, що мають відношення до булевого акумулятора, мають власні коди. Разом з тим до “C - акумулятора“ можливий доступ як до окремого біта PSW - регістра (PSW.7). Недоліком системи бітових команд є відсутність серед логічних операцій команди “АБО ЩО ВИКЛЮЧАЄ“ (XRL).

Серед команд передачі керування є дві операції з бітовим акумулятором, та три операції з будь якими бітами. По цим командам досягається передача керування у межах -128+127 байт адресного простору. Перелік булевих команд наведено у таблиці 3.2.

Таблиця 3.2.

Команда	**Операція**	**N**	**Команда**	**Операція**	**N**
ANL C, bit	C = C ANDbit	2	SETB bit	bit = 1	1
ANL C /bit	C = C AND NOT bit	2	CPL C	C = .NOT.C	1
ORL C, bit	C = C OR bit	2	CPL bit	bit = .NOT.bit	1
ORL C,/bit	C = C OR NOT bit	2	JC rel	Перехід, якщо C = 1	2
MOV C,bit	C = bit	1	JNC rel	Перехід, якщо C= 0	2
MOV bit,C	bit = C	2	JB bit,rel	Перехід, якщо bit = 1	2
CLR C	C = 0	1	JNB bit,rel	Перехід, якщо bit = 0	2
CLR bit	bit = 0	1	JBC bit,rel	Перехід, якщо bit = 1; CLR bit	2
SETB C	C = 1	1			

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3

ДОСЛІДЖЕННЯ КОМАНД ЦИКЛУ ТА РОЗГАЛУЖЕННЯ У МІКРОПРОЦЕСОРАХ СІМЕЙСТВА MCS-51

1. Мета практикуму:

- Ознайомитися з основними бітовими операціями та операціями з масивами мікропроцесорів сімейства MCS-51.

2. Програма практикуму:

2.1. Скласти програми, що охоплюють основні арифметичні та логічні інструкції для роботи з бітовим представленням інформації, а також роботу з масивами.

2.2. В програмному середовищі μ-VISION/51 написати, асемблювати і відлагодити програму відповідних дій, завдання яких приведені в таблиці 3.1;

2.3. З використанням програмного симулятора dScope-51 крок за кроком виконати програму (пункт 2.2.). Прослідкувати за зміною даних в відповідних регістрах SFR.

3. Завдання до практикуму

В таблиці 3.1 приведені вихідні дані щодо виконання практикуму.

4. Зміст звіту:

- Титульний лист з відомостями про назву виконаної практикуму і склад бригади;
- Текст програми з коментарями.

5. Контрольні запитання:

- Визначити класифікацію команд управління мікропроцесорів MCS-51.
- Які типи команд розгалуження програм існують, як працюють?
- Як реалізується цикл в програмі?

Таблиця 3.1.

№	Завдання
1	На третій порт вивести тільки позитивні числа. Довжина масиву 100 слів.
2	Вивести через перший порт числа, які діляться на 4. Довжина масиву 56H.
3	Вивести через другий порт числа, що не перевищують 16 бітів. Довжина масиву 100 слів.
4	Додати два масиву чисел. При отриманні переповнення, вивести на нульовий вивід першого порту одиницю. Довжина масиву 120 слів.
5	Поділити масив результату на байти масиву, що знаходиться в ОЗП починаючи з адреси 20H. Остаточний результат вивести через перший порт.
6	Вивести через другий порт числа, у яких біти D4 та D6 дорівнюють 1, а біти D2 та D7 дорівнюють 0. Довжина масиву 220 байт
7	Відняти два масиву чисел. Результати, що більше 50H вивести через третій порт. Довжина масиву 150 чисел.
8	Відсортуйте масив, на потрапляння чисел в діапазон 0…50H. Числа, які потрапили в діапазон вивести на третій порт. Довжина масиву 30H.
9	Вивести через другий порт ті результати, у яких два старших біти всіх байтів дорівнюють 1. Довжина масиву 20 байт
10	У випадку негативного результату виведіть на нульовий порт число AAH, в протилежному випадку – подвоєне число масиву. Довжина масиву 53d
11	Виведіть на другий порт числа з масиву результату, які більші, ніж число 200. Довжина масиву 40H.
12	Відсортуйте масив на позитивні та негативні числа. Довжина масиву 100 байт. Позитивні числа виводити на нульовий порт, а негативні на другий.
13	З масиву виберіть числа, у яких біти D0 та D5 дорівнюють 1, а біти D2 та D4 – 0. Остаточний результат вивести на третій порт.
14	Порівняйте масив результату з масивом, що записано в ОЗП з адреси 10H. У разі рівності на перший вивід першого порту подати 1. Довжина масиву 23h.
15	Виведіть на третій порт числа з масиву, які знаходяться в діапазоні 50…100. Довжина масиву 25H.
16	З масиву довжиною 30H виведіть на перший порт ті числа, третій біт яких дорівнює одиниці.
17	Порівняйте два масиви, довжиною 10H, якщо відповідні числа з першого масиву більші ніж другого, виведіть результат на другий порт.
18	Виведіть на другий порт числа з масиву перші біти яких дорівнюють нулю. Довжина масиву 50H
19	Порівняти два масиви, довжиною 15H, на рівність третіх бітів відповідних чисел масиву, у разі виконання умови, вивести на 3 вивід другого порту 1.
20	З масиву довжиною 45H виберіть числа з діапазону (30...90). Результати виведіть на другий порт.

6. Загальні відомості

Команди розгалуження та передачі керування

Існує два типи інструкції - команди **безумовних** та **умовних** переходів.

Серед команд **безумовного переходу** вирізняють, в залежності від довжини адреси розгалуження три типи команд:

- короткого переходу, по відносному збільшенню адреси в межах -128 +127 байт адресного простору, відносно наступної команди (SJMP). Команда займає 1 байт;

- довгого переходу на 16-бітову адресу (LJMP). Команда займає 3 байти.

- абсолютного переходу у межах однієї сторінки пам`яті програм (AJMP).

В таких командах використовується 11-бітова адреса. Існує 8 типів такої команди, які мають різні коди, в залежності від розміщення сторінки що адресується. Мнемоніка такої команди в програмах залишається незмінною, кросасемблерна програма автоматично визначає тип команди та присвоює їй певний код. Команди такого типу займають 2 байти;

Команди безумовного переходу звичайно використовують пряму адресацію. Вказується ім`я мітки або 16 - бітова адреса. Однак існує команда, що дозволяє реалізувати перехід на адресу, що вказана методом непрямої регістрової адресації - JMP @A+DPTR. Така команда використовується для організації параметричного розгалуження по багатьом напрямам:

```
                MOV DPTR,#TAB    ; запис адреси початку таблиці розгалужень
                MOV A, #DELTA    ; запис зміщення на потрібне розгалуження
                JMP  @A+DPTR     ; перехід в точку розгалуження
; таблиця розгалужень
TAB:    SJMP VAR_0               ; варіанти розгалужень
                SJMP VAR_5       ; варіанти розгалужень.
```

Досить часто такий спосіб використовується для написання багатопараметричних драйверів пристроїв.

Для повернення з підпрограм використовують команду RET, яка поновлює вміст програмного рахівника PC. Для повернення з підпрограм переривання використовують команду RETI, яка на відміну від попередньої команди поновлює також логіку переривань процесора. Перелік команд безумовних переходів розміщено в таблиці 3.3.

Таблиця 3.3.

Позначення команди	**Операція**	**N**
JMP addr	Перехід на адресу addr	**2**
JMP @A+DPTR	Перехід на адресу A + DPTR	**2**
CALL addr	Виконання підпрограми за адресою addr	**2**
RET	Повернення з підпрограми	**2**
RETI	Повернення з підпрограми переривань	**2**
NOP	Немає операції	**1**

Всі команди **умовних переходів** дозволяють реалізувати розгалуження в межах відносної адреси зміщення -128+127 байт адресного простору.

Такі команди, за винятком CJNE та JBC не впливають на прапори. Команда CJNE встановлює прапор C, якщо перший операнд менший за другий; команда JBC скидає прапор C після переходу, в разі використання цього біту, як операнда.

Команда DJNZ поєднує арифметичну операцію DEC з операцією умовного переходу JNZ. Ця команда часто використовується для контролю за кількістю циклів виконання процедур в програмі.

Список команд умовних переходів наведено в таблиці 3.4.

Таблиця 3.4.

Позначення команди	**Операція**	**Види адресації**				**N**
		1	2	3	4	
JZ rel	Перехід, якщо A = 0	Акумулятор				2
JNZ rel	Перехід, якщо A ≠ 0	Акумулятор				2
DJNZ <byte>, rel	Зменшення та перехід, якщо не нуль	X		X		2
CJNE A, <byte>, rel	Перехід, якщо A ≠ <byte>	X			X	2
CJNE <byte>, #data, rel	Перехід, якщо <byte> ≠ #data		X	X		2

Приклад 1. Інкрементувати склад комірок пам’яті з адресами 10 – 18:

```
INCR:   MOV R0, #10; завантаження в R0 початкову адресу
        MOV R3, #8; завантаження в R3 числа комірок, які інкрементуються
LOOP:   INC @R0; інкремент комірок внутрішньої ОЗП
        INC R0; пресування вказівника адреси
        DJNZ R3, LOOP; декремент R3 і повтор доки R3 не дорівнює нулю.
```

Приклад 2. Помножити акумулятор на число 2 в степені X, яке зберігається в R2. Множення на 2 заміниться арифметичним зсувом вліво акумулятора та розширювача R1:

```
        MOV R2, #8H; визначення числа X
        CLR C; скидання прапорця переносу
LOOP:   RLC A; зсув вліво
        XCH A, R1; 16 бітного результату
        RLC A; в регістровій парі R1 та A
        XCH A, R1
        DJNZ R2, LOOP; організація циклу.
```

Приклад 3. Скласти два двійкові багатобайтні числа. Початкові адреси масивів задані в R0, R1. Довжина масиву задана в R2.

```
        MOV R0,30H    ;R0 – початкова адреса массиву X
        MOV R1,40H    ;R1 - початкова адреса массиву У
        MOV R2,#8     ;введення інформації про кількість циклів
        CLR C         ;очистка прапорця переносу
M1:     MOV A, @R0    ;завантаження в акумулятор байта з першого масиву
        ADDC A, @R1   ;додавання з урахуванням переносу
        MOV @R0, A    ;розміщення результату
        INC R0        ;інкремент регістра (адреси) массиву X
        INC R1        ;інкремент регістра (адреси) массиву У
        DJNZ R2, M1   ;організація цикла, поки R2 не буде нульовим.
```

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

РОБОТА ТАЙМЕРІВ-ЛІЧИЛЬНИКІВ

1. Мета практикуму:

- Ознайомитись з принципами роботи таймерів-лічильників мікропроцесора MCS51;
- Освоїти алгоритми затримки цифрової інформації з використанням таймерів-лічильників.

2. Програма практикуму:

2.1. Вивчити особливості роботи таймерів-лічильників;

2.2. В програмному середовищі μ-VISION/51 написати, асемблювати і відлагодити програму затримки інформації на час $T_з$, за допомогою таймера-лічильника та без його використання при частоті генератора f_{osc} = 12 МГц.

2.3. З використанням програмного симулятора виконати програму (пункт 2.2.). Прослідкувати за зміною даних в відповідних регістрах SFR.

3. Завдання до практикуму

В таблиці 4.1 приведені вихідні дані для виконання практикуму.

Таблиця 4.1.

№	Таймер-лічильник	Режим	$T_з$, мкс	№	Таймер-лічильник	Режим	$T_з$, мкс
1	0	0	160	13	1	3	155
2	1	1	150	14	0	0	145
3	0	2	140	15	1	1	135
4	1	3	130	16	0	2	125
5	0	0	120	17	1	3	115
6	1	1	110	18	0	0	105
7	0	2	100	19	1	1	95
8	1	3	90	20	0	2	85
9	0	0	80	21	1	3	75
10	1	1	70	22	0	0	65
11	0	2	60	23	1	1	55
12	1	3	50	24	0	2	45

4. Зміст звіту:

- Титульний лист з відомостями про назву виконаної практикуму і склад бригади;
- Текст програми з коментарями.

5. Контрольні запитання:

- Назвіть основні режими роботи таймерів-лічильників і коротко опишіть їх?
- Які регістри мікроконтролера відповідають за роботу таймерів/лічильників?
- Яке призначення бітів регістрів TMOD та TCON?

6. Теоретичні відомості

У архітектурі MCS-51 два 16-розрядних таймера/лічильника 0 і 1. Кожен із них незалежно може бути запрограмований на роботу у якості таймера (відрахунок часу через підрахунок внутрішніх імпульсів синхронізації) або лічильника (підрахунок подій на зовнішньому вході). В обох випадках перехід через заздалегідь установлений рубіж приводить до формування запиту переривань.

Спрощена структурна схема нульового таймера/лічильника наведена на рис 4.1. До нього входять два регістра TL0 (молодший) і TH0 (старший), переддільник тактової частоти на 12 (Д), комутатор (К), який підключає вхід таймеру/лічильника або до виходу переддільника, або до входу мікроконтролера T0 (вхід P3.4), вимикач S, який керується вузлом на трьох логічних елементах, прапорець переповнення лічильника TF0. Структурна схема першого таймера/лічильника аналогічна, тільки при роботі в режимі лічильника він підключається до входу мікроконтролера T1 (вхід P3.5).

Регістри TH0 і TL0 (TH1, TL1) це регістри самого таймера/лічильника. В них завантажується число, яке установлює таймер/лічильник і саме з них зчитується його поточне значення.

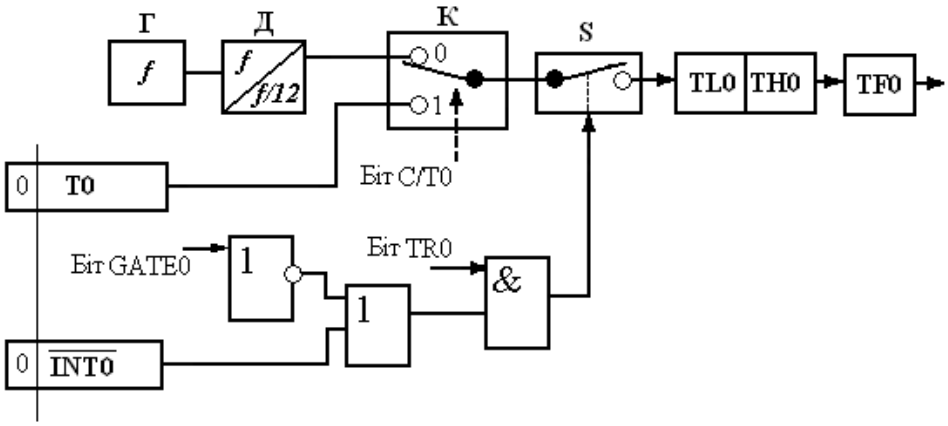


Рис.4.1

Управління ж таймерами/лічильниками відбувається за допомогою бітів, що входять до складу регістру режиму TMOD (табл. 4.2) і чотирьох старших бітів регістру управління/статусу TCON (табл. 4.3).

Таблиця 4.2.

Біт	**№ біту**	**Функція**
GATE1	TMOD.7	Управління блокіровкою таймера 1. GATE1 = 1 таймер працює поки на $\overline{INT}1$ = 1 і TR1 = 1. GATE1 = 0, то запуститься при TR1 = 1.
C/T1#	TMOD.6	Біт вибору типу роботи для таймера 1. C/$\overline{T}$ 1 = 1 працює як лічильник, при C/$\overline{T}$ 1 = 0 як таймер.
M1.1	TMOD.5	визначення режиму роботи таймера/лічильника 1
M1.0	TMOD.4	визначення режиму роботи таймера/лічильника 1
GATE0	TMOD.3	Біт управління таймером 0. GATE0 = 1 таймер працює поки на $\overline{INT}0$= 1 і TR0 = 1. GATE0 = 0, то запуститься при TR0 = 1.
C/$\overline{T}$ 0	TMOD.2	Біт вибору типу подій для таймера 0. C/$\overline{T}$ 0 = 1 працює як лічильник, при C/$\overline{T}$ 0 = 0 як таймер.
M0.1	TMOD.1	визначення режиму роботи таймера/лічильника 0.
M0.0	TMOD.0	Біт 0 визначення режиму роботи таймера/лічильника 0.

Як видно, управління запуском/зупинкою таймерів/лічильників залежить від стану бітів GATE, TRx і рівнем сигналу на вході $\overline{INT}$ x (P3.2. – це $\overline{INT}$ 0, P3.3. – це $\overline{INT}$ 1). При GATE = 0 запуск/зупинка відбувається тільки встановленням/скиданням біта TRx, незалежно від стану на вході $\overline{INT}$ x. Такий режим

використовується при формуванні затримки або часових інтервалів заданої тривалості. При GATE = 1 запуск/зупинка відбувається також встановленням/скиданням біта TRx. Тільки при цьому таймер/лічильник буде зчитувати вхідні сигнали тільки в тому випадку, коли на його вході $\overline{INT}$x буде одиничний сигнал. Такий режим використовується при вимірюванні тривалості імпульсу.

Таблиця 4.3.

Біт	**№ біту**	**Функція**
TF1	TCON.7	Прапорець переповнення таймера 1. Встановлюється при переповненні. Очищається при обслуговуванні переривання.
TR1	TCON.6	Біт запуску таймера 1. При TR1 = 1 рахунок дозволено.
TF0	TCON.5	Прапорець переповнення таймера 0. Встановлюється при переповненні. Скидається при обслуговуванні переривання.
TR0	TCON.4	Біт запуску таймера 0. При TR0 = 1 рахунок дозволено

Таймери можуть працювати у чотирьох режимах, вибір режиму для кожного таймера здійснюється комбінацією бітів M1, M0 регістру TMOD (табл. 4.4). Для обох таймерів/лічильників режими роботи 0, 1 і 2 однакові.

Таблиця 4.4.

M 1	**M0**	**Режим роботи**
0	0	**Режим 0**. THx - 8- розрядний таймер/лічильник. TLx як 5- розрядний переддільник.
0	1	**Режим 1**. 16- розрядний таймер/лічильник. THx і TLx включені послідовно.
1	0	**Режим 2**. 8- розрядний таймер/лічильник TLx з автоперезагрузкою значенням із THx.
1	1	**Режим 3**. TL0 - 8-розрядний таймер/лічильник, що управляється бітами управління таймера 0. TH0 - 8-розрядний таймер/лічильник, що управляється бітами управління таймера 1. Таймер 1 не працює.

Режим 0. В цьому режимі таймер/лічильник має розрядність 13-біт. Він складається для $\overline{T}$/C0 з восьми розрядів TH0 та п'яти розрядів TL0, а для $\overline{T}$/C1 - з восьми розрядів TH1 і п'яти розрядів для TL1. Молодша частина $\overline{T}$/C TL1 - ді-

льник частоти на 32, а старша частина TH1 - восьмирозрядний таймер або лічильник в залежності від значення біта C/T1. При C/T1 = 0 $\overline{T}$/C працює в режимі таймера, тобто на вхід $\overline{T}$/C приходять імпульси $f_{osc}/12$, де f_{osc} – частота внутрішнього або зовнішнього генератора OSC.

Режим 1. Цей режим аналогічний режиму 0, але використовуються всі 16 розрядів регістрів THx і TLx. Він працює наступним чином. Кожний імпульс, що надходить на вхід $\overline{T}$/Cx, збільшує на 1 вміст регістра TLx. Коли відбувається переповнення (тобто вміст TLx змінюється з 00FFH на 0) відбувається збільшення на 1 вмісту THx. Звичайно, збільшення THx і TLx можливе лише тоді, коли TRx встановлений в 1 і $\overline{INT}$x = 1 при GATE = 1. Коли відбувається переповнення THx, то встановлюється в 1 відповідний прапорець переповнення таймеру TFx.

Режим 2. В режимі 2 таймер/лічильник працює як восьмирозрядний $\overline{T}$/C з автоперезавантаженням, при чому, TLx – це восьмирозрядний регістр $\overline{T}$/C, а THx використовується в якості регістра збереження початкового коду - коду автоперезавантеження. Кожний із імпульсів, що поступають на вхід $\overline{T}$/Cx, збільшує на 1 вміст регістру TLx. Коли відбувається переповнення (вміст TLx змінюється із з 0FFH на 0), то встановлюється прапорець переповнення TFx. Одночасно з цим відбувається занесення в TLx числа, яке зберігається в регістрі THx, при цьому вміст THx залишається незмінним (рис.4.2).

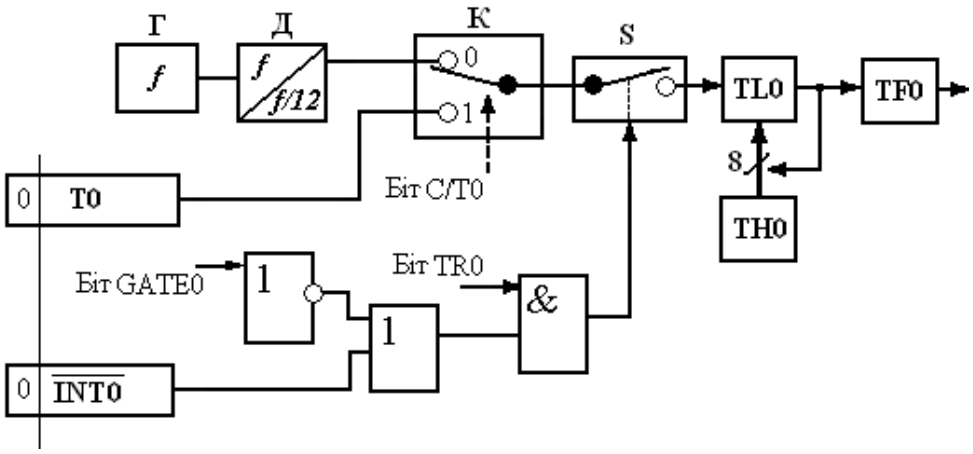


Рис. 4.2.

Режим 3. Таймер 1 у цьому режимі блокований, так, ніби біт TR1 був скинутий. Таймер 0 працює як два незалежних лічильних регістра, причому регістр TL0 управляється бітами управління таймера 0, а регістр TH0 управляється бітами управління таймера 1.

Приклади організації затримки інформації

Приклад 1. Організувати затримку інформації на 100 мкс, за допомогою 2-го режиму таймера/лічильника і вивести її на адресу 60H.

```
; Опис констант та змінних
INI_TMOD EQU 00000100B ;константа завантаження регістра TMOD
INI_TC0 EQU (0FF -100)       ; константа затримки
INI_P3 EQU 11111111B         ;перехід в режим альтернативних функцій
ORG 0H                        ;адреса рестарта після пуску процесора
SJMP W1                       ;перехід на початок основної програми
ORG 20H                       ;початкова адреса основної програми
W1:   MOV A, 0FFH             ;завантаження в акумулятор числа
MOV TMOD,#INI_TMOD     ;ініціалізація таймера T/C0 - 2 режим
MOV TH0, #INI_TC0             ;завантаження рахункового регістра таймера
MOV TL0,TH0                ;завантаження регістра перевантаження таймера
SETB TR0                      ;ввімкнення таймера/лічильника
MOV 60H, A                    ;передача даних на відповідну адресу
END
```

Приклад 2. Організувати затримку на 40 мкс без таймера/лічильника.

```
MOV A, #19 ; виконання команди - 1 мкс, при частоті МК 12 МГц
L1: DJNZ A, L1 ;однократне виконання - 2 мкс, весь цикл - 19 х 2 = 38 мкс.
NOP ; додатково ще 1 мкс
END
```

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5

СИСТЕМА ПЕРЕРИВАНЬ МІКРОКОНТРОЛЕРІВ x51

1. Мета практикуму:

- Ознайомитись з принципами роботи системи переривань мікроконтролера MCS51;
- Освоїти алгоритми формування міандру за допомогою системи переривань

2. Програма практикуму:

2.1. Вивчити особливості роботи переривань від різних джерел;

2.2. В програмному середовищі μ-VISION/51 написати, асемблювати і відлагодити програму формування імпульсної послідовності з періодом T, тривалістю τ на виході заданого виводу порту за допомогою системи переривань, якщо частота генератора f_{osc} = 12 МГц.

2.3. З використанням програмного симулятора виконати програму (пункт 2.2.). Прослідкувати за зміною даних в відповідних регістрах SFR.

3. Завдання до практикуму

В таблиці 5.1 наведені вихідні дані для виконання практикуму.

Таблиця 5.1.

№	Вивід порту	τ, мкс	T,мкс	№	Вивід порту	τ, мкс	T,мкс
1	P0.7	10	20	11	P2.1	60	120
2	P1.6	15	30	12	P3.2	65	130
3	P2.5	20	40	13	P0.3	70	140
4	P3.4	25	50	14	P1.4	75	150
5	P0.3	30	60	15	P3.6	10	20
6	P1.2	35	70	16	P0.7	15	30
7	P2.1	40	80	17	P1.7	20	40
8	P3.0	45	90	18	P2.6	25	50
9	P0.7	50	100	19	P3.5	30	60
10	P1.0	55	110	20	P0.4	35	70

4.Зміст звіту:

- Титульний лист з відомостями про назву виконаної практикуму і склад бригади;
- Текст програми з коментарями.

5. Контрольні запитання:

- Визначити класифікацію системи переривань.
- В якому порядку здійснюється вибір пріоритету переривань?
- Яке призначення бітів в регістрах IE, IP?
- Які біти регістри регістру TCON відповідають за роботу переривань?

6. Теоретичні відомості

Переривання - це такий внутрішній механізм, який дозволяє мікроконтролеру реагувати на ті чи інші зовнішні, по підношенню до програми, дії. Реакція мікроконтролера полягає в тому, що викликається відповідна підпрограма (за умови, що всі разом і кожне окреме переривання дозволені). Підпрограми, що викликаються, починатися з строго визначених адрес.

Мікроконтролер МК51 має 5 переривань: від нульового таймера, від першого таймера, від сигналу нульового логічного рівня (або від перепаду із 1 в 0) на $\overline{INT}0$ (вихід P3.2), від сигналу нульового логічного рівня (або від перепаду із 1 в 0) на $\overline{INT}1$ (вихід P3.3), від послідовного прийомопередавача (UART).

Джерело переривання (переповнення в таймері, нульовий сигнал на вході $\overline{INT}$, прийнявший байт прийомопередавач) викликає свою підпрограму обробки шляхом встановлення в 1 відповідного біта (табл 5.1).

$\overline{INT}0$, $\overline{INT}1$ може бути активізоване по рівню ("0") або по фронту (перехід з "1" в "0") сигналів, що визначається станом бітів IT0 і IT1 регістру TCON (табл. 5.2). При появі запиту зовнішнього переривання $\overline{INT}$x (x = 0,1) встановлюється прапорець IEx (x = 0,1) регістру TCON (табл. 5.2) і це викликає відповідні переривання.

Таблиця 5.1.

Джерело переривань	Прапорець	Адреса підпрограми (вектор переривання)
Сигнал на $\overline{INT}0$	IE0 (TCON.1)	0003 H
Таймер/лічильник 0	TF0 (TCON.7)	000B H
Сигнал на $\overline{INT}1$	IE1 (TCON.3)	0013 H
Таймер/лічильник 1	TF1 (TCON.5)	001B H
UART	TI, RI (SCON)	0023 H

Таблиця 5.2.

Біт	**№ біту**	**Функція**
IE1	TCON.3	Прапорець запиту переривань по входу $\overline{INT}1$.
IT1	TCON.2	Біт селектора типу активного сигналу на вході $\overline{INT}1$. При IT1 = 1 активним є перехід „1” - „0”, при IT1 = 0 активним є низький рівень сигналу.
IE0	TCON.1	Прапорець запиту переривань по входу $\overline{INT}0$.
IT0	TCON.0	Біт селектора типу активного сигналу на вході $\overline{INT}0$. При IT0 = 1 активним є перехід „1” - „0”, при IT0 = 0 активним є низький рівень сигналу.

Очищення прапорця IEx виконується при переході до підпрограми обробки переривання апаратно (автоматично) тільки, якщо переривання було викликано переходом сигналу із 1 в 0. При перериванні по нульовому рівню прапорець очищується при знятті запиту зовнішнього переривання, тобто коли на вході $\overline{INT}$ x встановиться 1.

Переривання від таймерів/лічільників викликаються встановленням прапорців TF0 и TF1 регістру TCON, які встановлюються при переповненні відповідних регістрів таймерів/лічільників (за виключенням режиму 3). Очистка прапорців TF0 и TF1 виконується внутрішньою апаратурою ОМЕОМ при переході до підпрограми обслуговування переривання.

Переривання від послідовного порту викликається встановленням прапорця переривання приймача RI або прапорця переривання передавача TI в регістрі SCON. На відміну від усіх інших прапорців, RI и TI очищаються тільки про-

грамним шляхом зазвичай в межах підпрограми обробки переривання, де визначається, якому з прапорців RI або TI відповідає переривання.

Кожне з перерахованих джерел переривань може бути індивідуально дозволене чи заборонено встановленням або очищенням відповідного біту в регістрі дозволу переривань IE (табл.5.3). Регістр IE вміщує також біт EA, встановлення якого в "0" забороняє відразу усі переривання. Необхідною умовою переривання є його дозвіл у регістрі IE.

Таблиця 5.3.

Біт	**№ біту**	**Функція**
EA	IE.7	Загальна блокіровка переривань. Встановлення/скидання відбувається програмно. При EA = 0 всі переривання заборонені.
	IE.6	Не використовується
	IE.5	Не використовується
ES	IE.4	Біт дозволу/заборони переривань від прийомопередавача
ET1	IE.3	Біт дозволу/заборони переривань від таймера/лічильника 1
EX1	IE.2	Біт дозволу/заборони переривань від входу $\overline{INT}$ 1
ET0	IE.1	Біт дозволу/заборони переривань від таймера/лічильника 0
EX0	IE.0	Біт дозволу/заборони переривань від входу $\overline{INT}$ 0

Усі біти, що викликають переривання (IE0, IE1, TF0, TF1, RI, TI), можуть бути програмно встановлені або скинуті з тим самим результатом, що і у випадку їх апаратного встановлення або скидання. Тобто переривання можуть програмно викликатися або переривання, які очікують обслуговування можуть програмно ліквідуватись. Окрім того, переривання по $\overline{INT}$ 0, $\overline{INT}$ 1 можуть викликатись програмним встановленням P3.2 = 0 та P3.3 = 0, як показано в наведеному нижче прикладі:

```
MAIN:   MOV IE, # 00000101B ; дозвіл переривання від INT 0, INT 1.
        MOV IP, # 04H       ; присвоєння INT 1 старшого пріоритету.
        SETB EA             ; загальний дозвіл переривань.
        MOV P3; # 11110011B ; імітація зовнішніх переривань.
```

SUBR: ORG 013H ; перехід до підпрограми обслуговування $\overline{INT}$ 1.

У наданому прикладі запити переривання $\overline{INT}$ 0 і $\overline{INT}$ 1 мають різний пріоритет, тобто різну ступінь важливості і відповідно різну послідовність реакції на них з боку мікроконтролера.

Структура пріоритетів переривань є дворівневою. Кожному джерелу переривання може бути індивідуально привласнено один з двох рівнів пріоритету: високий або низький. Виконується це встановленням (високий рівень пріоритету) чи скиданням (низький рівень пріоритету) відповідного біта в регістрі пріоритетів переривань IP (табл. 5.4).

Таблиця 5.4.

Біт	**№ біту**	**Функція**
EA	IP.7	Не використовується
	IP.6	Не використовується
	IP.5	Не використовується
PS	IP.4	Біт пріоритету прийомопередавача
PT1	IP.3	Біт пріоритету таймера/лічильника 1
PX1	IP.2	Біт пріоритету зовнішнього переривання 1
PT0	IP.1	Біт пріоритету таймера/лічильника 0
PX0	IP.0	Біт пріоритету зовнішнього переривання 0

Програма обробки переривань з низьким рівнем пріоритету може бути перервана запитом переривання з високим рівнем пріоритету, але не може бути перервана іншим запитом переривання з низьким рівнем пріоритету. Програма обробки переривань з високим рівнем пріоритету не може бути перервана ніяким іншим запитом переривання ні з якого джерела. Якщо два запита з різними рівнями пріоритету прийняті одночасно, спочатку буде обслуговано запит з високим рівнем пріоритету. Якщо одночасно прийняті запити з однаковим рівнем пріоритету, обробка їх буде виконуватись в черзі, яка задається послідовністю внутрішнього опитування прапорців переривань.

Таким чином, в межах одного пріоритетного рівня існує ще одна структура пріоритетів:

Джерело	Пріоритет всередині рівня
1. IE0	(найвищій)
2. TF0	
3. IE1	
4. TF1	
5. RI + TI	(найнижчий)

Необхідно особливо підкреслити, що структура "Пріоритет всередині рівня" працює тільки у тих випадках, коли визначається послідовність обслуговування запитів на переривання, які прийняті одночасно і при цьому мають однаковий рівень пріоритету.

Підпрограма обслуговування переривання продовжується до виконання команди RETI. Команда RETI відновлює стан логіки переривання та завантажує в лічильник команд PC 2 байта адреси повернення з двох верхніх комірок стека. Відновлення стану логіки переривання полягає у наступному: при переході по вектору на підпрограму обробки переривання автоматично до виконання команди RETI незалежно від стану біт регістру IE забороняються всі переривання з рівнем пріоритету, що дорівнює рівню пріоритету обслуговування переривання, тобто вкладені переривання з рівними рівнями пріоритету неможливі. Команда RETI знімає цю заборону.

Приклад роботи з системою переривань

Сформувати імпульсну послідовність з періодом 200мкс, тривалістю 100мкс на виході 1-го виводу другого порту за допомогою системи переривань, якщо частота генератора f_{osc} = 12 МГц

```
; Опис констант та змінних
INI_TMOD EQU 00000010B     ;константа завантаження регістра TMOD
INI_TC0 EQU (0FFH-100)     ;тривалість імпульсу
```

```
        INI_TC1 EQU (0FFH-200)        ;генерація періоду слідування імпульсів
        ME BIT P2.1                   ; лінія порту формування імпульсу
        ; Програма
ORG 0H                        ;адреса вектора рестарту після пуску процесора
            SJMP BEGIN        ;перехід на мітку BEGIN
ORG 0BH                       ;адреса переривань по 0-му таймеру
            SJMP INTER_T0 ;перехід на мітку переривань від таймера
        ORG 1BH               ;адреса переривань по 1-му таймеру
            SJMP INTER_T1 ;перехід на мітку переривань
ORG 20H                       ;початкова адреса блоку ініціалізації
BEGIN:  CLR ME                ;скидання вихідного сигналу
           MOV TMOD,#INI_TMOD;налагодження режиму таймера
            MOV TH0, #INI_TC0  ;завантаження регістрів 0 таймера
             MOV TL0,TH0
             MOV TH1, #INI_TC1 ;завантаження регістрів 1 таймера
             MOV TL1,TH1
             SETB TR0              ;ввімкнення таймера/лічильника
             SETB ET0              ;дозвіл переривання по 0-му Т/Сч
             SETB ET1              ;дозвіл переривання по 1-му Т/Сч
             SETB EA               ;загальний дозвіл на переривання
MAIN:       SJMP MAIN            ;очікування сигналу переривання по 0-му таймеру
INTER_T0:  SETB ME             ;встановлення в ″1″ вихідний імпульс
                SETB TR1          ;ввімкнення Т/Сч1- генератора періоду
                RETI              ;повернення із переривання по 1-му таймеру
INTER_T1: CLR ME              ;скидання в 0 стану вихідного імпульсу
               CLR TR1            ;вимкнення генератора періоду
               RETI
END
```

ЛІТЕРАТУРА

1. Микропроцессоры и микропроцессорные комплекты интегральных микросхем. Справочник. В 2-х т./В.-Б.Б.Абрайтис и др.-М.Радио и связь,1988

2. СверхБИС универсальных однокристальных микро-ЭВМ/ А.В.Кобылинский. - К: Техника,1986

3. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.: Энергоатомиздат,1990

4. Липовецкий Г.П. и др. Однокристальные микроЭВМ. Семейство МК48, Семейство МК51. Техническое описание и руководство по применению. -М.: МП "Бином", 1992

5. Лебедев О.Н. Микросхемы памяти и их применение, М., Радио и связь,1990

6. Гутников В.С. Интегральная электроника в измерительных устройствах.

7. Фрунзе А.В. Микроконтроллеры? Это же просто! ; - М.; ООО «ИД СКИМЕН», 2002.

8. Каспер Эрни. Программирование на языке Ассемблера для микроконтроллеров семейства i8051. – М.: Горячая линия – Телеком, 2003.