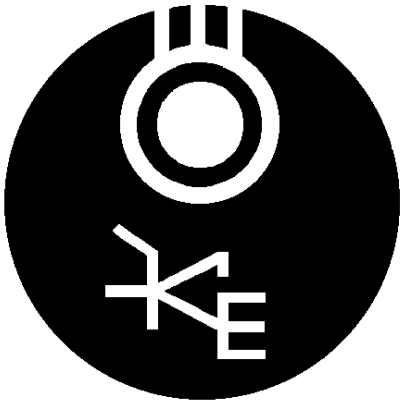
Міністерство освіти і науки України
Національний гірничий університет

П.І.Б. _____
Група _____
Варіант _____
Відмітка про залік:
_____



МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи МПП-2
“СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА 8051АН СІМЕЙСТВА MCS-51. ТИПОВІ ПРОГРАМНІ КОНСТРУКЦІЇ”,
індивідуальних завдань та самостійної роботи
з професійно-орієнтованої дисципліни
“Мікропроцесорні пристрої”

для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму “Електромеханіка”

Дніпропетровськ
2006

Міністерство освіти і науки України
Національний гірничий університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи МПП-2 “Система команд мікроконтролера 8051АН сімейства MCS-51. Типові програмні конструкції”, індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни “Мікропроцесорні пристрої” для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму "Електромеханіка”

Дніпропетровськ, НГУ
2006

Методичні вказівки до виконання лабораторної роботи МПП-2 “Система команд мікроконтролера 8051АН сімейства MCS-51. Типові програмні конструкції”, індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни “Мікропроцесорні пристрої” для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму "Електромеханіка” / Упорядн.: В. І. Кириченко, О. А. Яланський, К. В. Ковшов. – Дніпропетровськ: Національний гірничий університет, 2006. – 56 с.

Упорядники: В.І. Кириченко, д–р техн. наук, проф.
О.А. Яланський, канд. техн. наук, доц.
К.В. Ковшов, магістр

Відповідальний за випуск завідувач кафедри електропривода
О.С. Бешта, д–р техн. наук, проф.

ЗМІСТ

ВСТУП

Мета самостійної та лабораторної робіт – поглиблення знань з дисципліни “Мікропроцесорні пристрої” шляхом теоретичного вивчення та отримання навичок практичного використання системи команд мікроконтролера 8051АН сімейства MCS-51 при програмуванні типових програмних конструкцій. Зважаючи на простоту користування та наочність отриманих результатів програмування у роботі запропоновано користуватися налагоджувачем-симулятором FD51.

1. ЗМІСТ САМОСТІЙНОЇ ТА ЛАБОРАТОРНОЇ РОБІТ

1.1. Самостійна робота

Самостійна робота полягає у вивченні теоретичного матеріалу, який викладений у даних методичних вказівках. Слід ознайомитися з основними системами числення, якими користуються при програмуванні мікроконтролера 8051АН сімейства MCS-51, системою його команд та організацією типових програмних конструкцій. Після засвоєння матеріалу слід відповісти на запитання самоперевірки та виконати індивідуальне завдання до самостійної роботи.

1.2. Лабораторна робота

Виконується у комп’ютерних класах кафедри. Метою роботи є теоретичне ознайомлення з переліком команд мікроконтролера 8051АН сімейства MCS-51, отримання навичок його програмування та налагодження програм. До виконання допускаються студенти, які ознайомилися з теоретичним матеріалом п.п. 2-4 методичних вказівок, відповіли на запитання самоперевірки, виконали індивідуальне завдання до самостійної роботи, підготували попередній звіт з алгоритмом (із дотриманням ГОСТ) та програмою (із коментарями) відповідно індивідуального завдання (варіант – за номером у журналі групи) вибраного рівня складності. Захист роботи – шляхом демонстрації викладачеві роботи розробленої програми та заключної тестової перевіркою на ПЕОМ.

Програма виконання

- Ознайомитись з системами числення, які застосовують при розробці програм на мові Асемблера, засвоїти особливості переведення чисел з однієї системи числення в іншу.
- Ознайомитися теоретично і отримати навички практичного використання команд пересилання даних, арифметичних та логічних операцій, операцій над бітами та командами передачі керування.
- Ознайомитись теоретично і отримати навички практичного використання типових програмних конструкцій на мові Асемблера.
- Після засвоєння матеріалу п.п. 2-4 методичних вказівок відповісти на запитання самоперевірки, які наведені після п. 4.

- Виконати індивідуальний варіант самостійної роботи (номер варіанту повинен збігатися з порядковим номером студента у журналі групи.
- Розробити на налагодити програму.
- Скласти підсумковий звіт та захистити роботу шляхом тестової перевірки.

Вказівки щодо складання звіту

Попередній звіт повинен містити:

- Назву, мету та програму роботи.
- Відповіді на запитання самоперевірки.
- Відповіді на запитання індивідуального варіанту самостійної роботи.
- Чернетку з алгоритмом та програмою індивідуального завдання вибраного рівня складності.

Підсумковий звіт, окрім перших трьох пунктів попереднього звіту, повинен містити алгоритм та налагоджену програму індивідуального завдання вибраного рівня складності, перевірені та підписані викладачем.

Увага! Допуском до виконання лабораторної роботи є підготовлений попередній звіт згідно поставлених до нього вимог.

2. СИСТЕМИ ЧИСЛЕННЯ

2.1. Двійкова система числення

У знайомій нам позиційній десятковій системі числення кожне число складається з цифр від 0 до 9. Значення кожної цифри визначається місцем її розташування у числі. Так, десяткове число 8547 є сумою $8\cdot10^3+5\cdot10^2+4\cdot10^1+7\cdot10^0$, тобто містить вісім тисяч, п`ять сотень, чотири десятки та сім одиниць. Число 10 називається основою десяткової системи.

У цифровій техніці домінує двійкова система – позиційна система числення, основою якої є число 2, тобто система має лише цифри 0 та 1. Двійкова система більш практична через те, що для представлення цифр 0 та 1 достатньо мати два рівня напруги (зазвичай цифрі 0 відповідає напруга близько 0 В, а цифрі 1 – близько +5 В). Кожна цифра двійкової системи числення має назву біт (від англійського терміну binary digit – двійкова цифра, скорочено bit). Наприклад, двійковому числу 1101 відповідає десяткове $1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0=8+4+0+1=13$. У такий спосіб можна перевести число із двійкової системи числення у десяткову, тобто, множенням кожного біту на відповідну його розташуванню ступінь двійки та наступного знаходження суми отриманих добутків.

Для переведення цілого десяткового числа без знаку у двійкове необхідно розділити його на 2 і знайти залишок від ділення. Далі необхідно взяти частку від ділення і вже її розділити на 2 із знаходженням наступного залишку. Надалі слід знову взяти частку від попереднього ділення та знову розділити її на 2 із залишком. Ділення продовжується доти, доки частка від якогось із ділень не стане рівною нулю. Для отримання двійкового числа залишки (знайдені шляхом послідовного ділення) беруться у зворотному порядку. Отже, перший знайдений

залишок стане молодшим (крайнім з правого боку), а останній – старшим (крайнім зліва) бітом двійкового числа.

Приклади переведення чисел із десяткової системи числення у двійкову:

$11_{10} = 5 \cdot 2 + 1$
$5 = 2 \cdot 2 + 1$
$2 = 1 \cdot 2 + 0$
$1 = 0 \cdot 2 + 1$
Залишки: 1, 1, 0, 1
$11_{10} = 1011_2$

$6_{10} = 3 \cdot 2 + 0$
$3 = 1 \cdot 2 + 1$
$1 = 0 \cdot 2 + 1$
Залишки: 0, 1, 1
$6_{10} = 110_2$

2.2. Шістнадцяткова система числення

Позиційна шістнадцяткова система числення (Hexadecimal) є системою з основою 16, використовує 16 цифр (0, 1,… 9, A, B, C, D, E, F), причому числу $F_{16} = 15_{10}$. Належність числа до шістнадцяткової системи числення визначається індексом 16 або додаванням після числа літери H. Наприклад, 21D = 15H або $21_{10} = 15_{16}$. Ця система числення легко перетворюється у двійкову і навпаки. Для перетворення двійкового числа у шістнадцяткове необхідно розбити двійкове число (починаючи з молодшого біта) на групи по 4 двійкові цифри, а потім замінити ці групи на їх шістнадцятковий еквівалент (табл. 2.1.). Відповідно для переводу шістнадцяткового числа у двійкове необхідно кожній з цифр цього числа поставити у відповідність їх двійковий еквівалент. Так, наприклад, число $4E_{16} = 1001110_2$.

Таблиця 2.1

Десяткові, шістнадцяткові та двійкові еквіваленти *)

Десяткові	Шістнадцяткові	Двійкові
1	2	3
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

) для виключення помилок у визначенні приналежності числа до 10-ої системи його помічають індексом 10 або дописують у кінці числа літеру D. Для двійкових чисел відповідно індекс 2 та літера B.

Для переведення шістнадцяткового числа у десяткове можна скористатися даними табл. 2.2.

Таблиця 2.2

До визначення числа у шістнадцятковій системі числення

Позиція цифри (справа наліво)	3	2	1	0
Степінь основи 16	3	2	1	0
Значення позиції (степенева форма)	16^3	16^2	16^1	16^0
Значення позиції (десяткова форма)	4096_{10}	256_{10}	16_{10}	1_{10}

Так, число 4A3FH=410·409610+1010·25610+310·1610+1510·110=1900710

Для переведення десяткового числа у шістнадцяткове можна скористатись діаграмою рис. 2.1, де наведений приклад переведення числа 1579710 у шістнадцяткове число 3DB516.

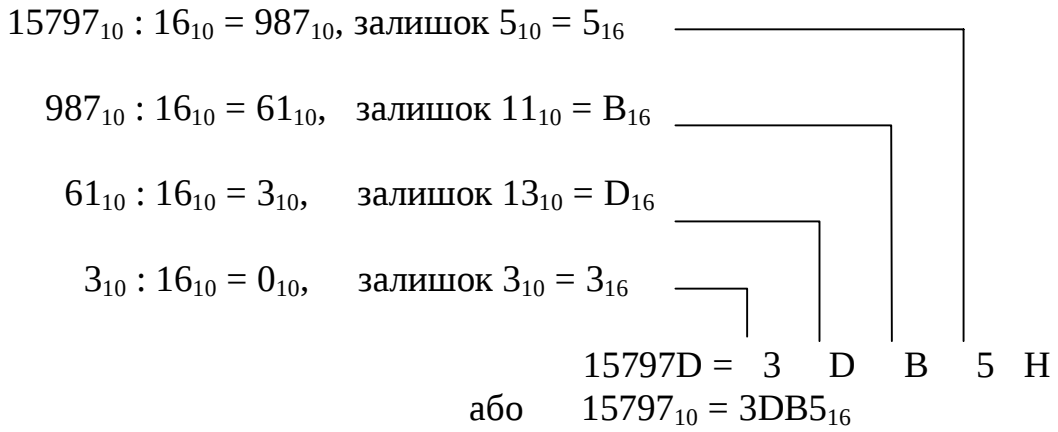


Рис. 2.1. До переведення десяткового числа в шістнадцяткове

2.3. Вісімкова система числення

Вісімкова позиційна система числення містить 8 цифр (від 0 до 7), має основу 8 та використовується для подання двійкових чисел. Для цього двійкове число (розпочинаючи з молодшого розряду) розбивається на групи по 3 біта (тріади), кожна з котрих замінюється вісімковим еквівалентом. Наприклад, двійкове число $10110011100101_2 \rightarrow 010\ 110\ 011\ 100\ 101_2 = 26345_8$. Вірним також буде запис: 10110011100101B = 26345O, де O – ознака вісімкового числа. Як і в інших позиційних системах числення, для знаходження десяткового еквівалента вісімкового числа 26345Q слід зробити наступне:

$$26345O=2*8^4+6*8^3+3*8^2+4*8^1+5*8^0=2*4096+6*512+3*64+4*8+5*1= 11493D$$

або $26345_8=11493_{10}$.

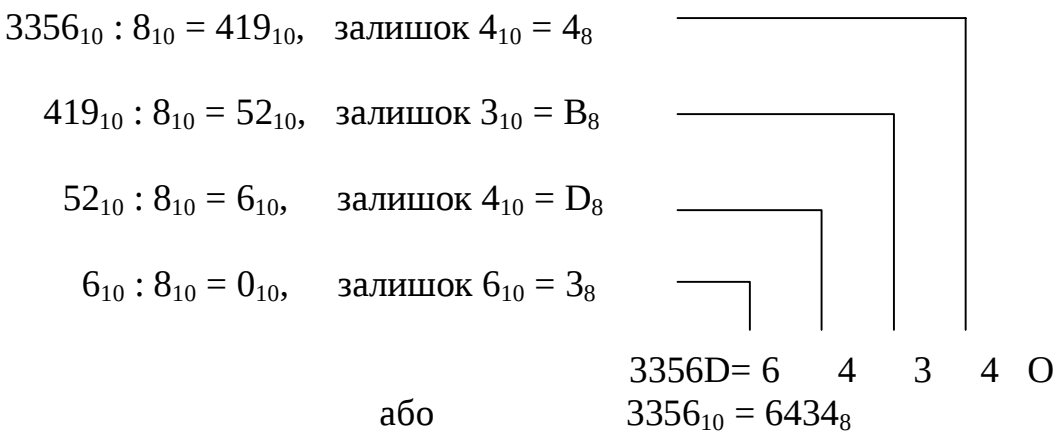


Рис. 2.2. До переведення десяткового числа у вісімкове

2.4. Двійково-десяткова система числення

Окрім перерахованих систем числення та форм запису чисел існує ще двійково-десятковий код (ДДК) десяткових чисел. Його особливість у тому, що для переведення десяткових чисел у двійково-десяткові числа кожній цифрі ставиться у відповідність двійкове число, яке складається з 4-х двійкових цифр (біт). При переведенні з двійково-десяткового коду у десятковий кожній тетраді бітів ставиться у відповідність десяткова цифра.

Приклади:

2356_{10}=0010 0011 0101 $1100_{ДДК}$;
1001 1000 0111 $0011_{ДДК}$=9873_{10}.

3. СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА MCS-51

3.1. Загальні відомості про систему команд мікроконтролера

Система команд мікроконтролера 8051AH – базова для сімейства MCS-51 – забезпечує реалізацію арифметичних і логічних операцій, а також керування у реальному часі. Мікроконтролер виконує операції над бітами, тетрадами (4 біта), байтами (8 біт) та словами (16 біт). Набір команд має 42 мнемонічних позначення (абревіатури) для визначення 33-х функцій системи. Мнемонічні позначення команд безпосередньо пов’язані з конкретними способами адресації та типами даних. Система команд 8051AH налічує 111 команд. Їх довжина – один, два або три байти, причому переважна більшість з них (94 %) – одно- або двобайтові. Всі команди виконуються за один або два машинних цикли (відповідно 1 або 2 мкс при тактовій частоті 12 МГц), виняток – команди множення і ділення, які виконуються за чотири машинних цикли (4 мкс). Мікроконтролери сімейства MCS-51 використовують пряму, непряму та неявну адресації.

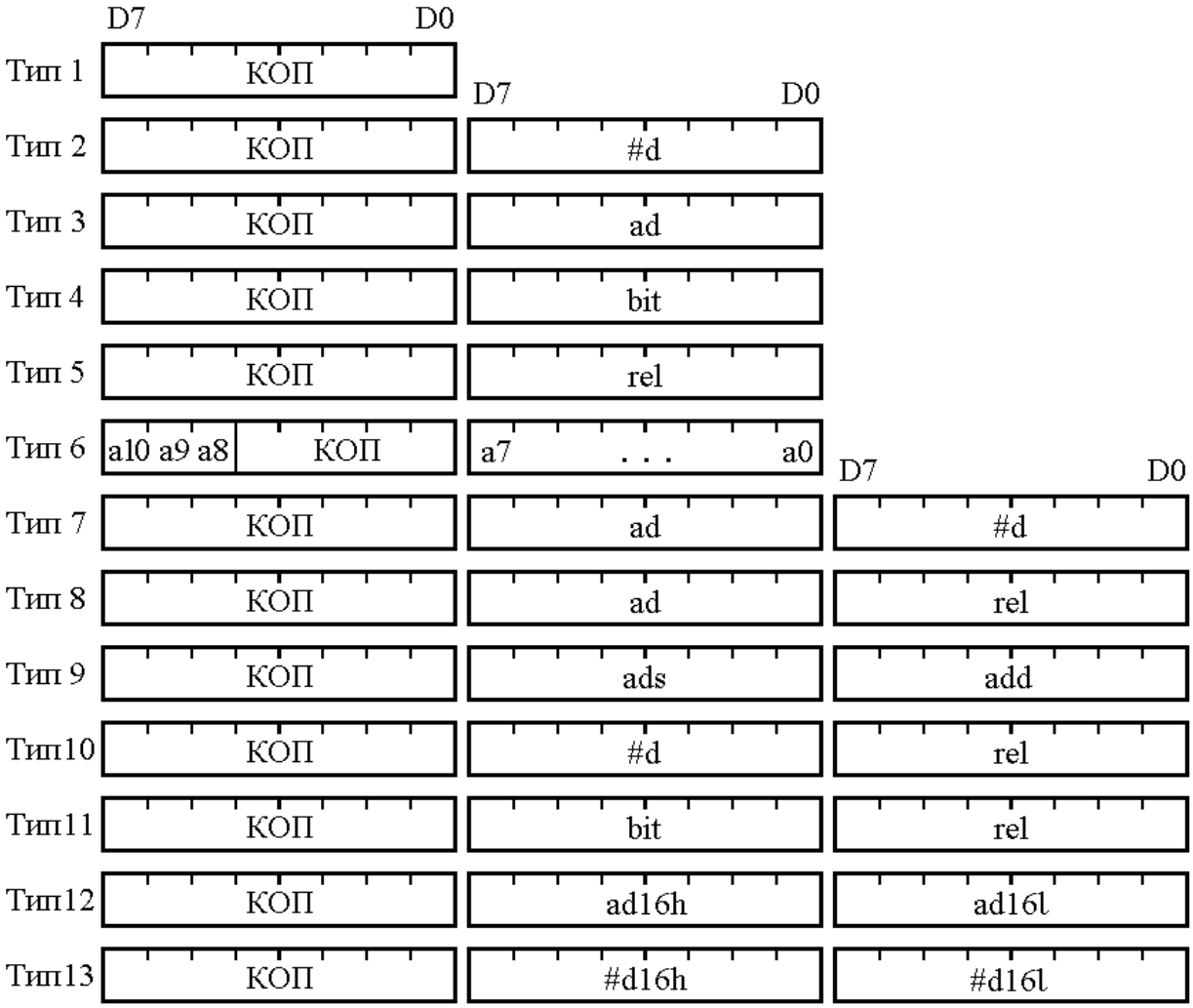


Рис. 3.1. Типи команд мікроконтролера 8051AH

Усього мікроконтролер виконує 13 типів команд (рис. 3.1). Перший байт команди завжди вміщує код операції (КОП), а другий і третій байти (якщо вони присутні) – адреси операндів або їх безпосередні значення. Вельми вагома особливість системи команд мікроконтролера 8051AH – можливість виконання операцій над бітами. Мікроконтролер може адресувати окремі біти у внутрішній пам’яті даних. Крім того, і деякі регістри спеціальних функцій (SFR) також допускають адресацію окремих біт. Карти адрес окремих бітів внутрішньої пам’яті даних і регістрів спеціальних функцій наведені на рис. 3.2 та 3.3 відповідно.

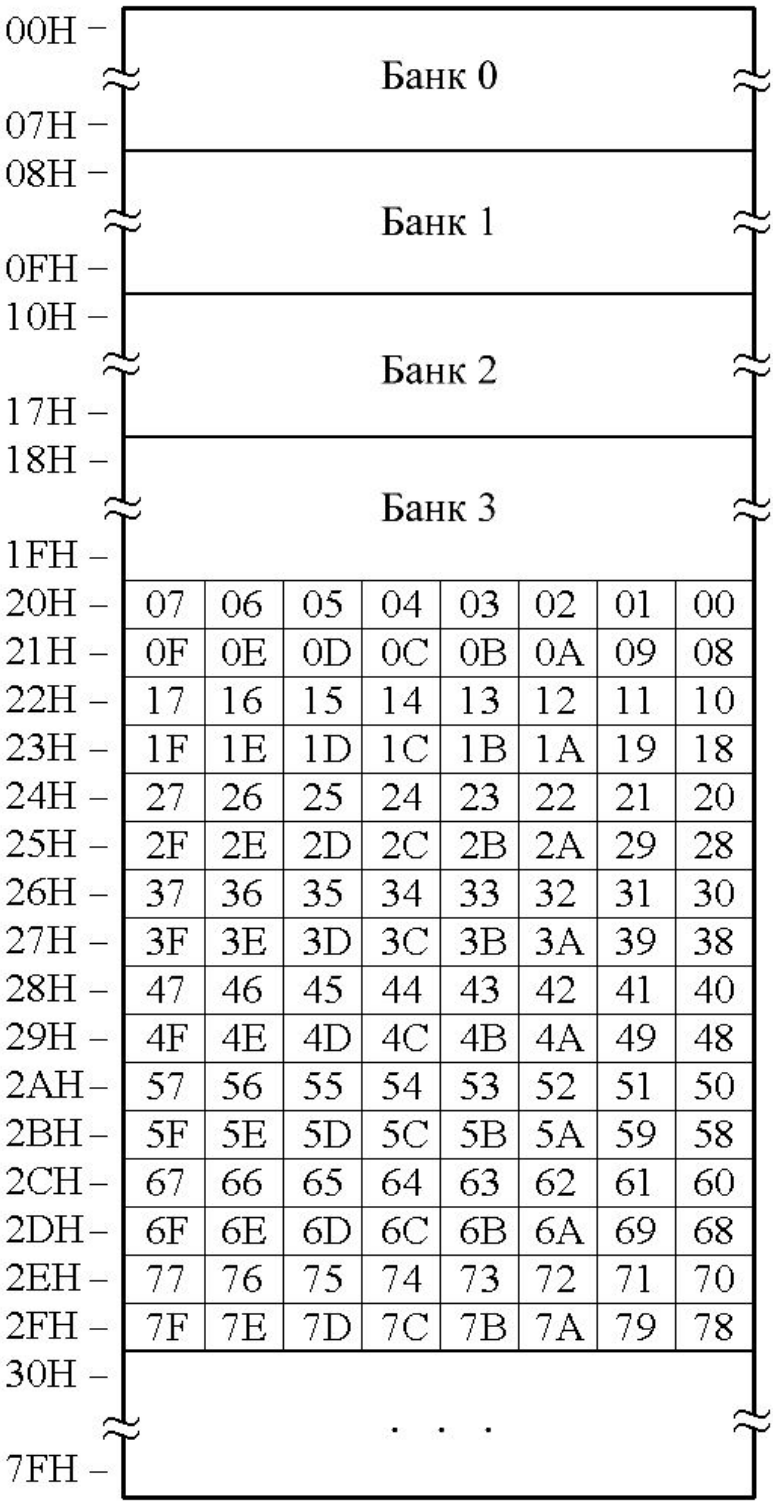


Рис. 3.2. Адреси програмно доступних бітів регістрів загального призначення

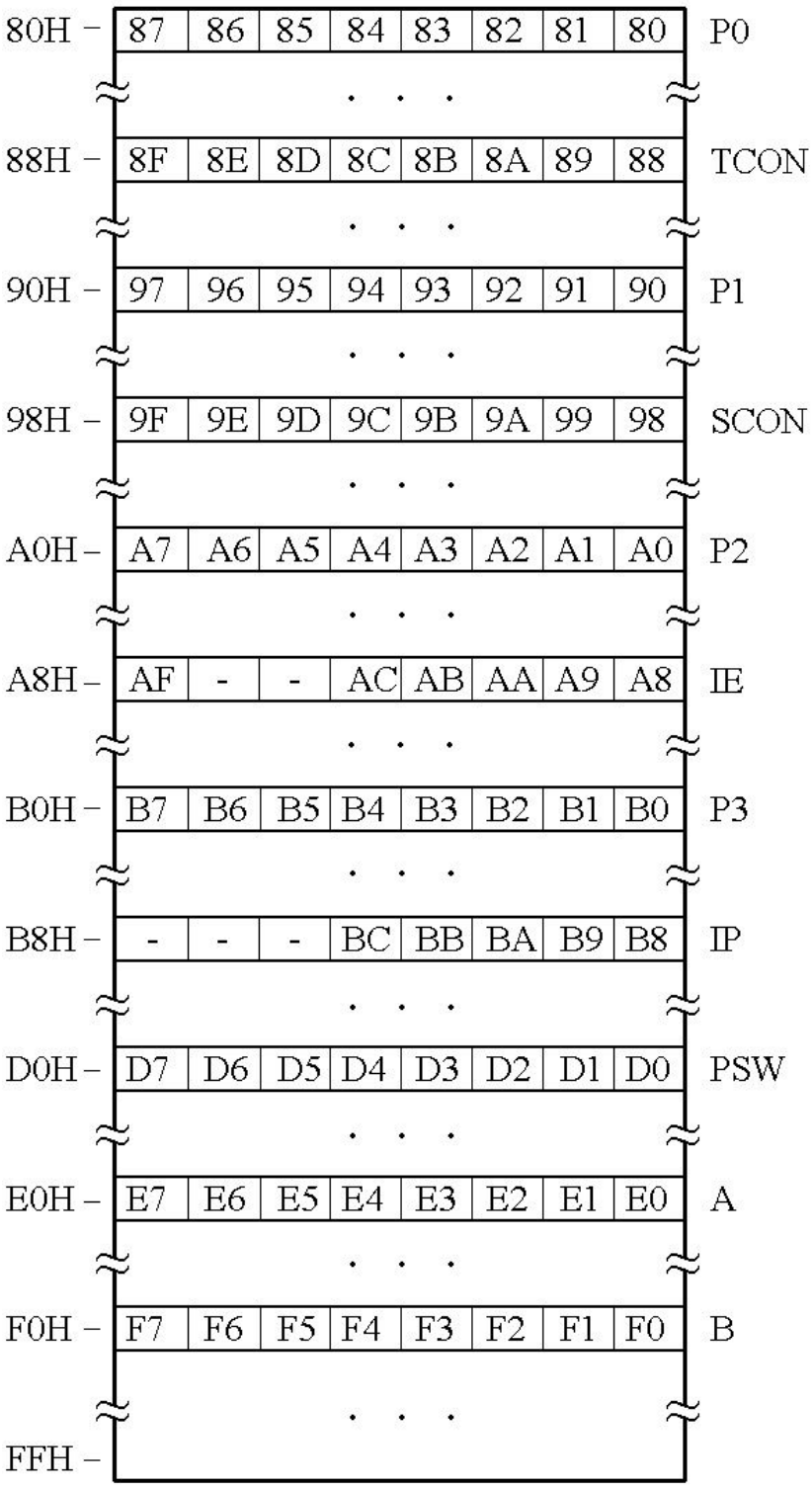


Рис. 3.3. Адреси програмно доступних бітів регістрів спеціальних функцій

Усі команди мікроконтролерів сімейства MCS-51 можна розбити на п’ять функціональних груп:

- пересилання даних;
- арифметичних операцій;
- логічних операцій;
- операцій над бітами;
- передачі управління.

При подальшому розгляді команд будуть використані такі позначення:

Rn (n=0, 1, …, 7)	Регістр загального призначення в обраному банку регістрів
Ri (i=0, 1)	Регістр загального призначення в обраному банку регістрів, що використовується як регістр посередньої адреси
Ad	Пряма адреса байта
Ads	Пряма адреса байта-джерела
add	Пряма адреса байта-одержувача
ad11	11-розрядна абсолютна адреса переходу
ad16	16-розрядна абсолютна адреса переходу
rel	Відносна адреса переходу
#d	Безпосереднє чисельне значення операнда
#d16	Безпосереднє значення двобайтного операнда
bit	Пряма адреса біта
/bit	Пряма адреса біта, який інвертується
A	Акумулятор
PC	Лічильник команд
DPTR	Регістр-покажчик даних
( )	Вміст елемента пам’яті чи регістра

3.2. Команди пересилання даних

Група команд пересилання даних містить 28 команд, короткі відомості про які наведені в табл. 3.1, де зазначені також тип команди (Т) відповідно до рис. 3.1, її довжина в байтах (Б) та час виконання в машинних циклах (Ц).

За командою MOV виконується перенесення даних з другого операнда до першого. При цьому вміст регістра або елемента пам’яті, позначеного другим оператором, не змінюється.

<u>Увага</u>! Оскільки регістри мікроконтролерів сімейства MCS-51 є одночасно елементами пам’яті з певними адресами, то командою MOV можна переслати дані із будь-якого елемента пам’яті до першого-ліпшого регістра; навпаки, із будь-якого регістра до будь-якого елемента пам’яті; і, нарешті, із будь-якого регістра до будь-якого регістра.

Команда MOV не має доступу ані до зовнішньої пам’яті даних, ані до зовнішньої пам’яті програм. Для цього призначені команди MOVX і MOVC відповідно. Перша з них забезпечує читання/запис байт із зовнішньої пам’яті даних, друга – читання байт із зовнішньої пам’яті програм.

Таблиця 3.1

Команди пересилання даних

Мнемокод		**КОП**	**Т**	**Б**	**Ц**	**Коментар**
MOV	A,Rn	11101rrr	1	1	1	(A)←(Rn)
MOV	A,ad	11100101	3	2	1	(A)←(ad)
MOV	A,@Ri	1110011i	1	1	1	(A)←((Ri))

MOV	A,#d	01110100	2	2	1	(A)←#d
MOV	Rn,A	11111rrr	1	1	1	(Rn)←(A)
MOV	Rn,ad	10101rrr	3	2	2	(Rn)←(ad)
MOV	Rn,#d	01111rrr	2	2	1	(Rn)←#d
MOV	ad,A	11110101	3	2	1	(ad)←(A)
MOV	ad,Rn	10001rrr	3	2	2	(ad)←(Rn)
MOV	add,ads	10000101	9	3	2	(add)←(ads)
MOV	ad,@Ri	1000011i	3	2	2	(ad)←((Ri))
MOV	ad,#d	01110101	7	3	2	(ad)←#d
MOV	@Ri,A	1111011i	1	1	1	((Ri))←(A)
MOV	@Ri,ad	1010011i	3	2	2	((Ri))←(ad)
MOV	@Ri,#d	0111011i	2	2	1	((Ri))←#d
MOV	DPTR,#d16	10010000	13	3	2	(DPTR)←#d16
MOVC	A,@A+DPTR	10010011	1	1	2	(A)←((A)+(DPTR))
MOVC	A,@A+PC	10000011	1	1	2	(PC)←(PC)+1 (A)←((A)+(PC))
MOVX	A,@Ri	1110001i	1	1	2	(A)←((Ri))
MOVX	A,@DPTR	11100000	1	1	2	(A)←((DPTR))
MOVX	@Ri,A	1111001i	1	1	2	((Ri))←(A)
MOVX	@DPTR,A	11110000	1	1	2	((DPTR))←(A)
PUSH	ad	11000000	3	2	2	(SP)←(SP)+1 ((SP))←(ad)
POP	ad	11010000	3	2	2	(ad)←((SP)) (SP)←(SP)-1
XCH	A,Rn	11001rrr	1	1	1	(A)↔(Rn)
XCH	A,ad	11000101	3	2	1	(A)↔(ad)
XCH	A,@Ri	1100011i	1	1	1	(A) ↔((Ri))
XCHD	A,@Ri	1101011i	1	1	1	$(A_{3\text{-}0})\leftrightarrow((Ri_{3\text{-}0}))$

За командою XCH виконується обмін байтами між акумулятором і регістром або акумулятором і елементом пам’яті. Виконання команди XCHD призводить до обміну молодшими тетрадами (бітами D0…D3) між акумулятором і елементом внутрішньої пам’яті даних.

При зверненні до портів мікроконтролера (як і до інших регістрів спеціальних функцій) задається відповідна пряма адреса. Наприклад, виведення в порт P3 числа, що зберігається в акумуляторі, може бути виконано командою MOV P3,A , яка перетворюється АСЕМБЛЕРом до вигляду MOV 0B0H,A (послідовність машинних кодів F5, B0, де F5 – код операції, B0 – адреса порту P3):

Мнемокод	**Код операції**	**Машинний код**	**Коментар**
MOV P3,A	11110101	F5B0	Обидві команди абсолютно рівнозначні

MOV 0B0H,A	11110101	F5B0	за своєю дією і відрізняються лише записом мнемонічного коду

Слід зазначити, що більшість АСЕМБЛЕРів як трансляторів у машинні коди допускає використання символічних імен для регістрів спеціальних функцій. Тобто, можливі такі команди: MOV P3,A; MOV B,A; MOV P3,@R1; MOV B,#31; XCH A,SBUF; MOV IP,IE; MOV TH0,TL1 та багато інших. При трансляції таких команд у машинні коди виконується заміна символічного імені регістра його прямою адресою.

Окремі біти деяких регістрів спеціальних функцій можуть адресуватися зазначенням імені регістра та номера біта через крапку. Наприклад, до нульового біта акумулятора можна звернутися на ім'я ACC.0. Таким чином, акумулятор мікроконтролера має два імені залежно від способу адресації: A – при непрямій адресації (наприклад, MOV A,R0) і ACC – при використанні прямої адреси. Перший спосіб кращий, оскільки більш економний з точки зору швидкодії та об'єму використаної пам'яті програм. Нижче наведені дві команди, які виконують однакові дії – переміщують байт із регістра B до акумулятора, але використовують різні способи адресації:

Мнемокод	**Код операції**	**Машинний код**	**Спосіб адресації**	**Коментар**
MOV ACC,B	10000101 (085H)	85 F0 E0	Пряма для акумулятора і регістра B	E0 – адреса акумулятора, F0 – адреса регістра B
MOV A,B	11100101 (0E5H)	E5 F0	Непряма для акумулятора, пряма для B	F0 – адреса регістра B

Коди першої команди із таблиці займають 3 байти пам'яті програм, а другої – 2 байти. Перша команда виконується за два машинних цикли, а друга – за один, але використання непрямої адресації не завжди можливе. Наприклад, перенесення байта із регістра TL0 до регістра TH0 може бути виконано однією командою тільки за умов прямої адресації обох регістрів.

Система команд мікроконтролерів сімейства MCS-51 має в своєму складі команди PUSH і POP роботи зі стеком. Перша з них виконує запис байту прямо адресованої змінної до стекової пам'яті, а друга – читання байта із стека в прямо адресовану змінну (наприклад, PUSH DPL; POP DPL). Розмір стека обмежений об'ємом резидентної (внутрішньої) пам'яті даних.

Більш докладно з командами перенесення даних можна ознайомитись, розглянувши приклади (додаток А).

3.3. Команди арифметичних операцій

До цієї групи належить 24 команди (табл. 3.2).

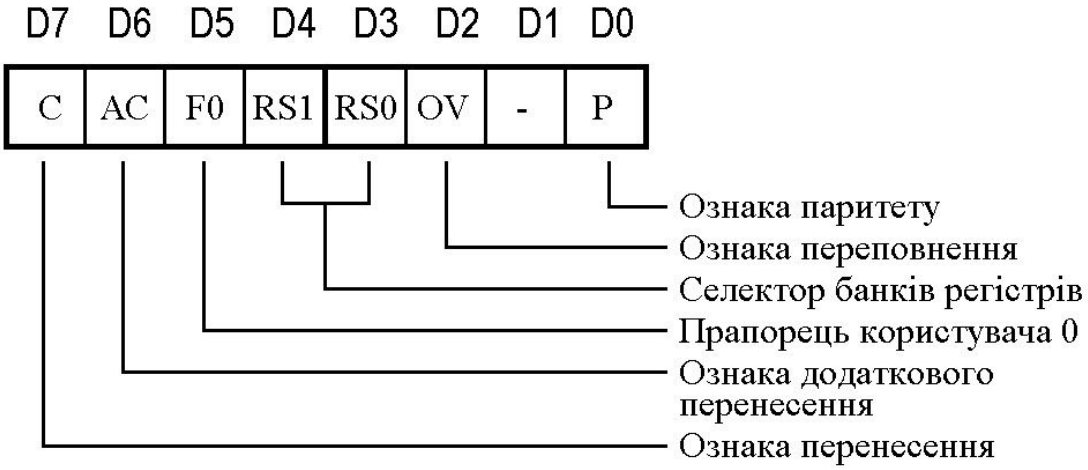
У таблиці зазначені тип команди (Т) за рис. 3.1, її довжина в байтах (Б) та час виконання в машинних циклах (Ц). Як видно, мікроконтролер виконує до-

сить широкий набір команд для організації обробки цілочисельних даних, у тому числі множення та ділення. За результатом виконання команд ADD, ADDC, SUBB, MUL і DIV модифікуються прапорці (ознаки) байта стану програми PSW із структурою вигляду

Таблиця 3.2

Команди арифметичних операцій

Мнемокод		**КОП**	**Т**	**Б**	**Ц**	**Коментар**
ADD	A,Rn	00101rrr	1	1	1	(A)←(A)+(Rn)
ADD	A,ad	00100101	3	2	1	(A)←(A)+(ad)
ADD	A,@Ri	0010011i	1	1	1	(A)←(A)+((Ri))
ADD	A,#d	00100100	2	2	1	(A)←(A)+#d
ADDC	A,Rn	00111rrr	1	1	1	(A)←(A)+(Rn)+(C)
ADDC	A,ad	00110101	3	2	1	(A)←(A)+(ad)+(C)
ADDC	A,@Ri	0011011i	1	1	1	(A)←(A)+((Ri))+(C)
ADDC	A,#d	00110100	2	2	1	(A)←(A)+#d+(C)
DA	A	11010100	1	1	1	Десяткова корекція
SUBB	A,Rn	10011rrr	1	1	1	(A)←(A)-(Rn)-(C)
SUBB	A,ad	10010101	3	2	1	(A)←(A)-(ad)-(C)
SUBB	A,@Ri	1001011i	1	1	1	(A)←(A)-((Ri))-(C)
SUBB	A,#d	10010100	2	2	1	(A)←(A)-#d-(C)
INC	A	00000100	1	1	1	(A)←(A)+1
INC	Rn	00001rrr	1	1	1	(Rn)←(Rn)+1
INC	ad	00000101	3	2	1	(ad)←(ad)+1
INC	@Ri	0000011i	1	1	1	((Ri))←((Ri))+1
INC	DPTR	10100011	1	1	2	(DPTR)←(DPTR)+1
DEC	A	00010100	1	1	1	(A)←(A)-1
DEC	Rn	00011rrr	1	1	1	(Rn)←(Rn)-1
DEC	ad	00010101	3	2	1	(ad)←(ad)-1
DEC	@Ri	0001011i	1	1	1	((Ri))←((Ri))-1
MUL	AB	10100100	1	1	4	(B)(A)←(A)*(B)
DIV	AB	10000100	1	1	4	(A).(B)←(A)/(B)



Прапорець перенесення C встановлюється в одиничний стан при перенесенні із розряду D7, тобто тоді, коли результат арифметичної дії не вміщується в один байт; прапорець допоміжного перенесення AC встановлюється при перенесенні із розряду D3 в командах додавання і віднімання і служить для реалізації десяткової арифметики (ознака використовується командою десяткової корекції DAA).

Прапорець переповнення OV встановлюється при перенесенні із розряду D6, тобто тоді, коли результат арифметичної дії не вміщується в 7 розрядів і старший (восьмий) біт акумулятора не може бути інтерпретований як знаковий. Ця ознака використовується для опрацювання чисел зі знаком.

Нарешті, прапорець паритету P встановлюється, якщо кількість одиничних біт в акумуляторі парна. Коли ж непарна – прапорець P скидається в нульовий стан.

3.4. Команди логічних операцій

Ця група містить 25 команд (табл. 3.3), які дають змогу виконувати операції над байтами: логічне множення (AND, $\wedge$), логічне додавання (OR, $\vee$), додавання за модулем 2 (XOR, $\oplus$), інверсію (NOT), скидання в нульовий стан і зсув.

Таблиця 3.3

Команди логічних операцій

Мнемокод		КОП	Т	Б	Ц	Коментар
ANL	A,Rn	01011rrr	1	1	1	(A)←(A)∧(Rn)
ANL	A,ad	01010101	3	2	1	(A)←(A)∧(ad)
ANL	A,@Ri	0101011i	1	1	1	(A)←(A)∧((Ri))
ANL	A,#d	01010100	2	2	1	(A)←(A)∧#d
ANL	ad,A	01010010	3	2	1	(ad)←(ad)∧(A)
ANL	ad,#d	01010011	7	3	2	(ad)←(ad)∧#d
ORL	A,Rn	01001rrr	1	1	1	(A)←(A)∨(Rn)
ORL	A,ad	01000101	3	2	1	(A)←(A)∨(ad)

ORL	A,@Ri	0100011i	1	1	1	(A)←(A)∨((Ri))
ORL	A,#d	01000100	2	2	1	(A)←(A)∨#d
ORL	ad,A	01000010	3	2	1	(ad)←(ad)∨(A)
ORL	ad,#d	01000011	7	3	2	(ad)←(ad)∨#d
XRL	A,Rn	01101rrr	1	1	1	(A)←(A)⊕(Rn)
XRL	A,ad	01100101	3	2	1	(A)←(A)⊕(ad)
XRL	A,@Ri	0110011i	1	1	1	(A)←(A)⊕((Ri))
XRL	A,#d	01100100	2	2	1	(A)←(A)⊕#d
XRL	ad,A	01100010	3	2	1	(ad)←(ad)⊕(A)
XRL	ad,#d	01100011	7	3	2	(ad)←(ad)⊕#d
CLR	A	11100100	1	1	1	(A)←0
CPL	A	11110100	1	1	1	(A)←NOT(A)
SWAP	A	11000100	1	1	1	$(A_{7\text{-}4})\leftrightarrow(A_{3\text{-}0})$
RL	A	00100011	1	1	1	Циклічний зсув ліворуч

Продовження таблиці 3.3

1	**2**	3	4	5	**6**
RLC	A	00110011	1	1	1	Зсув ліворуч через біт перенесення (C)
RR	A	00000011	1	1	1	Циклічний зсув праворуч
RRC	A	00010011	1	1	1	Зсув праворуч через біт (C)

3.5. Команди операцій над бітами

У складі цієї групи – 12 команд (табл. 3.4). Команди операцій над бітами дозволяють встановлювати в одиницю, скидати в нуль, інвертувати окремі біти, а також виконувати логічне множення (AND, ∧) та додавання (OR, ∨) біт тощо.

Таблиця 3.4

Команди операцій над бітами

Мнемокод		**КОП**	Т	Б	Ц	**Коментар**
CLR	C	11000011	1	1	1	(C)←0
CLR	bit	11000010	4	2	1	(bit)←0
SETB	C	11010011	1	1	1	(C)←1
SETB	bit	11010010	4	2	1	(bit)←1
CPL	C	10110011	1	1	1	(C)←NOT(C)
CPL	bit	10110010	4	2	1	(bit)←NOT(bit)
ANL	C,bit	10000010	4	2	2	(C)←(C)∧(bit)
ANL	C,/bit	10110000	4	2	2	(C)←(C)∧NOT(bit)
ORL	C,bit	01110010	4	2	2	(C)←(C)∨(bit)
ORL	C,/bit	10100000	4	2	2	(C)←(C)∨NOT(bit)
MOV	C,bit	10100010	4	2	2	(C)←(bit)
MOV	bit,C	10010010	4	2	2	(bit)←(C)

При виконанні операцій над бітами як “логічний акумулятор” виступає прапорець ознаки перенесення C (розряд D7 байта стану програми PSW). Він бере участь в усіх операціях над двома бітами. Ознака перенесення C є бітом-джерелом операнда і бітом-одержувачем результату. У ролі операндів можуть використовуватися 128 біт із резидентної (внутрішньої) пам’яті даних і біти деяких регістрів спеціальних функцій (карти адрес окремих бітів у внутрішній пам’яті даних і у регістрах спеціальних функцій наведені на рис. 3.2 та 3.3 відповідно).

Більшість Асемблерів як трансляторів програм у машинні коди надають можливість адресувати окремі біти регістрів спеціальних функцій (SFR) за допомогою використання символьних імен цих регістрів із зазначенням номера конкретного біта. Нижче, як приклад, наведені два можливих варіанти запису однієї команди скидання в нуль старшого біта акумулятора. Перший варіант використовує символічне ім’я акумулятора із зазначенням номера біта (ACC.7), а другий – запис безпосередньої адреси біта, що скидається, у мнемонічному коді команди (0E7H):

Мнемокод	**Код операції**	**Машинний код**	**Коментар**
CLR ACC.7	11000010 (0C2H)	C2 E7	Одна й та сама команда може бути записана двома способами: з використанням символічного імені регістра або прямої адреси біта
CLR 0E7H	11000010 (0C2H)	C2 E7	

Для зазначення окремих біт, які не є бітами регістрів спеціальних функцій використовується, як правило, пряма адреса біта, хоча деякі Асемблери дозволяють адресувати ці біти зазначенням адреси байта, до якого належить біт, та номера біта. Нижче наведені два можливих варіанти запису однієї команди скидання старшого біта елемента пам’яті з адресою 20H. Перший варіант вказує на адресу байта і через крапку – номер біта (20.7), другий варіант визначає біт, що скидається, безпосередньо за його адресою:

Мнемокод	**Код операції**	**Машинний код**	**Коментар**
CLR 20.7	11000010 (0C2H)	C2 07	Одна й та сама команда може бути записана двома різними способами, але транслюється в однакові машинні коди
CLR 07H	11000010 (0C2H)	C2 07	

3.6. Команди передачі керування

Ця група подана командами безумовного і умовного переходів, виклику підпрограм та повернення із підпрограм (табл. 3.5).

Таблиця 3.5

Команди передачі керування

Мнемокод		**КОП**	Т	Б	Ц	**Коментар**
1		**2**	3	4	5	**6**
LJMP	ad16	00000010	12	3	2	Довгий безумовний перехід на будь-яку адресу пам’яті
AJMP	ad11	$a_{10}a_9a_8$00001	6	2	2	Безумовний перехід у межах сторінки 2 Кбайт
SJMP	rel	10000000	5	2	2	Безумовний перехід у межах сторінки 256 байт
JMP	@A+DPTR	01110011	1	1	2	Безумовний перехід за непрямою адресою
JZ	rel	01100000	5	2	2	Перехід, якщо нуль
JNZ	rel	01110000	5	2	2	Перехід, якщо не нуль
JC	rel	01000000	5	2	2	Перехід, якщо (C)=1
JNC	rel	01010000	5	2	2	Перехід, якщо (C)=0
JB	bit,rel	00100000	11	3	2	Перехід, якщо (bit)=1

Продовження таблиці 3.5

1		**2**	3	4	5	**6**
JNB	bit,rel	00110000	11	3	2	Перехід, якщо (bit)=0
JBC	bit,rel	00010000	11	3	2	Перехід, якщо (bit)=1 із скиданням біта (bit)←0
DJNZ	Rn,rel	11011rrr	5	2	2	Команда циклу
DJNZ	ad,rel	11010101	8	3	2	Команда циклу
CJNE	A,ad,rel	10110101	8	3	2	Порівняння (A) з (ad) і перехід, якщо ≠
CJNE	A,#d,rel	10110100	10	3	2	Порівняння (A) з числом #d і перехід, якщо ≠
CJNE	Rn,#d,rel	10111rrr	10	3	2	Порівняння (Rn) з #d і перехід, якщо ≠
CJNE	@Ri,#d,rel	1011011i	10	3	2	Порівняння ((Ri)) з #d і перехід, якщо ≠
LCALL	ad16	00010010	12	3	2	Довгий виклик під-програми із всього ПЗП
ACALL	ad11	$a_{10}a_9a_8$10001	6	2	2	Виклик підпрограми у межах сторінки 2 Кбайт
RET		00100010	1	1	2	Повернення із підпрограми
RET	I	00110010	1	1	2	Повернення із підпрограми обробки переривання
NOP		00000000	1	1	1	Пуста операція

Команда безумовного переходу LJMP (L – long, довгий; JMP – jump, стрибок) здійснює перехід за абсолютною 16-бітною адресою, зазначеною в тілі ко-

манди. Таким чином, команда забезпечує перехід в будь-яку точку пам’яті програм.

Дія команди AJMP аналогічна попередній, проте в тілі команди – лише 11 молодших розрядів адреси. Тому перехід здійснюється у межах сторінки пам’яті розміром 2 Кбайт. При цьому треба мати на увазі, що спочатку вміст лічильника команд збільшується на 2 і лише після цього замінюються 11 молодших розрядів адреси.

На відміну від попередніх у команді SJMP (S – short, короткий) зазначена не абсолютна, а відносна адреса переходу. Величина зміщення rel визначає, наскільки одиниць збільшиться (або зменшиться) вміст програмного лічильника при переході. Зміщення rel – це знакове однобайтне число, старший біт якого визначає напрямок переходу – вгору чи вниз. Сім молодших бітів містять модуль зміщення. Таким чином, перехід може здійснюватись у межах –128…+127 байт відносно наступної команди.

Команда посереднього переходу JMP @A+DPTR дозволяє обчислювати адресу переходу в процесі виконання самої програми, що досить зручно для задання функцій табличним способом та побудови різноманітних регуляторів.

Командами умовного переходу можна перевіряти такі умови:

- JZ – акумулятор містить нульове значення;
- JNZ – акумулятор містить ненульове значення;
- JC – біт перенесення C встановлений (дорівнює 1);
- JNC – біт перенесення C не встановлений (дорівнює 0);
- JB – біт, що адресується, встановлений (дорівнює 1);
- JNB – біт, що адресується, не встановлений (дорівнює 0);
- JBC – біт, що адресується, дорівнює 1 і скидається при виконанні команди.

Усі команди умовного переходу мікроконтролера 8051AH містять коротку відносну адресу rel, тобто перехід може здійснюватись у межах –128…+127 байт відносно наступної команди.

Команда DJNZ призначена для організації програмних циклів. Регістр Rn або елемент пам’яті за адресою ad, що в тілі команди, містить лічильник повторень циклу, а зміщення rel – відносну адресу переходу до початку циклу. При виконанні команди вміст лічильника зменшується на 1 і перевіряється на 0. Якщо вміст лічильника не дорівнює 0, то здійснюється перехід до початку циклу, інакше виконується наступна команда (тобто цикл завершений).

Команда CJNE – зручна для реалізації процедур очікування зовнішніх подій. Тіло команди містить координати двох байт і відносну адресу переходу rel. У якості двох байт можуть бути використані значення вмісту акумулятора, регістра або елемента пам’яті та константа. При виконанні команди значення двох зазначених байт порівнюються і у випадку, якщо вони неоднакові, здійснюється перехід. Наприклад, команда

```
metka1:  CJNE A,P0,metka1
```

виконуватиметься до того часу, поки значення на лініях порту P0 не будуть збігатися із вмістом акумулятора.

Дія команд виклику підпрограм повністю аналогічна дії команд безумовного переходу. Єдиний виняток: при виконанні команд виклику підпрограм у стековій пам’яті зберігається адреса повернення. Після виконання підпрограми мікроконтролер продовжує виконувати основну програму з команди, що знаходиться безпосередньо після команди виклику.

Команда повернення із підпрограми RET відновлює із стека значення вмісту лічильника команд, а команда повернення із процедури обробки переривання RET I, окрім того, дозволяє переривання того рівня, який був щойно обслужений. Команди RET і RET I не розрізняють, якою командою – LCALL ad16 чи ACALL ad11 – була викликана підпрограма. В обох випадках у стековій пам’яті зберігається повна 16-розрядна адреса повернення.

Слід відзначити, що більшість Асемблерів допускають узагальнену мнемоніку JMP – для команд безумовного переходу і CALL – для команд виклику підпрограм. Конкретний тип команди (LJMP, AJMP чи SJMP; LCALL чи ACALL) визначається Асемблером, виходячи із “довжини” переходу або виклику. Команди LJMP і LCALL забезпечують доступ у будь-яку точку простору програмної пам’яті, а решта команд переходу і виклику мають обмежений доступний простір.

На прикладах додатка А можна детальніше ознайомитися з командами: пересилання даних (А.1), арифметичних операцій (А.2), логічних операцій (А.3), операцій над бітами (А.4), операцій передачі управління (А.5).

4. ТИПОВІ ПРОГРАМНІ КОНСТРУКЦІЇ

4.1. Організація підпрограми із використанням стекової пам’яті

Підпрограма є закінченим модулем програми, який можна викликати необхідну кількість разів. Звернення до підпрограми здійснюється командою виклику CALL NAME, де NAME – символічне ім'я процедури. Ім'я процедури використовують як мітку, яка позначає одну з команд (найчастіше першу) підпрограми. Для Intel 8051 мнемонічне значення CALL є узагальненим і транслюється в одну з команд ACALL або LCALL залежно від адресної відстані до підпрограми, що викликається.

При виконанні команди CALL в стеку зберігається вміст лічильника команд і повернення з підпрограми здійснюється в те місце основної програми, звідки був здійснений виклик (до команди основної програми, наступної за командою CALL). Для цього будь-яка підпрограма повинна закінчуватися командою повернення RET, яка відновлює вміст програмного лічильника зі стека. Кількість підпрограм, які можуть бути викликані таким чином (глибина вкладеності підпрограм), обмежується лише місткістю стека.

Достатньо часто виникає необхідність такої організації обчислювального процесу, за якої підпрограма викликає іншу підпрограму, та в свою чергу викликає наступну і т.д. Цей процес називається вкладенням підпрограм.

Іноді при зверненні до підпрограми виникає необхідність зберегти не тільки адресу повернення в основну програму, але і вміст окремих робочих регістрів. Зручним способом для цього є перемикання банку регістрів. Наприклад, якщо основна програма використовує банк регістрів 0, то підпрограма може використовувати банк регістрів 1. Проте перемикання банку регістрів не забезпечує збереження вмісту акумулятора, що приводить до необхідності створювати в одному з робочих регістрів або в пам'яті копію акумулятора. Звичайно для цих цілей використовують стекову пам'ять. При цьому на початку підпрограми вміст регістрів заноситься в стек, а в кінці підпрограми – його повернення в регістри.

Приклад підпрограми із збереженням вмісту регістрів, акумулятора і регістрів ознак:

```
Мітка    Команда      Коментар
NAME:    PUSH  0D0H   ; Збереження вмісту байта-стану
         PUSH  0E0H   ; Збереження вмісту акумулятора
         PUSH  0F0H   ; Збереження вмісту регістра B

                      ; Основна частина підпрограми.

         POP  0F0H    ; Відновлення вмісту регістра B
         POP  0E0H    ; Відновлення вмісту акумулятора
         POP  0D0H    ; Відновлення байта-стану
         RET          ; Повернення з підпрограми в основну програму
```

4.2. Організація програмних затримок часу

При керуванні у режимі реального часу часто виникає необхідність організації затримок часу. Можливі два варіанти вирішення цього завдання. Один – за допомогою використання спеціальних апаратних засобів. Другий варіант передбачає організацію затримок часу програмним шляхом за умови, що впродовж організації програмної затримки часу мікроконтролер вільний від інших операцій і необхідності у використанні спеціальних апаратних засобів не виникає. На практиці програмна організація затримок часу зводиться до виклику в необхідний момент часу спеціальної підпрограми, яка не формує ніяких керуючих сигналів або інших дій, але вимагає для свого виконання відповідних витрат часу.

У тому випадку, коли в період необхідної затримки часу мікроконтролер зайнятий іншими обчисленнями, його використання для програмної організації затримки часу неможливе і, отже, виникає необхідність пошуку апаратних засобів вирішення цього завдання (наприклад, програмованих 16-бітових таймерів/лічильників T/C0 і T/C1 мікроконтролера).

Часова затримка може бути реалізована на основі використання процедури програмного циклу. При цьому в заданий регістр завантажується число, яке потім при кожному повторенні групи команд циклу зменшується на одиницю.

Так продовжується до моменту, коли вміст вибраного регістру не стане рівним нулю (що інтерпретується мікропроцесорною програмою як момент виходу з циклу). При цьому тривалість затримки визначається числом, яке завантажене в регістр, і терміном виконання команд у тілі циклу. Наприклад, термін затримки, сформованої підпрограмою TIME1, визначатиметься виразом:

$$T = \frac{12(3+2n)}{f},$$

де f – частота синхронізації мікроконтролера.

```
Мітка    Команда      Коментар
TIME1:   MOV R2,#n    ; Занесення числа до регістру R2
M1:      DJNZ R2,M1   ; Зменшення на 1-цю вмісту R2,
                      ; якщо R2≠0 – перехід
         RET          ; Повернення з підпрограми.
```

При тій же частоті синхронізації тривалішу затримку можна отримати за допомогою вкладення циклів (підпрограма TIME2, для якої $T = 1 + n(2m + 2 + 1) + 2$, мкс):

```
Мітка    Команда      Коментар
TIME2:   MOV R1,#n    ; Занесення числа до регістру R1
M2:      MOV R2,#m    ; Занесення числа до регістру R2
M1:      DJNZ R2,M1   ; Зменшення на 1-цю вмісту R2,
                      ; якщо R2≠0 – перехід
         DJNZ R1,M2   ; Зменшення на 1-цю вмісту R1,
                      ; якщо R1≠0 – перехід
         RET          ; Повернення з підпрограми
```

За розрахунком максимальний термін затримки, сформованої підпрограмою TIME2, становить $T_{\max} = 130818\ мкс \approx 0{,}13\ с$.

4.3. Отримання додаткових кодів чисел

При виконанні операцій віднімання однобайтних чисел результатом може бути від'ємне число (коли із меншого числа віднімають більше). Про такий результат свідчить встановлення в одиницю ознаки переповнення C у байті стану PSW (див стор. 16). Одночасно це означає, що результат отримано у додатковому коді. Для отримання модулю цього від'ємного числа слід спочатку знайти його інверсний код (одиниці замінити на нулі і навпаки), а потім збільшити результат на одиницю. Наприклад, для знаходження модуля від'ємної різниці 7 - 15 = - 8 слід виконати наступну програму:

```
Мітка    Команда      Коментар
         MOV A,#07H   ; В аккумулятор занесено зменшуване число 710
         MOV B,#0FH   ; В регістр B занесено від'ємник 1510
         SUBB A, B    ; (A)←F8H. (C)=1 свідчить, що в акумуляторі
                      ; знаходиться додатковий код різниці (-8)
         CPL A        ; (A)←07H – інверсний код різниці F8H
```

```
        INC A         ; (A)←08H – модуль від’ємної різниці
```

****) При відніманні семирозрядних чисел функцію ознаки С бере на себе старший біт акумулятора (А).***

Операцію віднімання двох чисел можна замінити на операцію додавання до зменшуваного додаткового коду від’ємника. За цих умов ознака С=0 свідчить, що результатом є додатковий код від’ємного числа. Програма реалізації такої операції для попереднього прикладу може мати наступний вигляд:

```
Мітка    Команда      Коментар
         MOV B,#07H   ; В регістр В занесено зменшуване число 710
         MOV A,#0FH   ; В аккумулятор занесено від’ємник 1510
         CPL A        ; (A)←F0H – інверсний код від’ємника
         INC A        ; (A)←F1H – додатковий код від’ємника
         ADD A,B      ; (A)←F8H, (C)=0 свідчить, що в акумуляторі
                      ; знаходиться додатковий код різниці
         CPL A        ; (A)←07H – інверсний код різниці
         INC A        ; (A)←08H – модуль від’ємної різниці
```

Для отримання інверсного коду дво- або багатобайтного числа слід замінити вихідні байти на їх інверсні коди, а для знаходження додаткового – до отриманого інверсного коду прибавити 1.

4.4. Переміщення масивів

За необхідності переміщення масиву елементів внутрішньої пам’яті даних кількістю d1 (із заданою адресою першого байту adr1) в нову область, яка розпочинається з адреси adr2 (внутрішньої пам’яті даних) можна скористатися наступною програмою:

```
Мітка    Команда       Коментар
         MOV R0,#adr1 ; Адреса першого байту вихідного масиву
         MOV R1,#adr2 ; Адреса першого байта нового масиву
         MOV R3,#d1   ; Кількість елементів масиву
M1:      MOV A,@R0    ; Запис в акумулятор байту вихідного масиву
         MOV @R1,A    ; Запис байту у нову область
         INC R0       ; Адреса наступного елемента вихідного масиву
         INC R1       ; Нова адреса елемента
         DJNZ R3,M1   ; Контроль кількості переписаних байтів
```

ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

- Чи можна вважати регістри A, B, SP, DPH, DPL фактично відокремленими від внутрішньої пам’яті даних пристроями?
- Які функції частіше за все виконує 16-бітний регістр DPTR ?
- Чи може стек знаходитися в зовнішній оперативній пам’яті?
- Чи можна занести в стек дані із елемента зовнішньої оперативної пам’яті? Яким чином?
- Чи існують команди XCH A,@R3; MOV @R7,A; MOV 21H,@R5 ?
- Чи існують команди XCH A,B; XCH SP,IP; XCH R1,R3 ?

- Чи існує команда MOV R3,R5 ? Чи можливо за допомогою однієї команди переслати байт із регістра R5 до регістра R3 ?
- Які функції частіше всього виконують регістри R0 і R1 ?
- Яким чином вибирається необхідний банк регістрів загального призначення?
- Чи можна використовувати команду десяткової корекції після дій арифметичного віднімання, множення, ділення?
- Назвіть умови встановлення прапорців перенесення (C), допоміжного перенесення (AC), переповнення (OV) і паритету (P).
- Чи є в системі команд мікроконтролера команди логічного віднімання і ділення?
- Чи однакові дії виконують команди CLR ACC.5 та CLR 0E5H; SETB P3.6 та SETB B6; SETB C та SETB PSW.7 ?
- Чи однакову послідовність машинних кодів можна отримати при трансляції таких пар команд: CLR ACC.5 та CLR 0E5H; SETB P3.6 та SETB B6; SETB C та SETB PSW.7 ?
- Чи впливають команди ORL A,Rn; ANL A,@Ri; XRL A,#d; CLR A на прапорці слова стану програми PSW ? На які саме і чому?
- Чому команда CLR A завжди встановлює прапорець паритету (P) в одиничний стан?
- Чи впливає команда CPL A на прапорець паритету (P) слова стану програми PSW ? Чому саме, адже вміст акумулятора змінюється?
- Які з команд RR A; RRC A; RL A; RLC A можуть впливати на прапорці регістра PSW і на які саме?
- Чим відрізняються команди CLR C і CLR PSW.7 ?
- В яких випадках команда CLR bit може впливати на прапорці регістра PSW ?
- Які з команд CLR RS0; CLR 0D7H; SETB C; SETB 07; SETB ACC.3; CPL C; CPL ACC.5; CPL P3.3 можуть змінювати вміст регістра PSW і чому?
- В яких випадках команда MOV bit,C може впливати на прапорці регістра PSW ?
- Команди умовного переходу використовують абсолютну чи відносну адресу переходу?
- В яких межах може здійснюватися перехід при виконанні команди умовного переходу?
- В яких межах може здійснюватися перехід при виконанні команд: LJMP ad16; AJMP ad11; SJMP rel ?
- Чи існують команди AJMP 0; AJMP 800; AJMP 0FA1H; AJMP 7FFH ? Відповідь поясніть.
- Які умови повинні бути дотримані для переходу за командами JZ rel; JNZ rel; JC rel; JNC rel; JB rel; JNB rel; CJNE A,#d; CJNE Rn,#d; LJMP ad16; AJMP ad11; SJMP rel ?

- Чи може здійснюватись перехід у будь-яку точку програмного простору за командами умовного переходу?
- Чим відрізняються дії команд RET і RET I ?
- Для яких цілей частіше за все використовуються команди DJNZ і CJNE ?
- Виходячи із яких умов АСЕМБЛЕРом визначається конкретний тип команди (LCALL чи ACALL) при використанні узагальненої мнемоніки CALL ?
- Чому на Вашу думку мікроконтролер 8051AH не має команди програмної зупинки HLT ?

Додаток А

ПРИКЛАДИ КОМАНД АСЕМБЛЕРА МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА MCS-51:

А.1. Команди пересилання даних

1) командою MOV A,Rn дані з регістра Rn пересилаються до акумулятора, при цьому вміст регістра Rn не змінюється:

Команда	Дія команди	Коментар
`MOV A,R0`	`(A)←(R0)`	Байт з регістра R0 записується до акумулятора. Вміст R0 не змінюється
`MOV A,R1`	`(A)←(R1)`	Байт з регістра R1 записується до акумулятора. Вміст R1 не змінюється

---

2) командою MOV A,ad дані з елемента внутрішньої оперативної пам’яті за адресою ad пересилаються до акумулятора. Вміст елемента пам’яті за адресою ad при цьому не змінюється. Операнд ad (позначає адресу елемента пам’яті) – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
`MOV A,00H`	`(A)←(00H)`	Байт з елемента пам’яті даних за адресою 00H записується до акумулятора. Вміст елемента пам’яті за адресою 00H не змінюється
`MOV A,32H`	`(A)←(32H)`	Байт з елемента пам’яті даних за адресою 32H записується до акумулятора. Вміст елемента пам’яті за адресою 32H не змінюється
`MOV A,0A0H`	`(A)←(0A0H)`	Байт з елемента пам’яті даних за адресою 0A0H записується до акумулятора. Вміст елемента пам’яті за адресою 0A0H не змінюється. Операнд, який позначає адресу елемента пам’яті, розпочинається з цифри: 0A0H

---

3) командою MOV A,@Ri до акумулятора пересилаються дані з елемента внутрішньої оперативної пам’яті, адреса якого – у регістрі Ri вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW. Уміст регістра Ri та адресованого ним елемента пам’яті при виконанні команди не змінюється:

Команда	Дія команди	Коментар

MOV A,@R0	(A)←((R0))	До акумулятора записується байт з елемента пам’яті даних, адреса якого – в регістрі R0. Вміст регістра R0 та адресованого ним елемента пам’яті не змінюється
MOV A,@R1	(A)←((R1))	До акумулятора записується байт з елемента пам’яті даних, адреса якого – в регістрі R1. Вміст регістра R1 та адресованого ним елемента пам’яті не змінюється

---

4) командою MOV A,#d до акумулятора записується безпосередньо чисельне значення операнда #d. Операнд #d – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
MOV A,#33H	(A)←33H	До акумулятора записується число 33H
MOV A,#0E7H	(A)←0E7H	До акумулятора записується число 0E7H. Операнд, який позначає число, розпочинається з цифри: 0E7H

---

5) командою MOV Rn,ad дані з елемента внутрішньої оперативної пам’яті за адресою ad пересилаються до регістра Rn вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW. Уміст елемента пам’яті за адресою ad при цьому не змінюється. Операнд ad (позначає адресу елемента пам’яті) – чисельний і тому не повинен розпочинатися з цифри:

Команда	Дія команди	Коментар
MOV R0,33H	(R0)←(33H)	До регістра R0 записується байт з елемента пам’яті даних за адресою 33H
MOV R5,0A7H	(R5)←(0A7H)	До регістра R5 записується байт із елемента пам’яті даних за адресою 0A7H. Операнд, який позначає адресу елемента пам’яті, розпочинається з цифри: 0A7H

---

6) командою MOV Rn,#d до регістра Rn записується безпосередньо чисельне значення операнда #d; операнд #d – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
MOV R3,#73H	(R3)←73H	До регістра R3 записується число 73H
MOV R7,#0B9H	(R7)←0B9H	До регістра R7 записується число 0B9H. Операнд, який позначає число, роз-

		починається з цифри: 0B9H

---

7) командою MOV add,ads до елемента пам’яті за адресою add пересилаються дані з елемента пам’яті за адресою ads. Уміст елемента пам’яті за адресою ads при цьому не змінюється. Операнди add і ads, що визначають адреси елементів внутрішньої оперативної пам’яті – джерела та одержувача, – чисельні й тому не повинні розпочинатися з букви:

Команда	***Дія команди***	**Коментар**
MOV 33H,57H	(33H)←(57H)	Байт з елемента пам’яті за адресою 57H записується до елемента пам’яті за адресою 33H. Вміст елемента-джерела за адресою 57H при цьому не змінюється
MOV 37H,0A3H	(37H)←(0A3H)	Байт з елемента пам’яті за адресою 0A3H записується до елемента пам’яті за адресою 37H. Операнди add і ads, які визначають адреси елементів пам’яті, розпочинаються з цифр: 37H і 0A3H
MOV 03H,07H	(03H)←(07H) ≡ (R3)←(R7)	Байт з елемента пам’яті за адресою 07H записується до елемента пам’яті за адресою 03H. Оскільки 07H і 03H – адреси регістрів R7 і R3 відповідно, то байт з регістра R7 записується до регістра R3. Уміст R7 при цьому зберігається

MOV A,0F0H
≡
MOV A,B
	(A)←(0F0H)
	≡
	(A)←(B)
		Байт з елемента пам’яті за адресою 0F0H записується до акумулятора. Оскільки 0F0H – адреса регістра B, то байт даних з регістра B записується до акумулятора. Уміст регістра B при цьому зберігається

MOV 90H,80H
≡
MOV P1,P0
	(90H)←(80H)
	≡
	(P1)←(P0)
		Байт з елемента пам’яті за адресою 80H записується до елемента пам’яті за адресою 90H. Через те, що 80H і 90H – адреси портів P0 і P1 відповідно, то байт з порту P0 записується до порту P1. Уміст порту P0 при цьому зберігається

---

8) командою MOV ad,@Ri до елемента пам’яті за адресою ad пересилаються дані з елемента пам’яті, адреса якого – у регістрі Ri. Уміст регістра Ri та адресованого ним елемента внутрішньої оперативної пам’яті при виконанні команди не змінюється. Операнд ad (позначає адресу елемента пам’яті) – чисельний і тому не повинен розпочинатися з цифри:

Команда	**Дія команди**	**Коментар**
MOV 25H,@R0	(25H)←((R0))	До елемента пам’яті даних за адресою 25H записується байт з елемента

		пам’яті даних, адреса якого – у регістрі R0. Уміст регістра R0 та адресованого ним елемента пам’яті не змінюється
MOV 51H,@R1	(51H)←((R1))	До елемента пам’яті даних за адресою 51H записується байт з елемента пам’яті даних, адреса якого – у регістрі R1. Уміст регістра R1 та адресованого ним елемента пам’яті не змінюється

---

9) командою MOV ad,#d до елемента внутрішньої оперативної пам’яті за адресою ad записується безпосередньо чисельне значення операнда #d; операнд #d – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
MOV 23H,#71H	(23H)←71H	До елемента пам’яті даних за адресою 23H записується число 71H
MOV 57H,#0B6H	(57H)←0B6H	До елемента пам’яті даних за адресою 57H записується число 0B6H. Операнд, який позначає число, розпочинається з цифри: 0B6H

---

10) командою MOV @Ri,#d до елемента внутрішньої оперативної пам’яті, адреса якого – у регістрі Ri, записується безпосередньо чисельне значення операнда #d; операнд #d – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
MOV @R0,#17H	((R0))←17H	До елемента пам’яті даних, адреса якого – у регістрі R0, записується число 17H
MOV @R1,#0B1H	((R1))←0B1H	До елемента пам’яті даних, адреса якого – у регістрі R1, записується число 0B1H. Операнд, який позначає число, розпочинається з цифри: 0B1H

---

11) командою MOV DPTR,#d16 до двобайтового регістра DPTR (з регістрів DPH і DPL) записується безпосередньо чисельне значення двобайтового операнда #d16; операнд #d16 – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
MOV DPTR,#1732H	(DPTR)←1732H (DPH)←17H (DPL)←32H	До регістрової пари DPTR записується число 1732H. Старший байт числа записується до старшого регістра пари DPTR – регістр DPH, а молодший – до молодшого регістра пари – регістр DPL
MOV DPTR,#0B1AFH	(DPTR)←0B1AFH (DPH)←0B1H (DPL)←0AFH	До регістрової пари DPTR записується число 0B1AFH. Операнд, який позначає число, розпочинається з цифри: 0B1AFH

12) командою MOVC A,@A+DPTR до акумулятора пересилаються дані з елемента постійної пам’яті програм, тобто з ПЗП, адреса якого підраховується як сума вмісту акумулятора і двобайтового регістра DPTR. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
MOVC A,@A+DPTR	(A)←((A)+(DPTR))	До акумулятора записується байт з елемента пам’яті за адресою (A)+(DPTR) постійного запам’ятовуючого пристрою. Вміст регістрової пари DPTR не змінюється

13) командою MOVC A,@A+PC до акумулятора пересилаються дані з елемента постійної пам’яті програм (ПЗП), адреса якого підраховується як сума вмісту акумулятора, та програмного лічильника PC. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
MOVC A,@A+PC	(A)←((A)+(PC)) (PC)←(PC)+1	До акумулятора записується байт з елемента пам’яті за адресою (A)++(PC) постійного запам’ятовуючого пристрою. Вміст програмного лічильника інкрементується (збільшується на 1). Мікроконтролер переходить до виконання наступної команди

14) командою MOVX A,@Ri до акумулятора пересилаються дані з елемента зовнішньої оперативної пам’яті даних (ОЗП), адреса якого – у регістрі Ri вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW. Уміст регістра Ri та адресованого ним елемента зовнішньої пам’яті даних при виконанні команди не змінюється:

Команда	Дія команди	Коментар
MOVX A,@R0	(A)←((R0))	До акумулятора записується байт з елемента зовнішньої пам’яті даних, адреса якого – у регістрі R0. Вміст регістра R0 та адресованого ним елемента зовнішньої пам’яті не змінюється
MOVX A,@R1	(A)←((R1))	До акумулятора записується байт з елемента зовнішньої пам’яті даних, адреса якого – у регістрі R1. Уміст регістра R1 та адресованого ним елемента зовнішньої пам’яті не змінюється

15) командою MOVX A,@DPTR до акумулятора пересилаються дані з елемента зовнішньої оперативної пам’яті даних (ОЗП), адреса якого – у регістровій парі DPTR. Уміст регістрової пари DPTR і адресованого нею елемента зовнішньої пам’яті даних при виконанні команди не змінюється. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
MOVX A,@DPTR	(A)←(DPTR)	До акумулятора записується байт з елемента зовнішньої пам’яті даних, адреса якого – у двобайтовому регістрі DPTR. Уміст регістрової пари DPTR не змінюється

---

16) командою PUSH ad до стеку пересилаються дані з елемента внутрішньої оперативної пам’яті за адресою ad. Регістр-покажчик стеку SP інкрементується (збільшується на 1). Уміст елемента пам’яті за адресою ad при цьому не змінюється. Операнд ad (позначає адресу елемента пам’яті) – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
PUSH 36H	((SP))←(36H) (SP)←(SP)+1	Байт з елемента внутрішньої пам’яті даних за адресою 36H записується до стеку. Уміст елемента пам’яті за адресою 00H не змінюється
PUSH 0A5H	((SP))←(0A5H) (SP)←(SP)+1	Байт з елемента внутрішньої пам’яті даних за адресою 0A5H записується до стеку. Операнд, який позначає адресу елемента пам’яті розпочинається з цифри: 0A0H

---

17) командою POP ad дані із стеку пересилаються до елемента внутрішньої оперативної пам’яті за адресою ad. Регістр-покажчик стеку SP декрементується (зменшується на 1). Операнд ad (позначає адресу елемента пам’яті даних) – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
POP 53H	(53H)←((SP)) (SP)←(SP)-1	Байт з стеку записується до елемента внутрішньої оперативної пам’яті даних за адресою 53H
POP 0B7H	(0B7H)←((SP)) (SP)←(SP)-1	Байт з стеку записується до елемента внутрішньої оперативної пам’яті даних за адресою 0B7H. Операнд, який позначає адресу елемента пам’яті, розпочинається з цифри: 0B7H

---

18) командою XCH A,Rn здійснюється обмін даними між акумулятором та регістром Rn вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW:

Команда	Дія команди	Коментар
XCH A,R3	(A)↔(R3)	Здійснюється обмін байтами між акумулятором та регістром R3

XCH A,R7	(A)↔(R7)	Здійснюється обмін байтами між акумулятором та регістром R7

---

19) командою XCH A,ad здійснюється обмін даними між акумулятором та елементом внутрішньої оперативної пам'яті за адресою ad. Операнд ad (позначає адресу елемента пам'яті) – чисельний і тому не повинен розпочинатися з букви:

Команда	Дія команди	Коментар
XCH A,19H	(A)↔(19H)	Виконується обмін байтами між акумулятором і елементом внутрішньої пам'яті даних за адресою 19H
XCH A,0B5H	(A)↔(0B5H)	Виконується обмін байтами між акумулятором та елементом внутрішньої пам'яті даних за адресою 0B5H. Операнд, який позначає адресу елемента пам'яті, розпочинається з цифри: 0B5H

---

20) командою XCH A,@Ri здійснюється обмін даними між акумулятором і елементом внутрішньої оперативної пам'яті, адреса якого – у регістрі Ri вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW:

Команда	Дія команди	Коментар
XCH A,@R0	(A)↔((R0))	Виконується обмін байтами між акумулятором та елементом внутрішньої пам'яті даних, адреса якого – у регістрі R0
XCH A,@R1	(A)↔((R1))	Виконується обмін байтами між акумулятором і елементом внутрішньої пам'яті даних, адреса якого – у регістрі R1

---

21) командою XCHD A,@Ri виконується обмін молодшими тетрадами між акумулятором та елементом внутрішньої оперативної пам'яті, адреса якого – у регістрі Ri вибраного банку регістрів. Вибір одного з чотирьох банків регістрів визначається станом бітів RS0, RS1 слова стану програми PSW:

Команда	Дія команди	Коментар
XCHD A,@R0	(A0-3)↔((R0)0-3)	Виконується обмін молодшими тетрадами між акумулятором та елементом внутрішньої пам'яті даних, адреса якого – у регістрі R0
XCHD A,@R1	(A0-3)↔((R1)0-3)	Виконується обмін молодшими тетрадами між акумулятором та елементом внутрішньої пам'яті даних, адреса якого – у регістрі R1

А.2. Команди арифметичних операцій

22) командами ADD A,Rn ; ADD A,ad ; ADD A,@Ri ; ADD A,#d до акумулятора заноситься арифметична сума вмісту акумулятора і відповідного операнда. Команда модифікує прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ADD A,R7	(A)←(A)+(R7)	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та регістрі R7. Уміст регістра R7 не змінюється
ADD A,27H	(A)←(A)+(27H)	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних з адресою 27H. Уміст елемента пам’яті з адресою 27H не змінюється
ADD A,@R0	(A)←(A)+((R0))	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних, адреса якого - у регістрі R0. Уміст адресованого регістром R0 елемента пам’яті не змінюється
ADD A,#0E4H	(A)←(A)+0E4H	До акумулятора заноситься арифметична сума вмісту акумулятора та числа 0E4H

---

23) командами ADDC A,Rn ; ADDC A,ad ; ADDC A,@Ri ; ADDC A,#d до акумулятора заноситься арифметична сума вмісту акумулятора і відповідного операнда з урахуванням ознаки перенесення C. Команда модифікує прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ADDC A,R2	(A)←(A)+(R2)+(C)	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та регістрі R2 з урахуванням ознаки перенесення. Вміст регістра R2 не змінюється
ADDC A,43H	(A)←(A)+(43H)+(C)	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних з адресою 43H з урахуванням ознаки перенесення. Вміст елемента пам’яті з адресою 43H не змінюється

ADDC A,@R1	(A)←(A)+((R1))+(C)	До акумулятора заноситься арифметична сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних, адреса якого – у регістрі R1 з урахуванням ознаки перенесення. Вміст адресованого регістром R1 елемента пам’яті не змінюється
ADDC A,#94H	(A)←(A)+94H+(C)	До акумулятора заноситься арифметична сума вмісту акумулятора та числа 94H з урахуванням ознаки перенесення

---

24) командами SUBB A,Rn; SUBB A,ad; SUBB A,@Ri; SUBB A,#d до акумулятора заноситься різниця між умістом акумулятора і відповідного операнда з урахуванням ознаки перенесення C. Зменшуване міститься в акумуляторі, а від’ємник – у другому операнді команди. Команда модифікує прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
SUBB A,R6	(A)←(A)-(R6)-(C)	До акумулятора заноситься арифметична різниця між умістом акумулятора та числом, що в регістрі R6 з урахуванням ознаки перенесення. Уміст регістра R6 не змінюється
SUBB A,27H	(A)←(A)-(27H)-(C)	До акумулятора заноситься різниця між вмістом акумулятора та числом, що в елементі внутрішньої пам’яті даних за адресою 43H з урахуванням ознаки перенесення. Вміст елемента пам’яті за адресою 43H не змінюється
SUBB A,@R1	(A)←(A)-((R1))-(C)	До акумулятора заноситься різниця між умістом акумулятора та числом, що в елементі внутрішньої пам’яті даних, адреса якого – в регістрі R1 з урахуванням ознаки перенесення. Уміст адресованого регістром R1 елемента пам’яті не змінюється
SUBB A,#64H	(A)←(A)-64H-(C)	До акумулятора заноситься різниця між умістом акумулятора та числом 64H з урахуванням ознаки перенесення

---

25) командою DAA виконується десяткова корекція числа, що в акумуляторі. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар

DA A	Якщо A3-0>9, або (AC)=1, то (A3-0)← (A3-0)+6; Якщо A7-4>9, або (C)=1, то (A7-4)← (A7-4)+6;	Команда застосовується тільки після виконаного раніше арифметичного додавання двох змінних, кожна з яких задана в двійково-десятковому форматі. Для додавання може використовуватися будь-яка з команд типу ADD або ADDC. Команд DAA не застосовують при відніманні та ін.

---

26) командами INC A; INC Rn; INC ad; INC @Ri; INC DPTR інкрементується (збільшується на 1) вміст регістра або елемента пам’яті, що позначається відповідним операндом

Команда	Дія команди	Коментар
INC A	(A)←(A)+1	Інкрементується уміст акумулятора
INC R4	(R4)←(R4)+1	Інкрементується уміст регістра R4 обраного банку регістрів
INC 28H	(28H)←(28H)+1	Інкрементується уміст елемента внутрішньої оперативної пам’яті даних за адресою 28H
INC @R1	((R1))←((R1))+1	Інкрементується уміст елемента внутрішньої пам’яті даних, адреса якого – у регістрі R1 обраного банку регістрів
INC DPTR	(DPTR)←(DPTR)+1	Інкрементується значення двобайтового регістра DPTR

---

27) командами DEC A; DEC Rn; DEC ad; DEC @Ri декрементується (зменшується на 1) вміст регістра або елемента пам’яті, що позначається відповідним операндом

Команда	Дія команди	Коментар
DEC A	(A)←(A)-1	Декрементується уміст акумулятора
DEC R2	(R2)←(R2)-1	Декрементується уміст регістра R2 обраного банку регістрів
DEC 36H	(36H)←(36H)-1	Декрементується уміст елемента внутрішньої оперативної пам’яті даних з адресою 36H
DEC @R0	((R0))←((R0))-1	Декрементується уміст елемента внутрішньої пам’яті даних, адреса якого – в регістрі R0 обраного банку регістрів

---

28) командою MUL AB виконується арифметичне множення 8-бітових цілих беззнакових чисел, що зберігаються в акумуляторі та регістрі B. Старший байт 16-бітового добутку вміщується до регістра B, а молодший – до акумулятора. Якщо

добуток більший за 0FFH (255D), то встановлюється прапорець переповнення OV слова стану програми PSW. Прапорець перенесення завжди скидається. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
MUL AB	(BA)←(A)×(B)	До регістрової пари BA записується добуток чисел, що зберігаються в регістрах A і B

---

29) командою DIV AB виконується ділення 8-бітових цілих беззнакових чисел, що зберігаються в акумуляторі (ділене) та регістрі B (дільник). Ціла частина результату вміщується до акумулятора, а дрібна (залишок) – до регістра B. Прапорці перенесення (C) та переповнення (OV) скидаються. Якщо (A) < (B), то прапорець додаткового перенесення не скидається. Прапорець перенесення скидається завжди. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
DIV	(A).(B)←(A)/(B)	До регістрової пари AB записується арифметична частка чисел, що зберігаються в регістрах A і B

A.3. Команди логічних операцій

30) командами ANL A,Rn; ANL A,ad; ANL A,@Ri; ANL A,#d до акумулятора заноситься побітовий логічний добуток умісу акумулятора і відповідного операнда. Команда модифікує прапорець паритету слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ANL A,R7	(A)←(A)∧(R7)	До акумулятора заноситься логічний добуток чисел, що зберігаються в акумуляторі та регістрі R7. Вміст регістра R7 не змінюється
ANL A,27H	(A)←(A)∧(27H)	До акумулятора заноситься логічний добуток чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних за адресою 27H. Уміст елемента пам’яті за адресою 27H не змінюється
ANL A,@R0	(A)←(A)∧((R0))	До акумулятора заноситься логічний добуток чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних, адреса якого – у регістрі R0. Уміст адресованого регістром R0 елемента пам’яті не змінюється
ANL A,#0E4H	(A)←(A)∧0E4H	До акумулятора заноситься логічний добуток умісту акумулятора та числа

		0E4H

---

31) командами ANL ad,A; ANL ad,#d до елемента пам’яті даних з адресою ad заноситься побітовий логічний добуток умісту цього елемента та відповідного операнда. Команда не впливає на прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ANL 31,A	(31H)←(31H)∧(A)	До елемента пам’яті даних з адресою 31H заноситься логічний добуток чисел, що зберігаються в цьому елементі та акумуляторі. Уміст акумулятора не змінюється
ANL 15H,#0B7H	(15H)←(15H)∧0B7H	До елемента пам’яті даних з адресою 15H заноситься логічний добуток умісту цього елемента та числа 0B7H

---

32) командами ORL A,Rn; ORL A,ad; ORL A,@Ri; ORL A,#d до акумулятора заноситься побітова логічна сума вмісту акумулятора та відповідного операнда. Команда модифікує прапорець паритету слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ORL A,R3	(A)←(A)∨(R3)	До акумулятора заноситься логічна сума чисел, що зберігаються в акумуляторі та регістрі R3. Уміст регістра R3 не змінюється
ORL A,21H	(A)←(A)∨(21H)	До акумулятора заноситься логічна сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних за адресою 21H. Уміст елемента пам’яті за адресою 21H не змінюється
ORL A,@R1	(A)←(A)∨((R1))	До акумулятора заноситься логічна сума чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних, адреса якого – у регістрі R1. Уміст адресованого регістром R1 елемента пам’яті не змінюється
ORL A,#0ECH	(A)←(A)∨0ECH	До акумулятора заноситься логічна сума умісту акумулятора та числа 0ECH

---

33) командами ORL ad,A; ORL ad,#d до елемента пам’яті даних за адресою ad заноситься побітова логічна сума умісту цього елемента та відповідного операнда. Команда не впливає на прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
ORL 16,A	(16H)←(16H)∨(A)	До елемента пам’яті даних за адресою 16H заноситься логічна сума чисел, що зберігаються в цьому елементі та акумуляторі. Уміст акумулятора не змінюється
ORL 20H,#0D3H	(20H)←(20H)∨0D3H	До елемента пам’яті даних за адресою 20H заноситься логічна сума умісту цього елемента та числа 0D3H

---

34) командами XRL A,Rn; XRL A,ad; XRL A,@Ri; XRL A,#d до акумулятора заноситься побітова логічна сума за модулем 2 чисел умісту акумулятора та відповідного операнда. Команда модифікує прапорець паритету слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
XRL A,R6	(A)←(A)⊕(R6)	До акумулятора заноситься логічна сума за модулем 2 чисел, що зберігаються в акумуляторі та регістрі R6. Уміст регістра R6 не змінюється
XRL A,39H	(A)←(A)⊕(39H)	До акумулятора заноситься логічна сума за модулем 2 чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних за адресою 39H. Уміст елемента пам’яті за адресою 39H не змінюється
XRL A,@R1	(A)←(A)⊕((R1))	До акумулятора заноситься логічна сума за модулем 2 чисел, що зберігаються в акумуляторі та елементі внутрішньої пам’яті даних, адреса якого – у регістрі R1. Уміст адресованого регістром R1 елемента пам’яті не змінюється
XRL A,#0F6H	(A)←(A)⊕0F6H	До акумулятора заноситься логічна сума за модулем 2 чисел вмісту акумулятора та числа 0F6H

---

35) командами XRL ad,A; XRL ad,#d до елемента пам’яті даних з адресою ad заноситься побітова логічна сума за модулем 2 чисел вмісту цього елемента та відповідного операнда. Команда не впливає на прапорці слова стану програми PSW. Позначення чисельних операндів не повинні розпочинатися з букви:

Команда	Дія команди	Коментар
XRL 12,A	(12H)←(12H)⊕(A)	До елемента пам’яті даних за адресою 12H заноситься логічна сума за модулем 2 чисел, що зберігаються в цьому елементі та акумуляторі. Уміст акумулятора не змінюється

XRL 51H,#0B4H	(51H)←(51H)⊕0B4H	До елемента пам'яті даних за адресою 51H заноситься логічна сума за модулем 2 чисел вмісту цього елемента та числа 0B4H

---

36) командою CLR A всі розряди акумулятора скидаються в нульовий стан, тобто до акумулятора записується число 0. Команда встановлює прапорець паритету слова стану програми PSW в одиничний стан. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
CLR A	(A)←0	До акумулятора заноситься число 0. Прапорець паритету встановлюється в одиничний стан

---

37) командою CPL A вміст акумулятора інвертується. Команда не впливає на прапорці слова стану програми PSW. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
CPL A	(A)←NOT (A)	Уміст акумулятора інвертується, стан прапорців PSW не змінюється

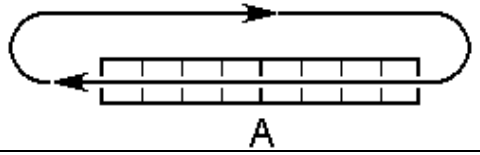
---

38) командою SWAP A здійснюється обмін між молодшими чотирма і старшими чотирма бітами акумулятора (між молодшою і старшою тетрадами). Команда не впливає на прапорці слова стану програми PSW. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
SWAP A	(A3-0)↔(A7-4)	Дія команди рівнозначна чотирибітовому циклічному зсуву вмісту акумулятора

---

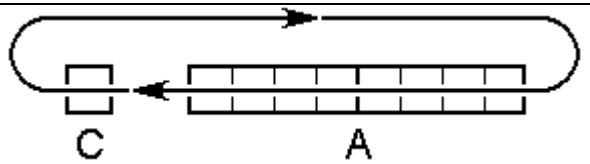
39) командою RL A виконується циклічний зсув ліворуч на один біт числа, що в акумуляторі. Біт A_7 потрапляє на місце біта A_0. Команда не впливає на прапорці слова стану програми PSW. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
RL A	(A7-1)←(A6-0) (A0)←(A7)	

		Зсув умісту акумулятора ліворуч на один біт

---

40) командою RLC A виконується циклічний зсув ліворуч на один біт через прапорець перенесення (C) числа, що в акумуляторі. Вміст біта A_7 потрапляє до прапорця перенесення, а вміст прапорця перенесення потрапляє до біта A_0 акумулятора. На інші прапорці команда не впливає:

Команда	Дія команди	Коментар
RLC A	(A7-1)←(A6-0) (C)←(A7) (A0)←(C)	 Зсув умісту акумулятора ліворуч на один біт через прапорець (C)

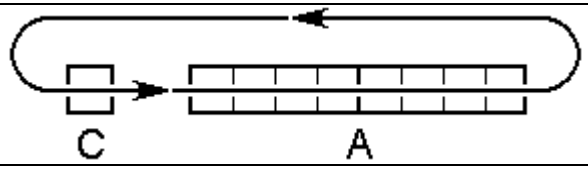
---

41) командою RR A виконується циклічний зсув праворуч на один біт числа, що в акумуляторі. Біт A_0 потрапляє на місце біта A_7. Команда не впливає на прапорці слова стану програми PSW. Команда не має змінних операндів і тому записується однозначно:

Команда	Дія команди	Коментар
RR A	(A6-0)←(A7-1) (A7)←(A0)	 Зсув умісту акумулятора праворуч на один біт

---

42) командою RRC A виконується циклічний зсув праворуч на один біт через прапорець перенесення (C) числа, що в акумуляторі. Уміст біта A_0 потрапляє до прапорця перенесення, а уміст прапорця перенесення потрапляє до біта A_7 акумулятора. На інші прапорці команда не впливає:

Команда	Дія команди	Коментар
RRC A	(A6-0)←(A7-1) (C)←(A0) (A7)←(C)	

		Зсув умісту акумулятора праворуч на один біт через прапорець (C)

A.4. Команди операцій над бітами

43) командами CLR C; CLR bit скидається в нульовий стан відповідно прапорець ознаки перенесення (C) або біт, що прямо адресується командою:

Команда	Дія команди	Коментар
`CLR C`	`(C)←0`	Прапорець ознаки перенесення (C) скидається в нульовий стан. На інші прапорці регістра PSW команда не впливає

`CLR PSW.7`	`(PSW7)←0`
	≡
	`(C)←0`
		Дія команди аналогічна до попередньої, оскільки біт PSW.7 – це прапорець ознаки перенесення. На відміну від вищенаведеної команди тут використовується пряма адресація біта (C)

`CLR ACC.3`	`(A3)←0`	Біт A_3 акумулятора скидається в нульовий стан. Команда модифікує прапорець ознаки паритету (P) через те, що модифікується вміст акумулятора

`CLR 20.5`
≡
`CLR 05`
	`(20H5)←0`	Біт D_5 елемента пам'яті даних з адресою 20H скидається в нульовий стан. Пряма адреса цього біта – 05H. Команда не впливає на прапорці регістра стану програми PSW

---

44) командами SETB C; SETB bit встановлюється в одиничний стан відповідно прапорець ознаки перенесення (C) або біт, що прямо адресується командою:

Команда	Дія команди	Коментар
`SETB C`	`(C)←1`	Прапорець ознаки перенесення (C) встановлюється в одиничний стан. На інші прапорці регістра PSW команда не впливає

`SETB PSW.7`	`(PSW7)←1`
	≡
	`(C)←1`
		Дія команди аналогічна до попередньої, оскільки біт PSW.7 – це прапорець ознаки перенесення. На відміну від вищенаведеної команди тут використовується пряма адресація біта (C)

`SETB ACC.6`	`(A6)←1`	Біт A6 акумулятора встановлюється в одиничний стан. Команда модифікує прапорець ознаки паритету (P), адже модифікується уміст акумулятора. На інші прапорці регістра PSW команда не впливає

SETB 20.3 ≡ SETB 03	(20H3)←1	Біт D_3 елемента пам'яті даних з адресою 20H встановлюється в одиничний стан. Пряма адреса цього біта – 03H. Команда не впливає на прапорці слова стану програми

---

45) командами CPL C; CPL bit інвертується відповідно прапорець ознаки перенесення або біт, що прямо адресується командою:

Команда	Дія команди	Коментар
CPL C	(C)←NOT(C)	Прапорець ознаки перенесення (C) інвертується. На інші прапорці регістра PSW команда не впливає
CPL 23.7 ≡ CPL 1FH	(237)←NOT (23H7)	Інвертується біт D_7 елемента пам'яті даних з адресою 23H. Пряма адреса цього біта – 1FH. Команда не впливає на прапорці слова стану програми

---

46) командою ANL C,bit визначається логічний добуток умісту прапорця ознаки перенесення (C) і біта, що адресується другим операндом. Добуток записується до прапорця (C). На решту прапорців слова стану програми PSW команда не впливає. Значення біта, який адресується другим операндом, не змінюється:

Команда	Дія команди	Коментар
ANL C,ACC.4	(C)←(C)∧(A4)	До прапорця (C) заноситься логічний добуток умісту (C) і біта A_4 акумулятора. На решту прапорців регістра PSW команда не впливає
ANL C,E0.4 ≡ ANL C,0E4H	(C)←(C)∧(0E0H4)	Дія цих команд аналогічна до дії попередньої команди, оскільки біт D_4 елемента пам'яті за адресою 0E0H – це і є біт A_4 акумулятора. 0E4H – пряма адреса цього біта
ANL C,20.3 ≡ ANL C,03	(C)←(C)∧(20H3)	До прапорця (C) заноситься логічний добуток умісту (C) і біта D_3 елемента пам'яті даних за адресою 20H. Пряма адреса цього біта – 03H. На решту прапорців регістра PSW команда не впливає

---

47) командою ANL C,/bit визначається логічний добуток умісту прапорця ознаки перенесення (C) та інверсного значення біта, що адресується другим операндом. Добуток записується до прапорця (C). На інші прапорці слова стану програми PSW команда не впливає. Значення біта, який адресується другим операндом, не змінюється:

Команда	Дія команди	Коментар
ANL C,/ACC.4	(C)←(C)∧NOT (A4)	До прапорця (C) заноситься логічний добуток умісту (C) та інверсного значення біта A_4 акумулятора. На решту прапорців регістра PSW команда не впливає
ANL C,/20.3 ≡ ANL C,/03	(C)←(C)∧NOT (20H3)	До прапорця (C) заноситься логічний добуток умісту (C) та інверсного значення біта D_3 елемента пам'яті даних за адресою 20H. Пряма адреса цього біта – 03H. На решту прапорців регістра PSW команда не впливає

48) командами ORL C,bit; ORL C,/bit визначається логічна сума вмісту прапорця ознаки перенесення (C) та значення другого операнда. Результат записується до прапорця (C). На решту прапорців слова стану програми PSW команда не впливає. Значення біта, який адресується другим операндом, не змінюється:

Команда	Дія команди	Коментар
ORL C,ACC.4	(C)←(C)∨(A4)	До прапорця (C) заноситься логічна сума умісту (C) і біта A_4 акумулятора. На решту прапорців регістра PSW команда не впливає
ORL C,EO.4 ≡ ORL C,0E4H	(C)←(C)∨(0E0H4)	Дія цих команд аналогічна до дії попередньої команди, оскільки біт D_4 елемента пам'яті за адресою 0E0H – це і є біт A_4 акумулятора. 0E4H –пряма адреса цього біта
ORL C,20.3 ≡ ORL C,03	(C)←(C)∨(20H3)	До прапорця (C) заноситься логічна сума вмісту (C) та біта D_3 елемента пам'яті даних за адресою 20H. Пряма адреса цього біта – 03H. На решту прапорців регістра PSW команда не впливає
ORL C,/ACC.4	(C)←(C)∨NOT (A4)	До прапорця (C) заноситься логічна сума умісту (C) та інверсного значення біта A_4 акумулятора. На решту прапорців регістра PSW команда не впливає
ORL C,/20.3 ≡ ORL C,/03	(C)←(C)∨NOT (20H3)	До прапорця (C) заноситься логічна сума вмісту (C) та інверсного значення біта D_3 елемента пам'яті даних з адресою 20H. Пряма адреса цього біта – 03H. На решту прапорців регістра PSW команда не впливає

49) командою MOV C,bit до прапорця ознаки перенесення (C) записується значення біта, що адресується другим операндом. На решту прапорців регістра PSW команда не впливає. Значення біта, який адресується другим операндом, не змінюється:

Команда	Дія команди	Коментар

MOV C,P3.1	(C)←(P31)	До прапорця (C) заноситься значення біта $P3_1$ (біт D_1 порту P3)

MOV C,2E.5
≡
MOV C,75H
	(C)←(2EH5)	До прапорця (C) заноситься значення біта D_5 елемента пам'яті даних за адресою 2EH. Пряма адреса цього біта – 75H

---

50) командою MOV bit,C до біта, що адресується першим операндом команди, записується значення прапорця ознаки перенесення (C) слова стану програми PSW. Якщо адресований першим операндом біт не є бітом акумулятора або регістра PSW, то команда не впливає на ознаки слова стану програми:

Команда	Дія команди	Коментар
MOV P1.6,C	(P16)←(C)	До біта D_6 порту P1 записується значення ознаки перенесення (C)

MOV 27.5,C
≡
MOV C,3DH
	(27H5)←(C)	До біта D_5 елемента пам'яті даних з адресою 27H заноситься значення ознаки перенесення (C). Пряма адреса цього біта – 3DH

A.5. Команди передачі керування

51) командами LJMP ad16; AJMP ad11; SJMP rel; JMP @A+DPTR здійснюється безумовний перехід за адресою пам'яті програм, визначеною операндом команди:

Команда	Дія команди	Коментар
LJMP 37A4H	(PC)←37A4H	Безумовний перехід за адресою 37A4H
AJMP 3C6H	(PC10-0)←3C6H, (PC15-11)=const	Безумовний перехід за адресою X3C6H у межах сторінки X розміром 2 Кбайти. Старші чотири біти лічильника команд не змінюються і визначають сторінку X пам'яті програм. Значення операнда, що задається в тілі команди, не повинне перевищувати 7FFH
SJMP 05H	(PC)←(PC)$\pm$|rel| ≡ (PC7-0)←05H, (PC15-8)=const	Безумовний перехід за адресою XX05H у межах сторінки XX розміром 256 байтів. Уміст старшого байта лічильника команд не змінюється і визначає сторінку XX пам'яті програм. Максимальне можливе значення операнда, що задається в тілі команди, залежить від поточного значення вмісту PC; $-128 \le rel \le 127$

JMP @A+DPTR	(PC)←((A))+(DPTR)	Адреса безумовного переходу обчислюється як сума вмісту елемента пам'яті даних, що адресується акумулятором, і двобайтового регістра DPTR. Команда не має змінних операндів і записується однозначно

---

52) командами LCALL ad16; ACALL ad11 викликають підпрограму за адресою, визначеною операндом команди. До стекової пам'яті записується повна 16-бітова адреса повернення:

Команда	Дія команди	Коментар
LCALL 6A94H	(PC)←6A94H, (SP)←(SP)+2	Виклик підпрограми за адресою 6A94H. Уміст регістра-покажчика стека збільшується на 2
ACALL 20DH	(PC10-0)←20DH, (PC15-11)=const, (SP)←(SP)+2	Виклик підпрограми за адресою X20DH у межах сторінки X розміром 2 Кбайти. Старші чотири біти лічильника команд не змінюються і визначають сторінку X пам'яті програм. Значення операнда, що задається в тілі команди, не повинне перевищувати 7FFH

---

53) командами RET; RET I здійснюється повернення з підпрограми. До програмного лічильника із стеку заноситься адреса повернення. Команди не впливають на ознаки регістра PSW:

Команда	Дія команди	Коментар
RET	(PC15-8)←((SP)) (PC7-0)←((SP)-1) (SP)←(SP)-2	Повернення з підпрограми. Команда використовується у парі з командою виклику підпрограми LCALL або ACALL. Виконання основної програми триває за адресою команди після LCALL або ACALL
RET I	(PC15-8)←((SP)) (PC7-0)←((SP)-1) (SP)←(SP)-2	Дія команди аналогічна до дії вищенаведеної команди. Команда дозволяє переривання того рівня, який був щойно обслугований. Виконання основної програми продовжується з команди, наступної після тієї, на якій був виявлений запит на переривання

Додаток Б

Індивідуальні завдання низького рівня складності

Для отримання оцінки “задовільно” слід розробити програму, в якій в регістр R0 банку 0 заноситься сума чисел 1 (для групи АУ) або 2 (Аур), року вступу до вищого навчального закладу (останні дві цифри) та номеру групи. В елемент внутрішньої пам'яті даних (адреса якого в R0) програма заносить порядковий номер k студента у журналі групи і за допомогою непрямої адресації забезпечує обмін кодами між акумулятором і цим елементом пам’яті. Далі послідовно визначає їх суму, здійснює обмін тетрадами акумулятора, збільшує результат на 1 і записує його в елемент пам’яті даних за адресою 2АН (всі дані програми – в шістнадцятковій системі числення).

Індивідуальні завдання середнього рівня складності

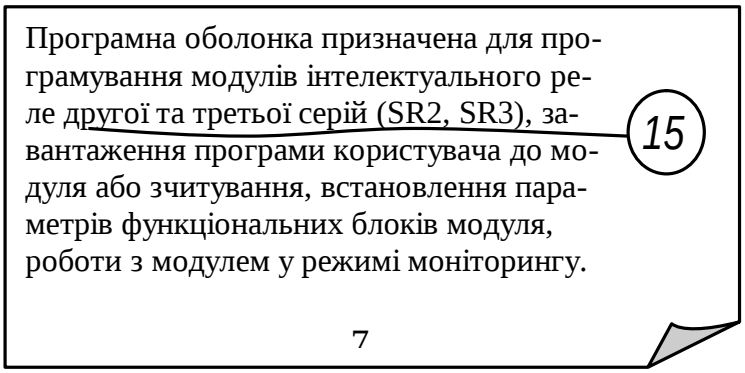
Для отримання оцінки “добре” необхідно додатково до завдання низького рівня складності розробити і налагодити програму обчислення молодшого байта арифметичної суми елементів масиву довжиною 5+k. Початкова адреса масиву 40 – k (для групи АУ) або 20 + k (для групи АУр). Знайдений байт повинен бути записаний до елемента пам’яті, номер якого збігається з номером k. Студенти, які виконують завдання з парним номером варіанта k повинні використовувати зовнішню пам’ять даних, а з непарним – внутрішню. Номер індивідуального завдання k слід вибирати за номером у журналі групи.

Індивідуальні завдання середнього рівня складності

Для отримання оцінки “відмінно” – скласти і налагодити програму сортування елементів масиву довжиною 5+k з довільними параметрами. У випадку парного номера k індивідуального завдання масив слід сортувати за збільшенням елементів, а непарного – за їх зменшенням. Окрім цього, за допомогою програми сформувати масив, який містить порядкові номери елементів вихідного масиву після його сортування. Номер індивідуального завдання k слід вибирати за номером студента у журналі групи. Метод сортування задає викладач.

Індивідуальні завдання до самостійної роботи

Завдання полягає у опрацюванні методичних вказівок з олівцем у руках. При виконанні завдання необхідно відшукувати та підкреслювати фрагменти тексту або позначати на ілюстраціях графічні елементи, про які йдеться у запитаннях. Поруч, на березі аркуша методичних вказівок, у кружечку слід проставити номер запитання, наприклад:



Також у графі «Посилання» треба зазначити номер сторінки, на якій знайдено відповідь на запитання.

Увага! Виконання наведених завдань дозволяється лише у власному примірнику методичних вказівок. Забороняється проставляти будь-які позначки або виконувати креслення у методичних вказівках бібліотечного фонду.

Індивідуальні варіанти завдань до самостійної роботи

№ зап.	№ вар.	Завдання або запитання	Посилання
1	1/1	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV R3,A.	
2	2/2	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди XCH A,R7.	
3	3/3	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV @R0,A.	
4	4/4	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди ADD A,#3.	
5	5/5	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди SUBB A,#7.	
6	6/6	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV R5,15.	
7	7/7	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди PUSH 19.	
8	8/8	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV 15,#3.	

9	9/9	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV 19,#7.	
10	10/10	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV 15,33.	
11	11/11	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV 33,15.	
12	12/12	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди ANL C,ACC.7	
13	13/13	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди ORL C,ACC.6	
14	14/14	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди MOV C,ACC.3	
15	15/15	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди JZ 32.	
16	1/16	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди SJMP 15.	
17	2/17	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди AJMP 2314.	
18	3/18	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди ACALL 3715.	
19	4/19	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди DJNZ 15,37.	
20	5/20	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди CJNE A,37,35.	
21	6/1	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди CJNE R5,#15,76.	
22	7/2	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди JNB 27.3,17.	
23	8/3	Позначте на рисунку, що відображає типи команд (інструкцій) асемблера, тип команди JBC 27.5,19.	
24	9/4	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 21.3	
25	10/5	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 23.7	
26	11/6	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 25.1	
27	12/7	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 27.5	
28	13/8	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 29.0	
29	14/9	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2A.3	
30	15/10	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2C.5	
31	1/11	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2B.7	
32	2/12	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2D.1	
33	3/13	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2F.3	
34	4/14	Позначте на рисунку, що відображує структуру внутрішньої пам’яті даних, біт 2E.0	
35	5/15	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт ACC.3	

36	6/16	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт P3.7	
37	7/17	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт P0.5	
38	8/18	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт P2.0	
39	9/19	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт P1.4	
40	10/20	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт TCON.5	
41	11/1	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт SCON.1	
42	12/2	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт PSW.7	
43	13/3	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт IE.0	
44	14/4	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт IP.4	
45	15/5	Позначте на рисунку, що відображує структуру сегменту регістрів спеціальних функцій, біт B.5	
46	1/6	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що усі регістри мікроконтролера одночасно є елементами внутрішньої пам’яті даних.	
47	2/7	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що система команд мікроконтролера має у своєму складі команди роботи зі стеком.	
48	3/8	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що однією командою MOV можна переслати дані із будь-якого одного елемента внутрішньої пам’яті даних до будь-якого іншого.	
49	4/9	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що однією командою MOV можна переслати дані із будь-якого одного регістра до будь-якого іншого.	
50	5/10	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що однією командою MOV можна переслати дані із будь-якого елемента внутрішньої пам’яті даних до будь-якого регістра.	
51	6/11	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що однією командою MOV можна переслати дані із будь-якого регістра до будь-якого елемента внутрішньої пам’яті даних.	
52	7/12	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про умови встановлення прапорця ознаки перенесення.	
53	8/13	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про умови встановлення прапорця ознаки допоміжного перенесення.	
54	9/14	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про умови встановлення прапорця ознаки переповнення.	
55	10/15	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про умови встановлення прапорця ознаки паритету.	
56	11/16	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про можливі межі переходу при виконанні команди AJMP.	
57	12/17	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про можливі межі переходу при виконанні команди SJMP.	
58	13/18	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про можливі межі переходу при виконанні команд умовного переходу.	

59	14/19	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що команда DJNZ найбільш придатна для організації циклів із заданою кількістю повторів..	
60	15/20	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому йдеться про те, що команда CJNE найбільш придатна для реалізації процедур очікування зовнішніх подій.	
61	1/1	Знайдіть та підкресліть у методичних вказівках фрагмент тексту, у якому пояснюється різниця між командами RET та RET I.	
62	2/2	Знайдіть та підкресліть серед команд пересилання даних у додатку А будь-яку однобайтову команду.	
63	3/3	Знайдіть та підкресліть серед команд пересилання даних у додатку А будь-яку двобайтову команду.	
64	4/4	Знайдіть та підкресліть серед команд пересилання даних у додатку А будь-яку трьохбайтову команду.	
65	5/5	Знайдіть та підкресліть серед команд арифметичних операцій у додатку А будь-яку однобайтову команду.	
66	6/6	Знайдіть та підкресліть серед команд арифметичних операцій у додатку А будь-яку двобайтову команду.	
67	7/7	Знайдіть та підкресліть серед команд логічних операцій у додатку А будь-яку однобайтову команду.	
68	8/8	Знайдіть та підкресліть серед команд логічних операцій у додатку А будь-яку двобайтову команду.	
69	9/9	Знайдіть та підкресліть серед команд операцій над бітами у додатку А будь-яку однобайтову команду.	
70	10/10	Знайдіть та підкресліть серед команд операцій над бітами у додатку А будь-яку двобайтову команду.	
71	11/11	Знайдіть та підкресліть у додатку А будь-яку команду типу 1.	
72	12/12	Знайдіть та підкресліть у додатку А будь-яку команду типу 3.	
73	13/13	Знайдіть та підкресліть у додатку А будь-яку команду типу 2.	
74	14/14	Знайдіть та підкресліть у додатку А будь-яку команду типу 5.	
75	15/15	Знайдіть та підкресліть у додатку А будь-яку команду типу 4.	
76	1/16	Знайдіть та підкресліть у додатку А будь-яку команду типу 7.	
77	2/17	Знайдіть та підкресліть у додатку А будь-яку команду типу 6.	
78	3/18	Знайдіть та підкресліть у додатку А будь-яку команду типу 9.	
79	4/19	Знайдіть та підкресліть у додатку А будь-яку команду типу 8.	
80	5/20	Знайдіть та підкресліть у додатку А будь-яку команду типу 11.	
81	6/1	Знайдіть та підкресліть у додатку А будь-яку команду типу 10.	
82	7/2	Знайдіть та підкресліть у додатку А будь-яку команду типу 12.	
83	8/3	Знайдіть та підкресліть у додатку А будь-яку команду, що працює із регістром-розширювачем акумулятора.	
84	9/4	Знайдіть та підкресліть у додатку А будь-яку команду, що працює із стеком.	
85	10/5	Знайдіть та підкресліть у додатку А будь-яку команду, що працює із регістром-покажчиком даних.	
86	11/6	Знайдіть та підкресліть у додатку А будь-яку команду, що працює із регістром-програмним лічильником.	
87	12/7	Знайдіть та підкресліть у додатку А будь-яку команду, що використовує пряму адресацію операндів.	
88	13/8	Знайдіть та підкресліть у додатку А будь-яку команду, що використовує непряму (посередню) адресацію операндів.	
89	14/9	Знайдіть та підкресліть у додатку А будь-яку команду, що використовує регістр загального призначення для непрямої (посередньої) адресації операндів.	

90	15/10	Знайдіть та підкресліть у додатку А будь-яку команду, що використовує регістр спеціальних функцій для непрямої (посередньої) адресації операндів.	
91	1/11	Знайдіть та підкресліть у додатку А будь-яку команду, що впливає на прапорець ознаки перенесення.	
92	2/12	Знайдіть та підкресліть у додатку А будь-яку команду, що впливає на прапорець ознаки допоміжного перенесення.	
93	3/13	Знайдіть та підкресліть у додатку А будь-яку команду, що впливає на прапорець ознаки паритету.	
94	4/14	Знайдіть у тексті методичних вказівок фрагмент тексту, у якому йдеться про вкладені підпрограми	
95	5/15	Знайдіть у тексті методичних вказівок фрагмент тексту, у якому йдеться про способи організації затримок часу	
96	6/16	У формулі для розрахунку тривалості підпрограми програмної затримки часу позначте частоту синхронізації мфкроконтролера	
97	7/17	Знайдіть у тексті методичних вказівок фрагмент тексту, у якому йдеться про те, що операцію віднімання можна замінити додаванням додаткового коду	
98	8/18	Знайдіть у тексті методичних вказівок фрагмент тексту, у якому йдеться про спосіб отримання додаткового коду багатобайтних чисел	
99	9/19	У підпрограмі затримки часу TIME2 позначете команди вкладеного циклу	
100	10/20	У підпрограмі затримки часу TIME2 позначете команду, якою організується вихід із внутрішнього циклу	
101	11/1	У підпрограмі затримки часу TIME2 позначете команду, якою організується вихід із зовнішнього циклу	
102	12/2	У програмі переміщення масиву позначте команду зчитування елемента вихідного масиву	
103	13/3	У програмі переміщення масиву позначте команду запису елемента вихідного масиву у нову область	
104	14/4	У програмі переміщення масиву позначте команду контролю кількості перезаписаних байтів	
105	15/5	У підпрограмі переміщення масиву позначте команди, що використовують непряму адресацію	

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс. – М.: ЭКОМ, 1999. – 400 с.
2. Встраиваемый микроконтроллер 8XC251SB: Руководство пользователя: Пер. с англ. – К.: Квазар-микро, 1995. – 418 с.
3. Гуртовцев А.Л., Гудыменко С.В. Программы для микропроцессоров: Справ. пособие. – Минск: Вышейш. шк., 1989. – 352 с.
4. Долгий А. Разработка и отладка устройств на МК // Радио. – 2001. – №№ 5-10. – С. 14-16, с. 17-19.
5. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропроцессорах. – М.: Высш. шк., 1991. – 303 с.
6. Каспер Э. Программирование на языке Ассемблере для микроконтроллеров семейства i8051. – М.: Горячая линия – Телеком, 2003. – 191 с.
7. Николайчук О.И. x51 – совместимые микроконтроллеры фирмы Cygnal. – М.: ИД СКИМЕН, 2002. – 230 с.
8. Однокристальные микроЭВМ: Справочник / А.В. Бобрыкин, Г.П. Липовецкий, Г.В. Литвинский и др. – М.: МИКАП, 1994. – 400 с.
9. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Интернет / Под ред. И.В. Коршуна; Составление, пер. с англ. Б.Б. Горбунова. – М.: Аким, 1998. – 272 с.
10. Фрунзе А.В., Фрунзе А.А. Микроконтроллеры? Это же просто! Т. 3. – М.: ИД СКИМЕН, 2003. – 224 с
11. Lopez S. Using the 87C51GB. AB-44 Application Brief. March 1991. Order Number: 270957-001. Intel Corporation, 1996, p. 1-18, A.1-A.7, B.1.
12. Matthew Chapman. THE FINAL WORD ON THE 8051, 1994, 255 p.
13. MCS®-51 and MCS®-96 Packaging Information. November 1994. Order Number: 272118-001. Intel Corporation, 1995, 16 p.
14. Schue R. 32-Bit Math Routines for the 8051. AB-40 Application Brief. October 1992. Order Number: 270530-002. Intel Corporation, 1996, 8 p.
15. 8XC251SA/SB/SP/SQ High-Performance CHMOS Microcontroller. Product Preview. December 1995. Order Number: 272783-002. Intel Corporation, 1995, 35 p.
16. 8XC51GB CHMOS Single-Chip 8-Bit Microcontroller. Preliminary. November 1994. Order Number: 272337-002. Intel Corporation, 1995, 22 p.

Упорядники:
КИРИЧЕНКО Віталій Іванович
ЯЛАНСЬКИЙ Олексій Анатолійович
КОВШОВ Кирил Валерійович

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ МПП-2 “СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА 8051АН СІМЕЙСТВА MCS-51. ТИПОВІ ПРОГРАМНІ КОНСТРУКЦІЇ”, ІНДИВІДУАЛЬНИХ ЗАВДАНЬ ТА САМОСТІЙНОЇ РОБОТИ З ПРОФЕСІЙНО-ОРІЄНТОВАНОЇ ДИСЦИПЛІНИ “МІКРОПРОЦЕСОРНІ ПРИСТРОЇ”

для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму “Електромеханіка”

Редакційно-видавничий комплекс
Друкується у редакційній обробці упорядників

Підписано до друку . Формат 30×42/4.
Папір офсет. Ризографія. Умовн. друк. арк. .
Обліково-видавн. арк. . Тираж 100 прим. Зам. №

НГУ
49027, м. Дніпропетровськ-27, просп. К. Маркса, 19.