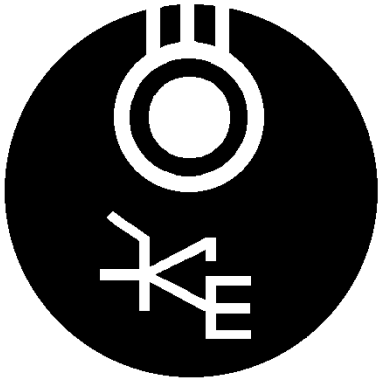
Міністерство освіти і науки України
Національний гірничий університет

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



МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи МПП-4
«ПРОГРАМУВАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ПРИ ВИКОРИСТАННІ МІКРОКОНТРОЛЕРІВ 8051АН СІМЕЙСТВА MCS-51»,
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-4 «Програмування арифметичних операцій при використанні мікроконтролерів 8051АН сімейства MCS-51», 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-4 «Програмування арифметичних операцій при використанні мікроконтролерів 8051АН сімейства MCS-51», 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. – 32 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 – подальше розширення і поглиблення знань з дисципліни «Мікропроцесорні пристрої». До вивчення пропонуються методи програмування арифметичних операцій над цілими числами та програмне забезпечення для їх виконання мікро контролером 8051АН сімейства MCS-51.

Методичні вказівки, окрім навчального матеріалу для ознайомлення із загальними принципами організації виконання арифметичних операцій, вміщують приклади програмних продуктів для забезпечення виконання арифметичних операцій з дво- та багайтовими числами без та зі знаком, в тому числі із використанням двійково-десяткової системи числення. Передбачено, що наведені у методичних вказівках приклади програм будуть використані для їх подальшої модифікації шляхом розробки відповідних підпрограм, які стануть основою при розробці програмного забезпечення відповідно індивідуальних завдань до виконання лабораторної роботи.

Далі наведені запитання для самоперевірки, індивідуальні завдання до самостійної роботи та індивідуальні завдання низького, середнього та підвищеного рівня складності до виконання лабораторної роботи, список літератури та адреси інформаційних сайтів виробників мікроконтролерів і програмного забезпечення у всесвітній мережі Internet.

1. ЗМІСТ САМОСТІЙНОЇ ТА ЛАБОРАТОРНОЇ РОБІТ

1.1. Самостійна робота

Самостійна робота полягає у вивченні основних засад програмування арифметичних операції над цілими числами з та без знаку, в тому числі одно-, дво- та багатобайтовими. Слід ознайомитися з призначенням, організацією, алгоритмами та прикладами програмного забезпечення для виконання арифметичних операцій за допомогою мікроконтролера 8051АН фірми Intel. Засвоївши матеріал, необхідно відповісти на запитання для самоперевірки та виконати індивідуальне контрольне завдання до самостійної роботи з метою оцінки рівня опанування матеріалу, який викладено у методичних вказівках.

1.2. Лабораторна робота

Виконується у комп’ютерних класах кафедри. До виконання допускаються студенти, які ознайомилися із теоретичним матеріалом п. 1.1 та підготували попередній звіт згідно поставлених вимог. Метою роботи є вивчення основних засад організації програм для виконання арифметичних операцій над цілими числами на мові асемблера мікроконтролера 8051АН із наступною перевіркою отриманих знань при виконанні індивідуальних завдань. Захист роботи – шляхом демонстра-

ції викладачу ефективності розробленого програмного забезпечення відповідно до вибраного рівня складності індивідуального завдання на прикладі чисел, які вибрані викладачем, доказу самостійності розробки при варіативних змінах викладачем умов вибраного завдання. Остаточний захист – шляхом виконання тестової контрольної роботи на ПЕОМ.

Програма виконання

Після засвоєння матеріалу п.1.1 методичних вказівок (МВ) відповісти на запитання для самоперевірки.

- Виконати індивідуальний варіант самостійної роботи (номер варіанту повинен збігатися із порядковим номером студента у журналі групи).
- Після ознайомлення із матеріалом п.п. 2…5 та вибору рівня складності індивідуального завдання на виконання лабораторної роботи модифікувати необхідні програмні продукти до вигляду підпрограм.
- На основі розроблених підпрограм та вимог ГОСТ розробити алгоритм та налагодити у середовищі COMPASS/51 необхідне програмне забезпечення відповідно до вибраного рівня індивідуального завдання.
- Скласти підсумковий звіт і захистити лабораторну роботу відповідно наведених вище умов захисту.

Вказівки щодо складання звіту

Підсумковий звіт повинен містити:

- Назву, мету та програму роботи.
- Відповіді на запитання самоперевірки.
- Алгоритм та лістинг розробленої програми.

2. ДОДАВАННЯ ТА ВІДНІМАННЯ ЦІЛИХ ЧИСЕЛ

2.1. Загальні відомості

При розробці програмного забезпечення перш за все слід визначити тип та розмір чисел, над якими необхідно виконувати арифметичні операції. Щодо типу, то слід мати на увазі, що цілі числа можуть бути без або із знаком. У останньому випадку слід забезпечити контроль та врахування знаку за допомогою значення того чи іншого біту. Розмір чисел визначається їх довжиною у бітах або байтах.

Якщо довжина чисел не перевищує 7 біт, то для визначення знаку таких чисел використовують старший 8-й біт кожного байту (якщо він дорівнює «1» – то число від’ємне, а якщо «0» – то додатне). Решта бітів призначені для визначення величини чисел і тому за цих умов одним байтом можна визначати числа у межах від -128_{10} до $+127_{10}$.

Якщо результат додавання виходить за вибраний для операндів розмір у 7 біт, то одночасне встановлення в 1 8-го біту акумулятора і ознаки (прапорця) OV регістра PSW слід інтерпретувати так: сума чисел перевищує 127_{10} і є байтом із значущим 8-им розрядом. Якщо отримана в результаті виконання операції віднімання 7-ми бітових чисел різниця додатна, то 8-ий біт акумулятора і ознака OV регістра PSW в 1 не встановлюються. Коли ж результатом віднімання є від'ємне число, то 8-ий біт акумулятора і ознака C регістра PSW встановлюються в 1, а ознака (прапорець) OV регістра PSW – ні. Це свідчить про те, що в акумуляторі знаходиться додатковий код від'ємного числа різниці і для визначення його модуля слід про інвертувати вміст акумулятора (отримати інверсний код), а потім збільшити його на 1.

Якщо при додаванні операнди 8-ми розрядні (однобайтні), то про переповнення акумулятора свідчить встановлення в 1 ознаки C регістра PSW, причому вона стає значущою цифрою. При виконанні операції віднімання однобайтних чисел встановлення в 1 ознаки C свідчить про отримання в акумуляторі від'ємної різниці і що для отримання її модуля вміст акумулятора слід проінвертувати і збільшити на 1. Наприклад, якщо в результаті віднімання в акумуляторі число 0Ah (00001010) і ознака C=1, то інверсний код числа 0F5h (11110101), а додатковий 0F6h (11110101 + 1 = 11110110).

Для виконання зазначених операцій додавання і віднімання однобайтових чисел користуються командами мови асемблера ADD, ADDC та SUBB. Звичайно для таких операцій необхідно і достатньо занести в акумулятор перший операнд (один із доданків при додаванні або зменшуване при відніманні), потім у допоміжний регістр – другий і за допомогою відповідної команди визначити операцію для виконання. Наприклад, скласти програму

```
MOV A,#32h      ;занесення до акумулятора першого доданку
MOV R1,#01h     ;занесення в регістр R1 другого доданку
ADD A,R1        ;розміщення в акумуляторі суми
SUBB A,R1       ;розміщення в акумуляторі різниці
```

При додаванні (відніманні) другий операнд може бути як безпосереднім числом, так і визначатися вмістом регістра, адреса якого указується. Результат обчислення завжди заноситься до акумулятора, а вміст регістрів – початкових операндів – не змінюється.

Операцію віднімання можна замінити на операцію додавання додаткового коду від'ємника. Якщо в результаті цього ознака перенесення C встановиться в 1, то це буде свідчити, що результат є додатнім числом. І навпаки, коли ознака перенесення C=0, то результатом буде від'ємне число і для отримання його модуля необхідно знайти відповідний додатковий код.

Наприклад, необхідно знайти різницю між числами 13 і 5. Спочатку взяті числа подамо у двійковій системі числення. Тоді число $13_{10}=00001101_2$, а число $5_{10}=00000101_2$. Тоді додатковий двійковий код числа $-5_{10}= 11111011_2$. В результаті додавання отримуємо наступний результат:

$$\begin{array}{rll} & 0000\ 1101 & (13_{10}) \\ + & 1111\ 1011 & (-5_{10}) \\ \hline C=1 & 0000\ 1000 & (8_{10}) \end{array}$$

Оскільки ознака C=1, то результатом такого віднімання є додатнє число, тобто результатом є число $+8_{10}$ або 1000_2. Нижче наведена програма реалізації такої процедури за умови, що попередньо занесений в акумулятор, а зменшуване – в регістр R1. Результат виконання програми розміщено в акумуляторі.

Програма додавання додаткового коду

```
      CLR C       ; очищення ознаки перенесення C
      CPL A       ; отримання в акумуляторі інверсного коду від'ємника
      INCA        ; отримання в акумуляторі додаткового коду від'ємника
      ADD A,R1    ; сума зменшуваного і додаткового коду від'ємника
      JC m1       ; перехід на мітку m1коли результат додатній
      CPL A       ; отримання інверсного коду різниці
      INCA        ; отримання модуля від’ємної різниці
m1:   NOP
```

При виконанні таких операцій віднімання над багатобайтними числами без знаку також при C=0 результат від'ємний, а при C=1 – додатній.

2.2. Двобайтні числа без знаку

Програмування операцій додавання та віднімання двобайтних чисел ускладнюється через відсутність у досліджуваного мікроконтролера необхідних команд для двобайтних операндів. Доводиться розробляти програмне забезпечення на основі команд для однобайтних операндів. Вручну операції додавання та віднімання можна здійснити наступним чином:

+1	-1
1F 23h	58 02h
+ 20 FAh	– 11 BFh
40 1Dh	46 43h

Як видно із наведених прикладів, операції додавання (віднімання) зводяться до операцій побайтного додавання (віднімання) старших і молодших байтів з урахуванням бітів переповнення (запозичення) C, позначених як +1 (-1).

Отже, при виконанні операції додавання слід спочатку визначити суму молодших байтів операндів за допомогою команди ADD, а потім – суму старших байтів з урахуванням ознаки переповнення C, тобто за допомогою команди

ADDC. При визначенні різниці (відніманні) необхідно виконати подібні дії, але при цьому слід врахувати наступне. Справа в тому, що система команд мікроконтролера 8051 не передбачає виконання звичного віднімання (без урахування біту запозичення). Тому на початку програми віднімання слід очищати (скинути в 0) біт перенесення C. Різниця між процедурами додавання і віднімання ще й у тому, що при додаванні слід враховувати перенесення, а при відніманні – запозичення. Хоча для збереження ознаки запозичення при відніманні використовується той же самий біт перенесення. Як приклади програмного забезпечення нижче наведені програми для виконання операцій додавання та віднімання двобайтних чисел.

У наведеній програмі додавання передбачено, що перед виконанням програми старший (СТБ) і молодший (МЛБ) байти першого доданку повинні бути розміщеними у регістрах R3 і R2 відповідно. Для другого доданку СТБ і МЛБ попередньо розміщують у регістрах R5 і R4 у такій же послідовності.

Програма додавання двобайтних чисел

```
MOV A, R4  ; перемістили МЛБ 2-го доданку в акумулятор
ADD A, R2  ; сума молодших байтів доданків в акумуляторі
MOV R4, A  ; молодший байт суми в регістрі  R4
MOV A, R5  ; перемістили СТБ 2-го доданку в акумулятор
ADDC A, R3 ; сума старших байтів з урахуванням ознаки
           ; перенесення C в акумуляторі
MOV R5, A  ; сума старших байтів доданків в регістрі R5
           ; результат додавання в R5,R4 та ознаці C
```

Наведена нижче програма віднімання теж передбачає попереднє розташування старшого (СТБ) і молодший (МЛБ) байтів зменшуваного у регістрах R3 і R2 відповідно. Старший (СТБ) і молодший (МЛБ) байт від'ємника теж заздалегідь розміщують у регістрах R5 та R4. Там же розміщують і шукану різницю.

Програма віднімання двобайтних чисел

```
CLR C      ; скидання ознаки перенесення C
MOV A, R2  ; в акумуляторі молодший байт (МЛБ) зменшуваного
SUBB A, R4 ; в акумуляторі різниця МЛБ зменшуваного і від'ємника
MOV R4, A  ; різниця молодших байтів на місці МЛБ результату
MOV A, R3  ; в акумуляторі старший байт (СТБ) зменшуваного
SUBB A, R5 ; в акумуляторі різниця СТБ зменшуваного і від'ємника
           ; з урахуванням можливого запозичення;
MOV R5, A  ; різниця СТБ зменшуваного і від'ємника
           ; результат віднімання знаходиться регістрах R5,R4
           ; та ознаці перенесення C
```

2.3. Багатобайтні числа без знаку

При розробці програм для виконання операцій над багатобайтними числами слід визначитися із їх довжиною. Оскільки у стандартного x51 всього 128 байт внутрішньої оперативної пам'яті (в якій повинні розташуватися не лише операнди,

але й стек, змінні програми та буфери інформації для відображення на дисплеї), то можливості внутрішнього ОЗП обмежені. Тому наведені нижче приклади програм будуть для чисел довжиною не вище 31 байт (на практиці вона рідко перевищує 10). Програми базуються на тих же принципах, що і для двобайтних чисел.

У програмі додавання багатобайтних чисел без знаку доданки розташовуються у внутрішній оперативній пам'яті мікроконтролера. Молодші байти доданків розміщені за меншими адресами, а старші – за більшими. Адреси молодших байтів доданків задаються в індексних регістрах R0 і R1. Розмір доданків в байтах (*NN*) формується у регістрі R2. Результат додавання (сума) розміщується на місці другого доданку. Попередньо ознака перенесення C встановлюється в 0 інструкцією CLR C. Далі *NN* разів виконується тіло циклу, яке складає байти доданків за допомогою інструкції ADDC. Додавання розпочинається з молодших байтів і закінчується найстаршими. Оскільки при виконанні циклу вміст регістрів R0 і R1 збільшується на *NN*, то відновлення вмісту цих регістрів перед виходом із програми відбувається шляхом віднімання *NN* із кожного (це зроблено для зручності, щоб не вносити наново в регістри R0 і R1 адреси доданків, які можуть бути записані в ОЗП МК на місце попередніх для визначення суми наступної пари чисел).

Програма додавання багатобайтних чисел

```
      MOV B, R2  ; занесення довжини операндів в регістр B
M1:   MOV A, @R1 ; занесення в акумулятор МЛБ 2-го доданку
      ADDC A, @R0; сума МЛБ доданків в акумуляторі
      MOV @R1, A ; занесення суми в ОЗП за вибраною адресою
      INC R0     ; встановлення адреси наступного байту 1-го доданку
      INC R1     ; встановлення адреси наступного байту 2-го доданку
      DJNZ B, M1 ; зменшення на 1 кількості байт та перехід на M1, якщо
                 ; вона не дорівнює 0
      PUSH 0D0H  ; тимчасове збереження ознак (прапорців) регістру PSW
      MOV A, R0  ; відновлення вмісту регістру R0 (початкової адреси
      CLR C      ; ОЗП, за якою зберігався МЛБ 1-го доданку)
      SUBB A, R2
      MOV R0, A
      MOV A, R1  ; відновлення вмісту регістру R1 (початкової
      CLR C      ; адреси ОЗП, за якою зберігався МЛБ 2-го доданку)
      SUBB A, R2
      MOV R1, A
      POP 0D0H   ; відновлення прапорців регістру PSW
```

У програмі віднімання багатобайтних чисел без знаку, як і вище, молодші байти розміщені за меншими адресами, а старші – за більшими. Адреса молодшого байту від'ємника задається індексним регістром R0, а молодшого байту зменшуваного – R1. Розмір операндів – у регістрі R2. Результат розміщується на місці зменшуваного. Ознака C свідчить про запозичення, а 0V – про переповнення. Після виконання програми вміст регістрів R0, R1 та R2 перед виходом із програми відновлюється.

Програма віднімання багатобайтних чисел

```
      MOV B, R2  ; занесення розміру операндів
      CLR C      ; очищення прапорця перенесення C перед відніманням
M2:   MOV A, @R1 ; занесення МЛБ зменшуваного в акумулятор
      SUBB A, @R0; занесення різниці МЛБ в акумулятор
      MOV @R1, A ; занесення різниці у вибрану область ОЗП
      INC   R0   ; встановлення адреси наступного байта від’ємника
      INC   R1   ; встановлення адреси наступного байта зменшуваного
      DJNZ B, M2 ; зменшення на 1 розмірності в байтах, якщо вона не 0,
                 ; то перехід на M2
      PUSH 0D0H  ; тимчасове збереження прапорців
      MOV A, R0  ; відновлення вмісту регістру R0 (початкової
      CLR C      ; адреси ОЗП, за якою зберігався МЛБ від'ємника)
      SUBB A, R2
      MOV R0, A
      MOV A, R1  ; відновлення вмісту регістра R1 (початкової
      CLR C      ; адреси ОЗП, за якою зберігався МЛБ зменшуваного
      SUBB A, R2
      MOV R1, A
      POP 0D0H   ; відновлення прапорців
```

3. МНОЖЕННЯ ЦІЛИХ ЧИСЕЛ БЕЗ ЗНАКУ

3.1. Загальні відомості

Як і у випадку із додаванням операція множення однобайтних операндів є простою і здійснюється за допомогою стандартної інструкції асемблера. Різниця лише в тому, що при додаванні один операнд заносився до акумулятора, а інший міг бути вмістом регістру або просто числом, то при виконанні множення операнди заносяться до акумулятора і його розширювача регістру B. Для виконання множення передбачена інструкція MUL AB, яка забезпечує множення вмісту 8-бітового регістра A на вміст 8-бітного регістра B. Старший байт добутку заноситься до регістру B, а молодший – регістру A. Отже мікроконтролер апаратно забезпечує виконання операції множення 8 біт × 8 біт = 16 біт.

Як приклад, нижче наведена програма множення однобайтних чисел, яка здійснює цю процедуру для двох операндів. Один із них попередньо занесений в акумулятор A, а другий – в регістр B. Двобаштовий результат множення у цих же регістрах, причому старший байт добутку – в регістрі B, а молодший – в акумуляторі A.

Програма множення однобайтних чисел

```
      MOV A,#23h ; занесення в акумулятор першого операнда (23h)
      MOV B,24h  ; занесення у розширювач акумулятора
                 ; вмісту елемента пам’яті (ОЗП) з адресою 24h
      MUL AB     ; виконання операції множення (в акумуляторі A
                 ; молодший байт результату, а у розширювачі B – старший
```

Команди множення дають правильні результати лише за умови, що операнди додатні, а тому для правильного обчислення добутку при від'ємних операндах спочатку необхідно визначити їх модулі. Після множення у разі потреби можна повернутися до первинного кодування з урахуванням знаку результату. Таким чином, для множення чисел, хоча б одне з яких від'ємне, слід знати способи визначення модуля числа та обчислення знаку результату.

Знак результату множення двох чисел буде від'ємним коли одне з них від'ємне. Для прикладу приймаємо, що знак першого операнду занесений в старший біт регістра R0, а знак другого – в старшому біті регістра R1. Тоді знак результату можна записати, наприклад, в біт 07 (ОЗП) за допомогою наступних 4-х команд:

```
MOV A, R0  ; знак першого числа у старшому біті накопичувача
XRL A, R1  ; обчислення знаку результату
RLC A      ; старший біт накопичувача в прапорці перенесення
MOV 07, C  ; знак результату збережений в біті 07
```

Вибраний біт зручний для запам'ятовування знаку результату тому, що на нього не впливають результати команд, які виконують арифметичні операції.

Як приклад нижче наведена програма множення однобайтних чисел з довільним знаком. Співмножники знаходяться в регістрах R0 та R1, а їх добуток – в регістрі B (старший байт) і накопичувачі A (молодший байт).

Програма множення однобайтних чисел зі знаком

```
      MOV A, R0  ; множене
      MOV C, ACC.7; знак першого числа в біті C
      JNC mls1   ; перехід, коли множене додатне
      CPL A      ; обчислення інверсного коду множеного
      INC A      ; обчислення модуля множеного
mls1: MOV B, R1  ; множник
      JNB B.7,mls2; перехід по додатному множнику
      CPL C      ; обчислення знаку добутку
      XRL B, #FFh; обчислення інверсного коду множника
      INC B      ; обчислення модуля множника
mls2: MOV 07, C  ; запам'ятовування знаку добутку
      MUL AB     ; обчислення добутку модулів
      JB 07,mls3 ; перехід по позитивному добутку
      CPL A      ; обчислення інверсного коду молодшого байту
      ADD A, #01h; отримання додаткового коду
      XRL B, #FFh; отримання інверсного коду старшого байту
      JNC mls3   ; перехід за відсутності перенесення
      INC B      ; корекція старшого байту
mls3: NOP        ; для запису мітки
```

3.2. Двобайтні числа без знаку

Правила, за якими виконуються арифметичні операції, єдині для всіх позиційних систем числення. Множення шістнадцяткових чисел виконується подібно

тому, як для десяткових чисел. Для розробки програми множення двобайтних чисел необхідно детально розглянути порядок перемноження чисел у стовпчик і на цій основі розробити алгоритм. Як приклад розглянемо множення в стовпчик:

	3Fh	
*	0A5h	
+	4Bh	
+	0Fh	+ 13Bh
+	96h	276h
	1Eh	
	289Bh	= 289Bh

У прикладі така послідовність дій: спочатку множимо молодшу цифру множника на молодшу цифру множеного і отримуємо дворозрядне шістнадцяткове число 4Bh (5h·0Fh = 4Bh). Цифру B залишаємо молодшим (нульовим) розрядом першого часткового добутку, а старшу цифру 4 запам'ятовуємо. Після цього множимо молодшу цифру множника на старшу цифру множеного і отримуємо шістнадцяткове число 0Fh (5h·3h=0Fh). Додавши до 0Fh отримане раніше число 4, одержуємо 13h – старший розряд, що разом із молодшим розрядом дає перший частковий добуток 13Bh (іншими словами, 3Fh·5h=13Bh). Далі подібним чином множимо старшу цифру множника на множене, отримуємо число 276h (0Ah·3Fh=276h). Зсунувши цей другий частковий добуток на один розряд ліворуч і склавши його із першим (13Bh), отримуємо остаточний результат: 2760h+3Bh=289Bh.

Для реалізації алгоритму двобайтних чисел слід мати в розпорядженні байтову таблицю множення (від 1h·1h=1h до 0FFh·0FFh=0FE01h). Враховуємо, що інструкція MUL AB і є «байтовою» таблицею множення, яка вбудована в систему команд МК 51 і ми можемо використати її при розробці програми багатобайтного множення.

При розробці алгоритму множення один на одного двох двобайтних чисел слід забезпечити процедуру множення формату 16 · 16 = 32 біт із використанням стандартної інструкції, яка виконує процедуру множення операндів формату 8 біт · 8 біт=16 біт. Якщо є два 16-бітових числа X та Y із старшими (XH, YH) і молодшими (XL, YL) байтами відповідно, то справедливі вирази:

$$X=XH\cdot 28+XL;$$
$$Y=YH\cdot 28+YL.$$

В результаті послідовного виконання операцій множення отримуємо підсумковий вираз вигляду

$$X\cdot Y=(XH\cdot 28+XL)\cdot(YH\cdot 2^8+YL)=XH\cdot YH\cdot 216+(XH\cdot YL+XL\cdot YH)\cdot 28+XL\cdot YL.$$

Цілком очевидно, що обчислення добутку X·Y двобайтних чисел X і Y зводиться до знаходження чотирьох добутків (молодших та старших байтів обох чисел один на одного) і подальшого підсумовування із відповідними зсувами отриманих добутків. Як приклад розглянемо використання такої послідовності дій для чисел X=46 21h та Y=25 87h. Маємо результати множення в стовпчик (ліворуч) та відповідні пояснення (праворуч):

	46 21h	X (XH=46h; XL=21h)
	* 25 87h	Y (YH=25h; YL=87h)
+	11 67h	XL·YL
+	24 EAh	XH·YL·2^8
+	04 C5h	XL·YH·2^8
	0A 1Eh	XH·YH·2^{16}
	0A 47 C0 67h	XH·YH·2^{16}+(XH·YL+XL·YH) 2^8+XL·YL

При розробці програми множення для автоматичного урахування пере-повнень при додаванні слід користуватися командою ADDC (множення на 2^8 і 2^{16} еквівалентні тривіальним зрушенням ліворуч на 8 і 16 біт відповідно).

Розглянемо програму множення двобайтних чисел за умови, що попередньо XH – старший байт (СТБ) множеного занесено до регістру R3, а YH – старший байт множника – в регістр R5. Відповідно молодший байт (МЛБ) множеного XL розташовано в регістрі R2, а молодший байт множника YL – в регістрі R4.

Програма множення двобайтних чисел

```
MOV A,R3   ; занесли XH множеного в акумулятор A
MOV B,R5   ; занесли YH множника в регістр B
PUSH B     ; зберегли в стеку YH (вміст регістра B)
MUL AB     ; отримали добуток XH·YH
MOV R7,B   ; зберегли в R7 СТБ добутку XH·YH
MOV R6,A   ; зберегли в R6 МЛБ добутку XH·YH
MOV A,R2   ; занесли XL множеного в акумулятор A
MOV B,R4   ; занесли YL множника в регістр B
PUSH B     ; зберегли в стеку YL (вміст регістра B)
MUL AB     ; отримали добуток XL·YL
MOV R5,B   ; зберегли в R5 СТБ добутку XL·YL
MOV R4,A   ; зберегли в R4 МЛБ добутку XL·YL
MOV A,R3   ; занесли XH множеного в акумулятор A
POP  B     ; відновили YL із стеку в регістр B
MUL AB     ; отримали добуток XH·YL
ADD A,R5   ; сума МЛБ добутку XH·YL і СТБ добутку XL·YL
MOV R5,A   ; сума МЛБ XH·YL і СТБ XL·YL збережена в R5
MOV A,R6   ; занесли МЛБ добутку XH·YH в акумулятор A
ADDC A,B   ; сума МЛБ XH·YH і СТБ XH·YL (з урахуванням C)
MOV R6,A   ; зберегли суму МЛБ XH·YH і СТБ XH·YL у регістрі R6
MOV A,R7   ; занесли СТБ добутку XH·YH в акумулятор A
ADDC A,#0  ; сума СТБ добутку XH·YH і біту перенесення C
```

```
      MOV R7,A   ; сума СТБ XH·YH і біту C у регістрі R7
      MOV A,R2   ; занесли МЛБ множеного XL в акумулятор A
      POP B      ; відновили YH із стеку в регістр B
      MUL AB     ; отримали добуток XL·YH
      ADD A,R5   ; сума МЛБ XL·YH +МЛБ XH·YL+СТБ XL·YL
      MOV R5,A   ; МЛБ XL·YH+МЛБ XH·YL+СТБ XL·YL в регістрі R5
      MOV A,R6   ; занесли суму МЛБ XH·YH+СТБ XH·YL в акумулятор
      ADDC A,B   ; сума (з урахуванням C) СТБ XL·YH, МЛБ XH·YH
      MOV R6,A   ; та СТБ XH·YL в регістрі R6
      MOV A,R7   ; СТБ добутку XH·YH в акумуляторі
      ADDC A,#0  ; додали біт C до СТБ добутку XH·YH
      MOV R7,A   ; і зберегли суму у регістрі R7
      ORL A,R6   ;
      JZ M_1     ; C=0, якщо результат помістився у 16 біт;
      SETB C     ; C=1, якщо результат не помістився у 16 біт;
M_1:  JMP M_1
```

Результат множення розташований в регістрах R7 R6 R5 R4.

3.3. Багатобайтні числа без знаку

Добуток двох багатобайтних чисел отримують наступним чином. Спочатку множать множене на молодший байт множника і запам'ятовують результат. Після цього множать множене на другий справа байт множника, зсувають результат цього множення на один байт ліворуч і складають із першим добутком. Після цього множать множене на третій справа байт множника, зсувають результат цього множення вже на два байти ліворуч і складають із попередньою сумою. І так до старшого байта множника, коли до накопичуваної суми його прибавимо його добуток на все те ж множене.

Програма множення таких чисел наведена нижче, де R0 – адреса МЛБ множеного; R1 – адреса МЛБ множника, за якими розташовані ще N байт для старшої половини добутку (старші байти чисел розташовані за старшими адресами, тобто, для двобайтових чисел R0 і R1 вказуватимуть на МЛБ чисел, а R0+1 і R1+1 – на СТБ чисел); R2 – розмір множника/множеного в байтах (тобто число N (N<32); R1 – адреса 2N-байтного добутку (записується на місце множника).

Програма множення беззнакових багатобайтних чисел

```
      MOV A,R1
      MOV R3,A   ; зберегли R1 в R3
      ADD A,R2
      MOV R1,A   ; R1 – адреса МЛБ старшої половини добутку
      MOV B,R2   ; в регістр B занесли розмір множника/множеного
m_1:  MOV @R1,#0
      INC R1
      DJNZ B,m_1 ; встановили нулі в старшій половині добудку
      MOV A,R3
      MOV R1,A   ; відновили R1 із R3
      MOV A,R2   ; (A)=N. N – кількість циклів додавання і зсуву
m_2:  PUSH  ACC
```

```
      MOV A,@R1
      MOV R4,A   ; в R4 МЛБ множника
      MOV A,R1
      ADD A,R2
      MOV R1,A   ; R1 – адреса МЛБ старшої половини добутку
      MOV R3,#0  ; байт перенесення С на початку встановлений в 0
      MOV A,R2   ; (A)=N – N однобайтних множень в циклі
m_3:  PUSH ACC   ; додавання зсуву
      MOV A,@R0
      MOV B,R4
      MUL ABADD A,R3  ; врахували байт перенесення С
      XCH A,B
      ADDC A,#0
      XCH A,B
      ADD A,@R1
      MOV @r1,A  ; байт часткового добутку готовий
      MOV A,B
      ADDC A,#0
      MOV R3,A   ; наступний байт перенесення готовий
      INC R0
      INC R1
      POP ACC
      DJNZ ACC,m_3    ; зациклення
      MOV A,R0
      SUBB A,R2
      MOV R0,A   ; R0 – адреса МЛБ множеного
      MOV A,R2
      RL A
      MOV B,A
m_4:  DEC R1
      MOV A,@R1
      XCH A,R3
      MOV @R1,A
      DJNZ B,m_4 ; частковий добуток зсунуто на байт праворуч
      POP ACC
      DJNZ ACC,m_2    ; зациклення
      MOV A,R1
      MOV R3,A   ; зберегли R1 в R3
      ADD A,R2
      MOV R1,A   ; R1 – адреса МЛБ старшої половини добутку
      CLR A      ; порівняння старшої половини добутку з нулем
      MOV B,R2
mm:   ORL A,@R1
      INC R1
      DJNZ B,mm
      MOV B,A
      MOV A,R1
      CLR C
      SUBB A,R2
      MOV R1,A
      MOV A,B
      CLR C
      JZ m_5     ; C=0, якщо результат розмістився в N байт
      SETB C     ; C=1, якщо результат не розмістився в N байт
```

```
M_5:  MOV A,R3
      MOV R1,A   ; відновили R1 із R3
```

4. ДІЛЕННЯ ЦІЛИХ ЧИСЕЛ

4.1. Загальні відомості

Як і у випадку зі складанням та відніманням програмна реалізація дій множення та ділення однобайтних операндів не представляє великої складності, вони так само оформлені у вигляді стандартних інструкцій асемблера. Різниця полягає тільки у тому, що якщо при виконанні складання (віднімання) один операнд заносився до акумулятору, а другий міг бути вмістом регістру або просто числом, то при виконанні множення (ділення) операнди заносяться до акумулятору і його розширювач (регістр B). Для виконання множення передбачена інструкція MUL AB. Ця інструкція перемножує вміст 8-бітового регістра A на вміст 8-бітового регістра B. Старша частина добутку записується в регістр B, а молодша — в регістр A. Таким чином, мікроконтролер вміє апаратно здійснювати множення (ділення) 8 • 8 = 16 біт. Для виконання ділення передбачена інструкція DIV AB, результат ділення записується також в регістрах A і B, причому, в A ціла частина частки, а у B – залишок.

Програма розподілу однобайтних чисел

```
MOV A,#48h ;внесення в акумулятор числа 48h
MOV B,24h  ;внесення у розширювач акумулятора
           ;вміст регістра 24h
DIV AB     ;виконуємо ділення, в акумуляторі
           ;залишається частка, у розширювачі – залишок
```

4.2. Ділення двобайтних чисел

Програмування ділення великих чисел важке через відсутність кінцевих формул для обчислення частки та залишку. Розглянемо детально задачу обчислення частки від ділення. Операція ціло-чисельного ділення полягає в приведенні неправильного дробу до правильного, при цьому ціла частина числа є часткою, а чисельник дробової частини – залишком. Позначимо частку і залишок буквами Q і R відповідно. Тоді задача ціло-чисельного ділення зводиться до знаходження двох додатних цілих чисел, які одночасно задовольняють рівнянню X=Q*Y+R і нерівності R<Y

Це рівняння розв’язується підбором частки. Наприклад, це можна робити багатократним відніманням дільника з ділимого до тих пір, доки залишок не стане меншим дільника. Тоді частка дорівнюватиме кількості операцій віднімання. Економнішим є метод пробних обчислень різниці між ділимим і добутком дільника на наближене значення частки. При цьому спочатку підбирається старша цифра частки – як найбільше із значень, за якого залишок ще додатній, а потім наступні ци-

фри. Це метод ділення в «стовпчик». Для двійкового кодування чисел цей метод найбільш ефективний, оскільки дозволяє при кожній пробі обчислювати чергову цифру частки.

Програма ділення двобайтних чисел:

```
      MOV R7,#07Ch     ; МЛБ ділимого
      MOV R6,#02Ah     ; СТБ ділимого
      MOV R5,#0E9h     ; МЛБ дільника
      MOV R4,#4h       ; СТБ дільника
div1:
      MOV A,R7
      SUBB A,R5
      MOV R7,A
      MOV A,R6
      SUBB A,R4
      MOV R6,A         ; віднімання дільника з ділимого
      jc div4          ; виконуємо наступну команду коли дільник
                       ; менший ділимого
      CLR C            ; очищення біта перенесення C
      MOV A,R3
      ADD A,#1h        ; в регістрах, відведених для частки, формується
      MOV R3,A         ; її значення у кожному циклі
      MOV A,R2
      ADDC A,#0h
      MOV R2,A
      Jmp div1
div4:
      MOV A,R7
      ADD A,R5
      MOV R7,A
      MOV A,R6
      ADDC A,R4
      MOV R6,A         ; відновлюємо значення ділимого
```

4.3. Ділення багатобайтних чисел

Програма ділення багатобайтних цілих чисел із відновленням залишку (наведена нижче) передбачає, що вихідний результат формується в елементах пам’яті з початковою адресою, яка буде занесеною до регістру R1 (спочатку ціла частина, а потім залишок).

Програма ділення багатобайтних чисел

```
      MOV 20H,#11H     ; занесли шестибайтне ділене
      MOV 21H,#46H
      MOV 22H,#10H
      MOV 23H,#32H
      MOV 24H,#46H
      MOV 25H,#11H
      MOV 26H,#78H     ; занесли 3-х байтний дільник
```

```
      MOV 27H,#36H
      MOV 28H,#12H
      MOV R0,#26H      ; адреса МЛБ дільника
      MOV R1,#20H      ; адреса МЛБ діленого
      MOV R2,#3H       ; занесли розмір дільника в байтах
      mov A,R1
      mov R4,A         ; зберегли R1 в R4
      ADD A,R2
      mov R1,A         ; занесли адресу МЛБ старшої половини ділимого
      LCALL CMPN       ; порівняння старшої половини дільника та ділимого
      mov A,R4
      mov R1,A
      JC DIVN_0
      SETB C
      RET
DIVN_0:
      mov A,R2
      RL A
      RL A
      RL A
      mov R4,A
DIVN_1:
      mov A,R2
      RL A
      mov R2,A
      LCALL RLAN
      PUSH PSW
      mov A,R2
      RR A
      mov R2,A
      ADD A,R1
      mov R1,A
      POP PSW
      JC DIVN_2
      LCALL SUBN
      JNC DIVN_3
      LCALL ADDN
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      SJMP DIVN_4
DIVN_2:
      LCALL SUBN
DIVN_3:
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      INC @R1
DIVN_4:
      DJNZ R4,DIVN_1
      CLR C
M:    JMP M
```

```
CMPN:
      mov B,R2
      mov R3,#0
      CLR C
CMPN_1:
      mov A,@R1
      SUBB A,@R0
      XCH A,R3
      ORL A,R3
      XCH A,R3
      INC R0
      INC R1
      DJNZ B,CMPN_1
      JNB OV,CMPN_2
      CPL OV
CMPN_2:
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUbb A,R2
      mov R1,A
      mov PSW,B
      mov A,R3
      RET
RLAN:
      CLR C
      SJMP RLCN
RLCN:
      mov B,R2
RLCN_1:
      mov A,@R1
      RLC A
      MOV @r1,A
      INC R1
      DJNZ B,RLCN_1
      mov B,PSW
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
      RET
SUBN:
      MOV B,R2
      CLR C
SUBN_1:
      mov A,@R1
      SUBB A,@R0
      mov @R1,A
      INC R0
```

```
      INC R1
      DJNZ B,SUBN_1
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
      RET
ADDN:
      mov B,R2
      CLR C
ADDN_1:
      mov A,@R1
      ADDC A,@R0
      mov @R1,A
      INC R0
      INC R1
      DJNZ B,ADDN_1
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
      RET
```

Як приклад, нижче наведена програма ділення шестибайтного числа на трибайтне. Передбачено, що результат ділення формується в елементах пам’яті з початковою адресою в регістрі R1 (спочатку ціла частина, а потім залишок)

Програма ділення багатобайтних цілих чисел із відновленням залишку

```
      MOV 20H,#11H     ; занесли шестибайтне ділене
      MOV 21H,#46H
      MOV 22H,#10H
      MOV 23H,#32H
      MOV 24H,#46H
      MOV 25H,#11H
      MOV 26H,#78H     ; занесли трибайтний дільник
      MOV 27H,#36H
      MOV 28H,#12H
      MOV R0,#26H      ; адреса МЛБ дільника
```

```
      MOV R1,#20H      ; адреса МЛБ діленого
      MOV R2,#3H ; занесли розмір дільника в байтах
      mov A,R1
      mov R4,A         ; зберегли R1 в R4
      ADD A,R2
      mov R1,A         ; занесли в R1 адресу МЛБ старшої половини діленого
LCALL CMPN             ; порівняння старшої половини дільника та діленого
      mov A,R4
      mov R1,A
      JC DIVNB_0
      SETB C
      RET
DIVNB_0:
      mov A,R2
      RL A
      RL A
      RL A
      mov R4,A
      CLR A
      PUSH ACC
DIVNB_1:
      mov A,R2
      RL A
      mov R2,A
LCALL RLAN
      POP ACC
      RR A
      RLC A
      PUSH ACC
      mov A,R2
      RR A
      mov R2,A
      ADD A,R1
      mov R1,A
      POP ACC
      PUSH ACC
      JB ACC.1,DIVNB_2
LCALL SUBN
      SJMP DIVNB_3
      DIVNB_2:
LCALL ADDN
DIVNB_3:
      POP ACC
      RLC A
      ADD A,#2
      movc A,@A+PC
      SJMP DIV16B_4
      DB 01B, 10B, 01B, 01B
      DB 10B, 10B, 10B, 01B
DIV16B_4:
      PUSH ACC
      mov A,R1
      CLR C
      SUBB A,R2
```

```
      mov R1,A
      POP ACC
      PUSH ACC
      ANL A,#1
      ORL A,@R1
      mov @R1,A
      DJNZ R4,DIVNB_1
      POP ACC
      JNB ACC.1,DIVNB_5
      mov A,R2
      ADD A,R1
      mov R1,A
      LCALL ADDN
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
DIVNB_5:
      CLR C
M:    JMP M
CMPN:
      mov B,R2
      mov R3,#0
      CLR C
CMPN_1:
      mov A,@R1
      SUBB A,@R0
      XCH A,R3
      ORL A,R3
      XCH A,R3
      INC R0
      INC R1
      DJNZ B,CMPN_1
      JNB OV,CMPN_2
      CPL OV
CMPN_2:
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUbb A,R2
      mov R1,A
      mov PSW,B
      mov A,R3
RET
RLAN:
      CLR C
      SJMP RLCN
RLCN:
      mov B,R2
RLCN_1:
```

```
      mov A,@R1
      RLC A
      MOV @r1,A
      INC R1
      DJNZ B,RLCN_1
      mov B,PSW
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
RET
SUBN:
      Mov B,R2
      CLR C
SUBN_1:
      mov A,@R1
      SUBB A,@R0
      mov @R1,A
      INC R0
      iNc R1
      DJNZ B,SUBN_1
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
RET
ADDN:
      mov B,R2
      CLR C
DDN_1:
      mov A,@R1
      ADDC A,@R0
      mov @R1,A
      INC R0
      INC R1
      DJNZ B,ADDN_1
      mov B,PSW
      mov A,R0
      CLR C
      SUBB A,R2
      mov R0,A
      mov A,R1
      CLR C
      SUBB A,R2
      mov R1,A
      mov PSW,B
      RET
```

5. ДВІЙКОВО-ДЕСЯТКОВА АРИФМЕТИКА

5.1. Загальні відомості

Опрацювати числа у двійково-десятковому коді можна за допомогою команди десяткової корекції DA A, яка корегує результат після попередньої операції додавання. Необхідність корекції зумовлена тим, що двійково-десяткові цифри підсумовуються на двійковому суматорі і в результаті чого виникаючий з молодшої тетради зменшує її значення на 16 (вага переносу в одиницях мо-лодшого розряду даної тетради), а не на 10, як це повинне бути при десятковому додаванні. З іншого боку, перенос із тетради суматора виникає за умови, що сума більша 15, а не 9 як при десятковому додаванні. Викликати перенос і відкоригувати двійково-десяткову цифру, як у першому, так і в другому випадку можна за рахунок додавання до даної тетради цифри 6. Зазначене додавання треба робити, якщо був перенос із даної тетради або вміст даної тетради більший 9, або цифра в старшій тетраді дорівнює 9, а в молодшій – більша 9. Для фіксації переносу з молодшої тетради байта служить ознака AC, а зі старшої – ознака C. Приклади дій над двійково-десятковими цифрами наведені нижче наступним чином.

Дії над числами				Команда	Коментар
Десятковими	Двійковими				
	CY	AC	в АЛУ і регістрах		
1	2	3	4	5	6
21 +12 33 Корекція +00	0	0	0010 0001 +0001 0010 0011 0011	ADD A,B	В акумуляторі А сума 21H+12H= 33H
	0	0	0011 0011 +0000 0000 0011 0011	DA A	В акумуляторі А двійково-десятковий код суми чисел 21+12 = 33D
24 +19 43 Корекція +06	0	0	0010 0100 +0001 1001 0011 1101	ADD A,B	В акумуляторі А сума 24H+19H=3DH
	0	1	0011 1101 +0000 0110 0100 0011	DA A	В акумуляторі А двійково-десятковий код суми 3DH = 43D
39 +78 117 Корекція +66	0	1	0011 1001 +0111 1000 1011 0001	ADD A,B	В акумуляторі А сума 39H+78H=0B1H
	1	0	1011 0001 +0110 0110 0001 0111	DA A	В акумуляторі двійково-десятковий код 2-х молодших розрядів 17D

Передбачається, що в кожному прикладі перший доданок попередньо був занесений в акумулятор А, а другий – у регістр В. У третьому прикладі результат додавання перевищує 99 і це вимагає для його запису 3-х двійково-десяткових цифр. За старший розряд (цифру) використаний біт перенесення С регістру PSW.

Для операції віднімання команди десяткової корекції не існує, тому її слід здійснювати шляхом додавання. При цьому від'ємник (наприклад, Y) подають у додатковому коді. Додатковий код десяткового числа слід знаходити за формулою

10^k-Y (k – число десяткових розрядів числа Y). Додатковий код отримують у такий спосіб:

1. Визначається число 10^k-1.

2. Із числа 10^k-1 віднімається число Y (отримуємо інверсний код).

3. Для отримання додаткового коду інверсний код збільшують на 1 (або здійснюють цю операцію при додаванні).

Приклад. Послідовність дій для визначення різниці 79D-39D=40D наступна. Спочатку отримуємо інверсний код від'ємника як 100-1-39=60H, а після цього знаходимо суму 79H+60H+1H=DAH. Далі виконуємо корекцію наступним чином: DAH+66H=140H=140D. Результат перевищує один байт, а тому старший розряд переноситься в біт C, що свідчить про те, що додатне число в акумуляторі і є результатом (коли в результаті виконаних дій біт C=0, то слід знайти модуль отриманого числа і знову здійснити корекцію).

Програма віднімання двійково-десяткових чисел

```
      MOV R1,#79H      ; занесення зменшуваного в R1
      MOV B,#39H       ; занесення від'ємника в регістр B
      MOV A,#99H
      SUBB A,B         ; отримання інверсії від'ємника
      SETB C           ; встановлення біту C для додаткового коду
      ADDC A,R1        ; додавання з урахуванням біту C
      DA A             ; десяткова корекція
      JC m1            ; перевірка знаку результату
      SETB C           ; встановлення C=1
      MOV B,A
      MOV A,#99H
      SUBB A,B         ; отримання модуля результату
      DA A             ; десяткова корекція
m1:   NOP
```

5.2. Додавання та віднімання багатобайтних чисел

При розробці програм додавання та віднімання багатобайтних двійково-десяткових чисел можна скористатися програмами для цих операцій з двійковими або шістнадцятковими багатобайтними числами. Для операцій додавання слід після команд ADD, ADDC необхідно виконати команду DA A десяткової корекції. Нижче наведена програма додавання NN-байтних чисел, для користування якою слід попередньо розташувати доданки у внутрішній оперативній пам'яті мікроконтролера. Молодші байти доданків розміщують за меншими адресами, а старші – за більшими. Адреси молодших байтів доданків передаються в підпрограму індексними регістрами R0 (адреса МЛБ першого доданку) і R1 (адреса МЛБ другого доданку), а розмір (довжина в байтах) – регістром R2. Результат виконання (сума) розміщується на місці другого доданку.

Програма додавання NN-байтних двійково-десяткових чисел

```
      MOV R2,NN  ; занесення розміру доданків в байтах
      CLR C      ; очищення прапорця перенесення перед додаванням
M_1:  MOV A,@R1  ; переміщення МЛБ 2-го доданку
      ADDC A,@R0 ; сума МЛБ доданків в акумуляторі
      DA A       ; десяткова корекція суми МЛБ доданків
      MOV @R1,A  ; занесення суми МЛБ доданків за вибраною адресою
      INC R0     ; встановлення адреси наступного байту 1-го доданку
      INC R1     ; встановлення адреси наступного байта 2-го доданку
      DJNZ R2,M_1; зменшення на 1 розміру чисел, якщо він не 0, то пе-
                 ; рехід на M_1
      PUSH 0D0H  ; збереження прапорців у стековій пам’яті
      MOV A,R0   ; занесення в А адреси останнього байту 1-го доданку
      CLR C      ; очищення прапорця перенесення С
      SUBB A,NN  ; відновлення адреси МЛБ 1-го доданку
      MOV R0,A   ; в регістрі R0
      MOV A, R1  ; занесення в А адреси останнього байту 2-го доданку
      CLR C      ; очищення прапорця перенесення С
      SUBB A, NN ; відновлення адреси МЛБ 2-го доданку
      MOV R1, A  ; в регістрі R1
      POP 0D0H   ; відновлення прапорців
```

Для операції віднімання те ж використовують операцію десяткової корекції, однак при цьому замінюють операцію віднімання на операцію додавання.

Програма віднімання багатобайтних двійково-десяткових чисел:

```
      MOV R2,NN  ; занесення розміру операндів
      SETB C     ; встановлення біту С для отримання додаткового коду
M_2:  MOV B,@R0  ; занесення МЛБ від'ємника в регістр В
      MOV A,#99H
      ADDC A,#00H; врахування біту С із попереднього циклу
      SUBB A,B   ; отримання інверсного коду МЛБ від'ємника
      ADD A,@R1  ; в акумуляторі сума МЛБ зменшуваного
      DA A       ; десяткова корекція суми
      MOV @R1,A  ; переміщення різниці у вибрану область ОЗП
      INC   R0   ; встановлення адреси наступного байта від'ємника
      INC   R1   ; встановлення адреси наступного байта зменшуваного
      DJNZ R2,M_2; зменшення на 1 розміру чисел (коли він не нуль, то
                 ; перехід на M_2)
      PUSH 0D0H  ; збереження прапорців
      MOV A,R0   ; в акумуляторі адреса СТБ від'ємника
      CLR C ; встановлення в нуль ознаки перенесення С
      SUBB A,NN  ; відновлення початкової адреси МЛБ від'ємника
      MOV R0,A   ; в регістрі R0
      MOV A,R1   ; в акумуляторі адреса СТБ зменшуваного
      CLR C ; встановлення в нуль ознаки перенесення С
      SUBB A,NN  ; відновлення початкової адреси МЛБ зменшуваного
      MOV R1, A  ; в регістрі R0
      POP 0D0H   ; відновлення прапорців
```

Запитання для самоперевірки

- Які параметри арифметичних операндів слід задавати перед розробкою програмного забезпечення мікроконтролера?
- Чим визначається довжина чисел?
- Як можна врахувати знак чисел при їх різній довжині?
- Які межі змінювання чисел при залежно від їх довжини та типу?
- Як використовують ознаку перенесення С при операціях додавання?
- Як і коли використовують ознаку перенесення С при операціях віднімання?
- Яким чином можна позбутися операції віднімання цілих чисел без знаку?
- Яким чином можна отримати інверсний код багатобайтного числа?
- Як можна отримати додатковий код багатобайтного відємного числа?
- Яким чином слід інтерпретувати значення ознаки перенесення С в операціях додавання додаткових кодів?
- У чому основна відмінність програм віднімання чисел від програм їх додавання?
- Чи можна користуватися стандартною командою множення однобайтних без-знакових чисел для здійснення операції множення багатобайтних чисел?
- Яка розрядність добутку двох однобайтних чисел?
- Де і в якому порядку розташований добуток двох однобайтних чисел?
- Яка розрядність добутку двох однобайтних чисел, двобайтних?
- У чому основна особливість множення однобайтних чисел зі знаком?
- У чому основна особливість множення двобайтних чисел?
- Чи можливо використати програми отримання добутку багатобайтних операндів без знаку для отримання добутку операндів зі знаком?
- Чи можливе багатократне множення операндів без істотного збільшення розміру програми і якщо так, то завдяки чому?
- Яким чином можна здійснити операцію ділення однобайтних без знакових чисел, де буде розташований результат і який його розмір?
- У чому особливість ціло-чисельного ділення двобайтних чисел?
- Який результат ділення двобайтних чисел і який його розмір?
- У чому особливість двійково-десяткової форми операндів?
- Які є стандартні засоби переведення шістнадцяткової форми результату в двійково-десяткову форму?
- Які ознаки регістру стану мікроконтролера використовуються при десятковій корекції операції додавання?
- Як можна отримати двійково-десяткову форму різниці?
- У чому основні засади операції додавання та віднімання двійково-десяткових чисел?

Індивідуальні завдання до самостійної роботи

Для самоперевірки рівня опанування матеріалу для самостійної роботи слід виконати наступну послідовність дій:

- згідно власного порядкового номеру у журналі групи та варіантів наведених у методичних вказівках індивідуальних завдань до виконання лабораторної роботи *попередньо* визначитися для себе з рівнем складності завдання;
- на основі вибраного індивідуального варіанту завдання до виконання лабораторної роботи слід детально і *самостійно* ознайомитися з необхідними для виконання завдання прикладами програмного забезпечення, наведеними у методичних вказівках (програми прикладів можна отримати у викладача);
- модифікувати вибрані програми до вигляду підпрограм, які слід викликати у визначеній завданням послідовності та впевнитися у їх працездатності за допомогою IDE MPASS/51/251;
- підготувати з урахуванням ГОСТ алгоритм і основне програмне забезпечення для виконання лабораторної роботи та попередній звіт до неї.

Індивідуальні завдання до виконання лабораторної роботи

Низького рівня складності

Для отримання оцінки «задовільно» слід скласти і налагодити програмне забезпечення відповідно індивідуального завдання до самостійної роботи, особистому номеру у журналі групи та даним нижче наведеної таблиці. Для цього рівня складності всі числа обнобайтні, шістнадцяткові і зі знаком (можуть бути від’ємними).

Середнього рівня складності

Для отримання оцінки «добре» довжина операндів попереднього завдання збільшується до двох байтів. Допускається зменшення кількості операцій у виразі з відповідним зменшенням оцінки (використання двійково-десяткових операндів підвищує оцінку на 1 бал).

Підвищеного рівня складності

Для отримання оцінки «відмінно» довжина операндів попереднього завдання перевищує два байти, а інші вимоги і умови зменшення кількості операцій у виразі співпадають із вимогами до завдання для отримання оцінки «добре» (використання двійково-десяткових операндів підвищує оцінку на 1 бал).

Таблиця арифметичних виразів для індивідуальних завдань до виконання лабораторної роботи

Варіант	Вираз
1	2
1	(A+B-C)*D/A
2	A+B/D-C*D
3	(A-B)*D/(C+A)
4	A+B*C/D-D
5	(A*B-D+C)/B
6	(A-B+C)/D*B
7	A/B+C*D-A
8	A*B-C/D+C
9	8A-(D*C)/B+B
10	(A+B)/(C-D)*A
11	A2-B/C+D
12	A/B+C*D-B
13	(A+B)/C-D*A
14	A*B-B/C+D
15	(A+B)/C-D
16	A/(B+C-D*A)
17	A-B*C+D/A
18	A*B+D-C/A
19	A/B-C*D+A
20	(A*C+B)/D-C
21	A/B*C-D+B
22	2*A-B+C/D
23	A-2*B+C/D
24	A-B*2+C/D
25	A*B/C-D+A

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. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропроцессорах. – М.: Высш. шк., 1991. – 303 с.
5. Однокристальные микроЭВМ: Справочник / А.В. Бобрыкин, Г.П. Липовецкий, Г.В. Литвинский и др. – М.: МИКАП, 1994. – 400 с.
6. 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.
7. 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.
8. MCS-51 and MCS-96 Packaging Information. November 1994. Order Number: 272118-001. Intel Corporation, 1995, 16 p.
9. 8XC51GB CHMOS Single-Chip 8-Bit Microcontroller. Preliminary. November 1994. Order Number: 272337-002. Intel Corporation, 1995, 22 p.
10. Фрунзе А.В., Фрунзе А.А. Микроконтроллеры ? Это же просто! М.: ООО «ИД СКИМЕН», т.3.–2003.–224 с.

Упорядники:
Кириченко Віталій Іванович
Яланський Олексій Анатолійович
КОВШОВ Кирило Валерійович

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ МПП-4 «ПРОГРАМУВАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ПРИ ВИКОРИСТАННІ МІКРОКОНТРОЛЕРІВ 8051АН СІМЕЙСТВА MCS-51», 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.