Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Основи мікропроцесорної техніки

Методичні вказівки
до виконання лабораторних робіт
для студентів напрямів підготовки
6.050803 «Акустотехніка»
та 6.050903 «Телекомунікації»
усіх форм навчання

Затверджено Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2008

Основи мікропроцесорної техніки [Текст] : метод. вказівки до викон. лаборатор. робіт для студ. напрямів підготов. 6.050803 «Акустотехніка» та 6.050903 «Телекомунікації» усіх форм навчання / Уклад.: В. Б. Швайченко, К. О. Трапезон, О. П. Чеверда. – К.: НТУУ «КПІ», 2008. – 60 с.

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № 9 від 22.05.2008 р.)

Навчальне видання

Основи мікропроцесорної техніки

Методичні вказівки
до виконання лабораторних робіт
для студентів напрямів підготовки
6.050803 «Акустотехніка»,
6.050903 «Телекомунікації»
усіх форм навчання

Укладачі: *Швайченко Володимир Борисович*, канд. техн. наук, доц.
Трапезон Кирило Олександрович
Чеверда Олександр Петрович

Відповідальний редактор *Г. М. Розорінов*, д-р техн. наук, проф.

Рецензент *Т. О. Терещеко*, д-р техн. наук, проф.

За редакцією укладачів
Надруковано з оригінал-макета замовника

Темплан 2008 р., поз. 2-053

Підп. до друку 13.06.2008.Формат 60×84 1/16. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 3,49. Обл.-вид. арк. 5,8. Зам №3164.Наклад 150 пр.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул. Політехнічна, 14, корп. 15
тел./факс (044) 241-68-78

ЗМІСТ

ВСТУП

Даний курс лабораторних робіт призначений для отримання початкових практичних навичок роботи з мікроконтролерами серії C8051F120DK фірми Silicon Laboratories.

В якості лабораторного стенду використовується налагоджувальний модуль, який працює на база мікроконтролера. Цей модуль дозволяє проводити налагодження програмного й апаратного забезпечення мікроконтролерів на базі ВІС (великої інтегральної схеми). На лабораторному стенді також можна вивчати структури системи команд, схемотехнічне і програмне забезпечення, порядок функціонування а також методи програмування мікроконтролерів.

ЗАГАЛЬНІ ВИМОГИ, ЩОДО ОФОРМЛЕННЯ ЛАБОРАТОРНИХ РОБІТ

(для студентів денної форми навчання)

Протокол лабораторної роботи повинен мати такі структурні елементи:

1) Титульний аркуш протоколу до лабораторної роботи, на якому треба обов'язково зазначити:
   - назва вузу, кафедри де проводиться лабораторна робота;
   - прізвище, ім'я та по-батькові студента, що виконав лабораторну роботу;
   - прізвище керівника лабораторних занять.
2) Мета роботи.
3) Відповіді на контрольні запитання.
4) Схеми, таблиці, діаграми, текст програм, необхідні рисунки.
5) Лістінг програми.
6) Висновки за результатами виконання лабораторної роботи.

ЛАБОРАТОРНА РОБОТА №1

Вивчення архітектури мікроконтролера C8051F120, його основних параметрів, ознайомлення з побудовою розподілу пам'яті пристрою, вивчення особливостей регістрів загального та спеціального призначень

Мета роботи – Ознайомлення з принципами побудови та роботи мікроконтролерів серії C8051F12x фірми SiLabs в різних режимах. Вивчити основні параметри мікроконтролера C8051F120 та призначення основних виводів пристрою.

Домашнє завдання

1. *Ознайомитися з описами мікроконтролерів, його типовими структурам, принципами підключення зовнішньої пам'яті та різноманітних периферійних пристроїв.*
2. *Вивчити двійкову, десяткову та 16-кову системи числення.*
3. *Визначити які інтерфейси використовуються в типових схемах мікроконтролерів та які особливості вони мають.*

Теоретичні відомості

Мікроконтролер – обчислювальна контрольно-вимірювальна або керуюча система, основним пристроєм в яких є цифровий пристрій (процесор).

МК C8051F120 використовує розроблене фірмою Silicon Labs процесорне ядро CIP-51, яке за системою команд повністю сумісно з ядром

MCS-51TM. Для розробки програмного забезпечення можуть використовуватися стандартні 803x/805x асемблери та компілятори.

Ядро містить периферію, що відповідає стандарту 8051, включаючи п'ять 16-розрядних таймерів/лічильників, 256 байт внутрішнього ОЗП, 128 байт адресного простору регістрів спеціального призначення, а також 8/4 8-розрядних порту вводу/виводу. CIP-51 використовує конвеєрну архітектуру, що істотно підвищує швидкість виконання команд в порівнянні із стандартною архітектурою 8051.

Мікроконтролер C8051F120 є повністю інтегрованою на одному кристалі системою для обробки змішаних (аналого-цифрових) сигналів, які мають 32 (МК в корпусі 64TQFP) цифрових входів/виходів.

Цей МК має наступні особливості:

- високопродуктивне мікропроцесорне ядро CIP-51 з конвеєрною архітектурою, сумісне із стандартом 8051 (з продуктивністю 100 або 50 MIPS);
- вбудовані засоби відладки, які забезпечують внутрисистемну та відладку в режимі реального часу;
- точний 12-розрядний АЦП (продуктивність – 100 тис. перетворень в секунду) з програмованим підсилювачем і 8-канальним аналоговим мультиплексором;
- точний 8-розрядний АЦП (максимальна продуктивність – 500 тис. перетворень в секунду) з програмованим підсилювачем і 8-канальним аналоговим мультиплексором;
- два 12-розрядні ЦАП з програмованим оновленням вихідного сигналу;
- помножувач-акумулятор 16×16, який потребує 2 такти SYSCLK для виконання операції множення;

- 128 кбайт Flash-пам'яті, що програмується внутрішніми системними засобами;
- 8448 (8k + 256) байт вбудованого ОЗП;
- інтерфейс зовнішньої пам'яті даних з доступним адресним простором 32 кбайт;
- апаратний реалізовані послідовні інтерфейси I^2C/SMBus, SPI і два УАПП;
- п'ять 16-розрядних таймерів загального призначення;
- програмований масив лічильників/таймерів (ПМС) з шістьма модулями захоплення/порівняння;
- вбудований сторожовий таймер, схема стеження за напругою живлення і датчик температури.

На рисунку 1 наведена структурна схема МК. Центральний процесор має регістри загального та спеціального призначення.

Молодші 32 байти пам'яті даних (0x00 - 0x1F) розбиті на **чотири банки регістрів загального призначення**. Кожен банк складається з восьми однобайтових регістрів, що позначаються **R0-R7**. У конкретний момент часу може бути активний лише один банк, визначений бітами RS0 (PSW.3) і RS1 (PSW.4) у слові стану програми (program status word) PSW. Це дозволяє здійснювати швидке перемикання контексту при виклику підпрограм та процедур обробки переривань. Режими непрямої адресації використовують регістри R0 і R1 як індексні регістри.

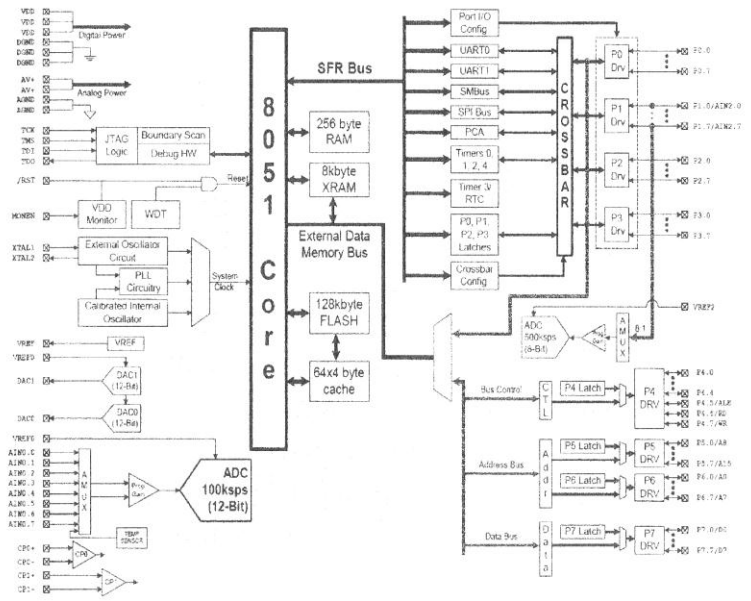

Рисунок 1 – Структурна схема МК C8051F120

Елементи пам'яті даних з адресами від 0x80 до 0xFF, доступні в режимі прямої адресації, і утворюють **регістри спеціального призначення** (special function registers - **SFR**). SFR дозволяють керувати ресурсами ядра CIP-51 і периферійними модулями, а також здійснювати обмін даними з ними. CIP-51 дублює SFR типові для архітектури 8051, і містить додаткові SFR, які використовуються для настройки підсистем, унікальних, для даного сімейства МК, і доступу до них. Це дозволяє реалізувати нові можливості збереження сумісності з системою команд MCS-51.

У CIP-51 використовується сторінкова організація SFR, що дозволяє відображати в адресному просторі 0x80 - 0xFF велику кількість регістрів SFR. Простір пам'яті SFR має 256 сторінок.

Таким чином, кожен елемент пам'яті з області 0x80 - 0xFF може адресувати до 256 регістрів SFR. У МК сімейства C8051F12x використовуються п'ять SFR сторінок: 0, 1, 2, 3 і F. SFR сторінки вибираються за допомогою регістра вибору сторінки SFR SFRPAGE. Послідовність дій при зчитуванні і записі SFR наступна:

1. Вибрати номер відповідної SFR сторінки, використовуючи регістр SFRPAGE.

2. Прочитати або записати регістр SFR, використовуючи режим прямої адресації (команда MOV).

Процесорне ядро має наступні основні регістри SFR:

1) два 8-розрядних регістри - акумулятори A та B.
2) 16-розрядний регістр-вказівник даних DPTR. Цей регістр використовується для доступу в режимі непрямої адресації до пам'яті XRAM та Flash, і складається з двох байтів: DPL – молодший, DPH – старший.
3) Покажчик стека SP містить адресу вершини стека. Покажчик стека інкрементується перед кожною операцією PUSH. Стек являє собою послідовність комірок пам'яті, організованих за принципом LIFO (Last In First Out), що дозволяє зберігати дані при перериваннях і викликах підпрограм. Щоразу при додаванні в стек нового байта значення регістра SP зменшується, а при витягу зі стека збільшується.
4) Регістр ознак PSW являє собою 8-розрядний регістр, біти якого відбивають стан системи АЛУ після виконання чергової команди. Ці біти можна роздільно протестувати в ході виконання програми і, залежно від результату, почати певні дії:

– Біт 7: CY: Прапор переносу. Цей біт встановлюється, якщо в результаті останньої арифметичної операції відбувся перенос (складання)

або займ (віднімання). Він скидається у 0 всіма іншими арифметичними операціями.

– Біт 6: AC: Прапор десяткового переносу. Цей біт встановлюється, якщо в результаті останньої арифметичної операції відбувся перенос (складання) в старшу тетраду або займ (віднімання) із старшої тетради. Він скидається в 0 всіма іншими арифметичними операціями.

– Біт 5: F0: Прапор користувача 0. Це доступний в бітовому режимі адресації прапор загального призначення, призначений для використання під керуванням програми.

– Біти 4-3: RS1-RS0: Біти вибору банку регістрів. Ці біти визначають активний банк регістрів. RS1 RS0 Банк регістрів Адреси:

0 0 0 0x00-0x07

0 1 1 0x08-0x0F

1 0 2 0x10-0x17

1 1 3 0x18-0x1F

– Біт 2: OV: Прапор переповнення. Цей біт встановлюється у 1 в наступних випадках:

- якщо в результаті виконання команди ADD, ADDC або SUBB відбулося переповнення з

зміною знаку;

- якщо в результаті виконання команди MUL відбулося переповнення (результат перевищує

значення 255);

- якщо при виконанні команди DIV відбулося ділення на нуль.

- Біт OV скидається в 0 командами ADD, ADDC, SUBB, MUL і DIV у всіх інших випадках.

– Біт 1: F1: Прапор користувача 1. Це доступний в бітовому режимі адресації прапор загального призначення, призначений для використання під управлінням програми.

– Біт 0: PARITY: Прапор парності(тільки для читання). Цей біт встановлюється в 1, якщо сума восьми біт в акумуляторі непарна і скидається, якщо сума парна.

5) Програмний лічильник PC є 8-розрядним регістром, що містить адресу наступної виконуваної команди.

Система внутрішнього процесора і схема зв'язку регістрів представлена на рисунку 2.

Карта пам'яті

CIP-51 має стандартну (8051) структуру адресного простору пам'яті програм і даних. У склад пам'яті входить ОЗП об'ємом 256 байт, старші 128 байт якого мають подвійну конфігурацію. У режимі непрямої адресації здійснюється доступ до старших 128 байтів ОЗП загального призначення, а в режимі прямої адресації здійснюється доступ до 128 байтів адресного простору регістрів спеціального призначення (SFR). Молодші 128 байт ОЗП доступні в режимі як прямою, так і непрямої адресації. З них перші 32 байти адресуються як чотири банки регістрів загального призначення, а наступні 16 байт адресуються побайтно або побітно. Всі МК мають вбудований блок 8-Кбайтного ОЗП. До цього вбудованого 8-кбайтному блоку пам'яті можна звертатися у всьому діапазоні адрес 64 кбайтної зовнішньої пам'яті даних (з перекриттям адрес по 8-кбайтним межам).

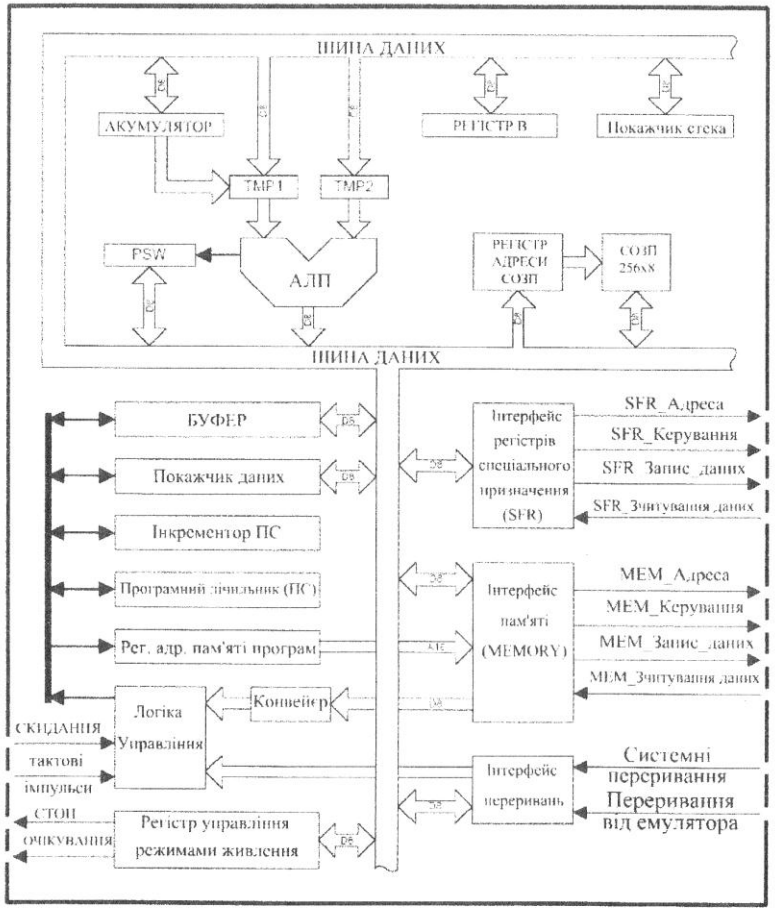

Рисунок 2 – Структурна схема процесорного ядра CIP-51

Всі МК мають також інтерфейс зовнішньої пам'яті (external memory interface - EMIF) для доступу до зовнішньої пам'яті даних або до периферійних модулів, відображених на цю пам'ять. На адресне простір

зовнішньої пам'яті даних може бути відображений або тільки вбудована пам'ять, або тільки зовнішня пам'ять, або їх комбінація (адреси до 8Кбайт відносяться до вбудованої пам'яті, адреси зверху 8 кбайт відносяться до EMIF). Пам'ять програм складається з 128 Кбайт розділеної на банки Flash-пам'яті. 1024 байт пам'яті з адресами 0x1FC00 - 0x1FFF зарезервовані.

Пам'ять програм МК C8051F120 складається з 64 Кбайт Flash-пам'яті. Ця пам'ять може бути легко перепрограмована внутрішньосистемно секторами по 1024 байт, не вимагаючи при цьому спеціальної зовнішньої напруги програмування. У всіх МК є також два 128-байтних сектори пам'яті, що займають адреси від 0x20000 до 0x200FF, які можуть використовуватися програмою користувача для зберігання даних. Карта пам'яті наведена на рис.3.

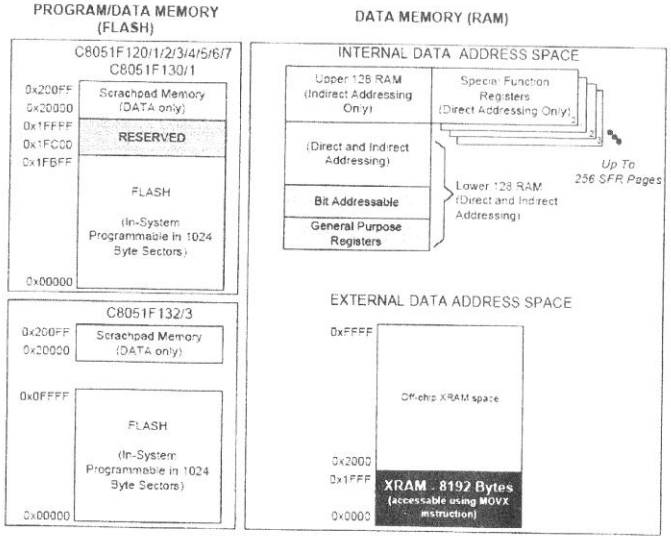

Рисунок 3 – Карта пам'яті C8051F120

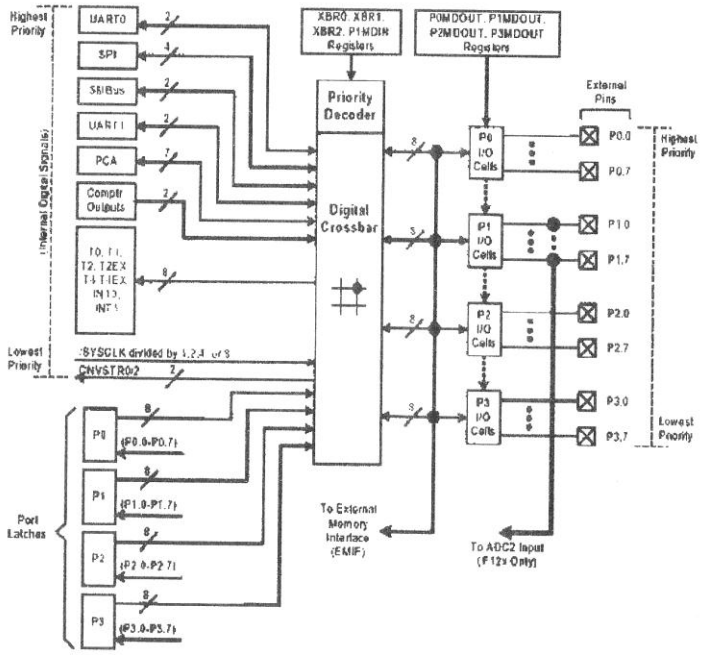

Рисунок 4 – Структурна схема портів вводу/виводу інформації

Комплекс налагодження

МК має вбудований інтерфейс граничного сканування і відладчик, які за допомогою 4-х дротяного інтерфейсу JTAG дозволяють здійснювати в режимі реального часу неруйнуючу (не використовуються внутрішні ресурси) внутрішньосхемну відладку, використовуючи МК, встановлений в кінцевий виріб (рис. 5). За допомогою JTAG інтерфейсу, повністю

сумісного з протоколом IEEE 1149.1, здійснюється граничне сканування, яке використовується для тестування і виробничих випробувань.

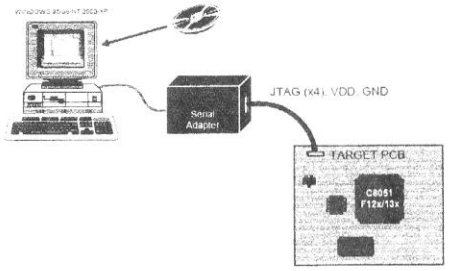

Рисунок 5 – Модуль налагодження програм

Контрольні запитання

1. Що таке мікроконтролер?
2. Що розуміють під поняттям "інтерфейс користувача" ?
3. Яким чином в програмах можуть бути представлені операнди?
4. Яку роль в мікроконтролері виконують регістри спеціального призначення?
5. Структурна схема 8-розрядного мікроконтролера. Призначення основних блоків.
6. Яку організацію памяті має C8051F120?
7. За якою технологією побудована стекова пам'ять в мікроконтролері?
8. Скільки в мікроконтролері паралельних портів, і які з них використовуються для вводу даних, а які для виводу?
9. Яким чином розподілена пам'ять в мікроконтролері ?

10. Що означає сторінкова організація пам'яті ?
11. Які основні функції виконує програмний лічильник, що входить до складу процесорного ядра МК C8051F120 ?
12. Яку основну роль відіграє інтерфейс JTAG при налагодженні програм в комплексі функціонування МК C8051F120 ?
13. Які основні функції покладаються на універсальний асинхронний прийомо-передавач (UART) в складі МК C8051F120 ?
14. Послідовний інтерфейс I^2C/SMBus. Призначення, основні функції та особливості реалізації.
15. Послідовний інтерфейс SPI. Призначення, основні функції та особливості реалізації.
16. Які основні інтерфейси використовуються при організації процесорного ядра ? Наведіть їх основне призначення та функціональні особливості.
17. Який адресний простір при організації па'яті відведено до потреб користувача в МК C8051F120 ?
18. Для чого використовується регістр-вказівник даних DPTR в процесорному ядрі МК C8051F120 ? Який об'єм має цей регістр та яким він організований в МК ?
19. Для чого необхідні переривання і яким чином вони реалізовані при роботі МК C8051F120 ?
20. Для чого використовуються таймери загального призначення в МК C8051F120 ?

ЛАБОРАТОРНА РОБОТА №2

Вивчення особливостей, синтаксису окремих команд а також отримання навиків щодо їх практичного застосування в інтегрованому програмному середовищі (IDE) Silicon Labs.

Мета роботи – Вивчити основні команди програми IDE Silicon Labs, порядок їх запровадження та виконання. Ознайомитись з принципами організації контролю працездатності налагоджувального модулю.

Домашнє завдання

Вивчити послідовність та способи налаштування МК за допомогою IDE Silicon Labs.

Теоретичні відомості

Інтерфейс користувача IDE Silicon Labs складається з трьох головних вікон (див. рис.6):

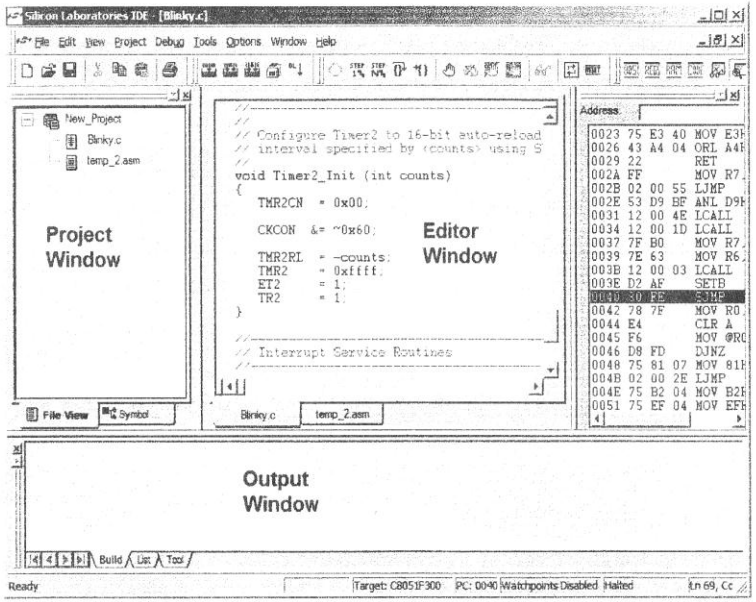

Рисунок 6 – Інтерфейс користувача IDE Silicon Labs

- Вікно проектів (Project Viewer window)
  - File View - використовується, для того щоб переглядати та здійснювати керування файлами, пов'язаними з проектом.
  - Symbol View - потрібне, аби переглядати адреси символів, що використовуються в проекті.
- Вікно редагування/налагодження (Edit/Debug window)
  - Використовується, щоб компонувати або редагувати обраний файл в межах проекту.
  - Після того, як код був завантажений, це вікно використовується, для перегляду кодового виконання протягом налагоджувальної сесії.

- Вікно виводу (Output window) - це вікно, що використовуються для висвітлення інформації від різних процесів протягом їхнього виконання.
    - Консоль виконання (build tab) показує звіти різних компонентів, що підключаються для відладки. Користувач може двічі кликнути по помилці у вікні build tab, і строчка коду, де знаходиться помилка, буде показана в редакторі.
    - Консоль списку (list tab) показує файл списку значень констант та адрес МК після виконання програми.
    - Консоль інструменту (tool tab) показує вихідні дані, якщо файл виводу названий "tool.out.

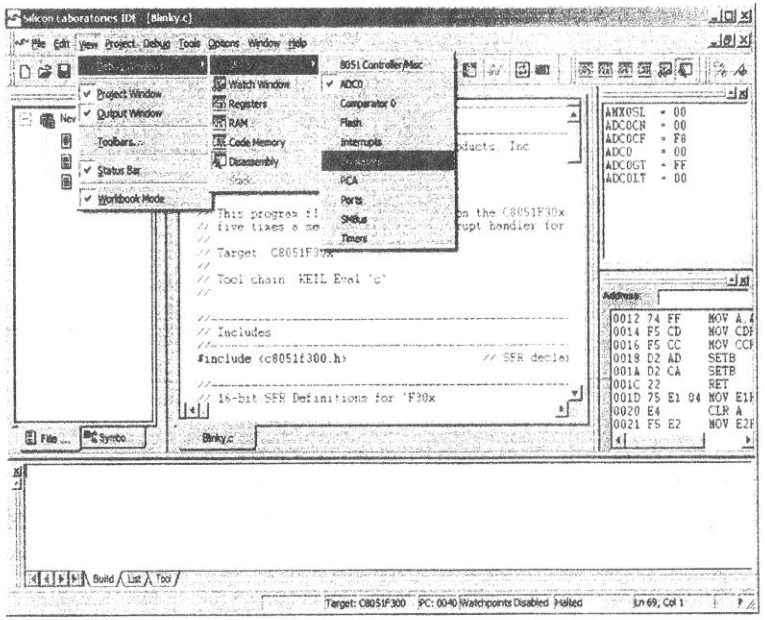

Рисунок 7 – Вікна відладки

Для налагодження програми та перегляду змін значень регістрів під час програмної сесії IDE включає багато налагоджувальних вікон. Налагоджувальні вікна можуть бути доступні з меню "View" програми:

– вікна SFR - вікна, що розбиті по функціональних групах. Перелік груп залежить від зовнішніх пристроїв, доступних при підключенні в даний час, наприклад: АЦП ADC0, ЦАП DAC0, Flash-пам'ять, переривання, PCA, порти, SMBus, SPIBus та таймери;

– вікно спостереження;

– регістри;

– оперативна пам'ять;

– кодова пам'ять;

– деассемблювання команд;

– зовнішня пам'ять;

– надоперативна пам'ять;

– стек;

– кеш пам'яті;

– інформація щодо трасування програм.

Особливості функціонування IDE Silicon Labs

1) Значення регістра може бути змінено, якщо над ним розташувати курсор та надрукувати бажане значення над існуючим. Нове значення буде завантажено до технічних засобів до моменту виконання коду. Для продовження роботи з новими значеннями треба натиснути кнопку "Refresh". Після цього всі змінені значення будуть записані до емулятора. Вікно регістра буде завантажене до емулятора і значення у вікнах будуть оновлені зі всіма заміненими значеннями, вказаними в червоному кольорі.

ПРИМІТКА. Значення регістра можуть бути змінені тільки, якщо відладчик знаходиться в даний час у положенні "Зупинка".

2) Для використання системи команд С51 та регістрів процесора необхідно завчасно підключити відповідні бібліотеки, наприклад:

$include (c8051f120.inc),

де c8051f120.inc – головна бібліотека, яка вміщує в собі дані, адреси всіх бітів та регістрів пам'яті процесора.

3) Програма повинна закінчуватись командою **END.**

4) Для завантаження або стирання кодового простору процесора необхідно виконати функцію Erase Code Space, а для завантаження коду – Download Code.

Робоче завдання

1. Ознайомитись з послідовністю виконання та запису коду програми у мікроконтролер.
2. Завантажити у МК тестову програму Blink та прослідити за зміною значень у регістрах у покроковому режимі.
3. Змінити вміст комірки пам'яті, яку вказує викладач заняття та перевірити виконання програми за допомогою вікон відладки.
4. Сформувати текстовий файл з програмою на мові Асемблера за допомогою текстового редактора:

```
CLR A
MOV A, #03h
MOV B, #02h
```



```
MOV R1,A
NOP
END
```

5. Виконати асемблювання програми та завантаження її до пам'яті МК, переглянути вміст чарунків пам'яті МК за допомогою вікон відладки.
6. Встановити точку зупинки на команді NOP за допомогою команди BREAKPOINT.
7. Запустити програму на виконання з ненульової адреси $0100.

Результати виконання робочого завдання відобразити у протоколі.

Контрольні запитання

1. Призначення та основні функціональні можливості програми-монітору IDE Silicon Labs.
2. Яке повідомлення виводиться на екран у випадку виконаної невірної операції?
3 Як інтегрована IDE Silicon Labs з мовами інших рівнів?
4. Яку особливість обробки командних рядків програми на мові Асемблера має програма-монітор IDE Silicon Labs.
5. Якими обмеженнями характеризується програма-монітор IDE Silicon Labs ?
6. Які дії необхідно виконати аби мати можливість використовувати команди системи С51 ?
7. Що відбувається зі змістом регістрів загального та спеціального призначень при трасуванні команд тестової програми Blink ?

8. Який порядок дій можна визначити вводі команд програми, у відповідності з її лістингом, та отриманні результатів прогарами в програмі-монітору IDE Silicon Labs ?

ЛАБОРАТОРНА РОБОТА №3

Ознайомлення з особливостями застосування системи команд передачі даних мікроконтролера C8051F120, вивчення режимів адресації.

Мета роботи – Вивчити основні команди передачі даних, що входять до системи команд мікроконтролера, порядку їх запровадження та виконання. Розібратися з режимами адресації, які використовуються при побудові програм на мові Асемблера в МК C8051F120 .

Домашнє завдання

1. Ознайомитися оглядово з основними групами команд, які утворюють систему команд мікро-контролера C8051F120.

2. Вивчити режими адресації мікроконтролера. Особливо звернути увагу щодо особливостей використання таких режимів адресації: відносна, абсолютна, широка, індексна.

3. Вивчити основні команди передачі даних.

Теоретичні відомості

Режими адресації

Мікроконтролер C8051F120 дозволяє реалізувати 8 режимів адресації:

- регістрова адресація (register addressing);
- пряма адресація (бітова та байтова) (direct addressing);
- неявна адресація (indirect addressing);

- безпосередньо-регістрова адресація (immediate constant addressing);
- відносна адресація (relative addressing);
- абсолютна адресація (absolute addressing);
- широка адресація (long addressing);
- індексна адресація (indexed addresing).

Розглянемо деякі режими адресації з наведеного переліку.

1) **Неявна адресація** означає, що вся інформація щодо операндів визначається кодом команди. А якщо і є операнди, то це обов'язково регістри; при цьому посилання на пам'ять відсутнє. Команди з неявною адресацією, як правило складаються, з одного або двох байт.

Тобто, команди з неявною адресацією або не мають операндів, або містять вказівку на операнд у мнемоніці команди.

Наприклад:

RET

Пояснення: Команда повернення з підпрограми.

2) **Регістрова адресація** використовується для звернення до восьми робочих регістрів вибраного банку робочих регістрів, а також для звернення до регістрів A, B, DPTR, і до прапорця переносу C. Номер регістру записується трьома молодшими бітами команди.

Наприклад:

MOV R5, A

Пояснення: В першому операнді застосовується регістрова адресація, а в другому – неявна.

3) **Пряма байтова адресація** використовується для звернення до елементів внутрішньої пам'яті (ОЗП) даних (адреси 0...127) і до регістрів спеціального призначення (адреси 128...256). Адреса елементу пам'яті міститься в другому байті команди.

Наприклад:

MOV A, 20h

Пояснення: У другому операнді використана пряма байтова адресація, а в першому - неявна

MOV 15h,R6

Пояснення: В першому операнді використана пряма байтова адресація, а в другому - регістрова

Пряма бітова адресація використовується для звернення до 128 біт, що окремо адресуються, і розташованих в осередках з адресами 20H-2FH.

Наприклад:

SETB 20h

Пояснення: Використовується пряма бітова адресація

CLR 15h

Пояснення: Використовується пряма бітова адресація

4) **Беспосередньо-регістрова адресація** використовується для звернення до осередків внутрішнього ОЗП даних.

Як регістри-покажчики (адреси) використовуються регістри R0, R1 вибраного банку регістрів.

Наприклад:

MOV A,@R0

Пояснення: В першому операнді задіяна неявна адресація, а в другому – безпосередньо-регістрова.

MOV @R1,A

Пояснення: В першому операнді застосовується безпосередньо-регістрова адресація, а в другому –неявна

Беспосередньо-регістрова адресація використовується також для звернення до зовнішньої пам'яті даних. В цьому випадку за допомогою регістрів-покажчиків R0 і R1 (робочого банку робочих регістрів) обирається осередок з блоку 256 байт зовнішньої пам'яті даних. Номер блоку заздалегідь задається вмістом порту P2.

Наприклад:

MOVX A,@R0

Пояснення: В першому операнді використана неявна адресація, а в другому – безпосередньо-регістрова.

MOVX @R1,A

Пояснення: В першому операнді використана безпосередньо-регістрова адресація, а в другому – неявна.

Якщо в якості регістра-покажчика використовується 16-розрядний покажчик даних (DPTR), то можна вибрати будь-який елемент зовнішньої пам'яті даних розміром до 64 кбайт.

Наприклад:

MOVX A,DPTR

Пояснення: В першому операнді використана неявна адресація, а в другому – безпосередньо-регістрова.

MOVX DPTR,A

Пояснення: В першому операнді застосовується безпосередньо-регістрова адресація, а в другому – неявна.

Безпосередньо-регістрова адресація за сумою базового та індексного регістрів (вміст акумулятора A) спрощує запит таблиць, записаних в пам'яті програм. Будь-який байт з таблиці може бути вибраний за адресою, яка визначається сумою вмісту DPTR або PC та вмісту A.

Наприклад:

MOV A @A+PC

Пояснення: В першому операнді використана неявна адресація, а в другому – безпосередньо-регістрова.

MOV A @A+DPTR

Пояснення: В першому операнді задіяна неявна адресація, а в другому – безпосередньо-регістрова.

Також цей режим адресації дозволяє вибрати з адресного простору пам'яті програм константи, які явно вказані в команді.

Наприклад:

MOV A, #14h

Пояснення: В першому операнді використана неявна адресація, а в другому – безпосередня.

MOV DPTR, #2048h

Пояснення: В першому операнді використана неявна адресація, а в другому – безпосередня.

Команди передачі даних

Команди передачі даних дозволяють виконати такі операції з операндами або регістрами:

– завантаження операндів з пам'яті даних у регістри;

– запис вмісту регістрів у пам'ять;

– операції передачі даних між регістрами.

Основні команди передачі даних та їх тлумачення наведено у таблиці 1.

Таблиця 1 – Основні команди передачі даних

Мнемоніка команди	Тлумачення
MOV A,R_n	Пересилання даних в акумулятор з регістра (n = 0...7)
MOV A,direct	Пересилання даних в акумулятор байта, що прямо-адресується
MOV A,@Ri	Пересилання в акумулятор байта ОЗП, що непрямо-адресується
MOV A,#data	Завантаження в акумулятор константи
MOV R_n,A	Пересилання даних в регістр з акумулятора
MOV R_n,direct	Пересилання даних в регістр байта, що прямо-адресується
MOV R_n,#data	Завантаження в регістр константи
MOV direct,A	Пересилання даних за прямою адресою у акумулятор
MOV direct,R_n	Пересилання даних за прямою адресою у регістр
MOV direct,direct	Пересилання байту, що прямо-адресується, за прямою адресою

Продовження таблиці 1 – Основні команди передачі даних

Мнемоніка команди	Тлумачення
MOV direct,@Ri	Пересилання байту ОЗП, що непрямо-адресується, за прямою адресою
MOV direct,#data	Пересилання за прямою адресою константи
MOV @Ri,A	Пересилання даних з осередка ОЗП в акумулятор
MOV @Ri,direct	Пересилання даних з осередку ОЗП байта, що прямо-адресується, у комірку що непрямо-адресується
MOV @Ri,#data	Пересилка в осередок ОЗП константи, що непрямо-адресується
MOV DPTR,#data16	Завантаження покажчика даних
MOVC A,@A+PC	Пересилання в акумулятор байта з пам'яті програм
MOVX A,@Ri	Пересилання в акумулятор байта із зовнішньої пам'яті даних
MOVX @Ri,A	Пересилання байта з акумулятора в зовнішню пам'ять даних
MOVX A,@DPTR	Пересилання даних в акумулятор з розширеної зовнішньої пам'яті даних
MOVX @DPTR,A	Пересилання даних з акумулятора в розширену зовнішню пам'ять даних
PUSH direct	Завантаження у стек
POP direct	Вивантаження із стеку
XCH A,R_n	Обмін даними акумулятора з регістром

Продовження таблиці 1 – Основні команди передачі даних

Мнемоніка команди	Тлумачення
XCH A,direct	Обмін даними акумулятора з байтом, що прямо-адресується
XCH A,@Ri	Обмін даними акумулятора з байтом ОЗП, що непрямо-адресується
XCHD A,@Ri	Обмін даними молодшої тетради акумулятора з молодшою тетрадою байта ОЗП, що непрямо-адресується

Арифметичні операції

Таблиця 2 – Основні арифметичні команди з даними

Мнемоніка команди	Тлумачення
ADD A,R_n	Операція підсумовування вмісту акумулятора та регістра (n = 0...7)
ADD A,direct	Операція підсумовування вмісту акумулятора та байта, що прямо-адресується
ADD A,@Ri	Операція підсумовування вмісту акумулятора та байта ОЗП, що непрямо-адресується
ADD A,#data	Операція підсумовування вмісту акумулятора з константою
ADDC A,R_n	Операція підсумовування акумулятора та регістру і переносу
ADDC A,direct	Операція підсумовування вмісту акумулятора та байту, що прямо-адресується, переносом

Продовження таблиці 2 – Основні арифметичні команди з даними

Мнемоніка команди	Тлумачення
ADDC A,@Ri	Операція підсумовування вмісту акумулятора та байту ОЗП, що непрямо-адресується, з переносом
ADDC A,#data	Операція підсумовування вмісту акумулятора та константи з переносом
SUBB A,R_n	Операція віднімання з вмісту акумулятора вмісту регістра з вивантаженням
SUBB A,direct	Операція віднімання з вмісту акумулятора байта, що прямо-адресується, з вивантаженням
SUBB A,@Ri	Операція віднімання з вмісту акумулятора байта ОЗП з вивантаженням, що непрямо-адресується
SUBB A,#data	Операція віднімання з вмісту акумулятора константи з вивантаженням
INC A	Інкремент акумулятора А
INC R_n	Інкремент регістра
INC direct	Інкремент байта, що прямо-адресується
INC @Ri	Інкремент байта ОЗП, що побічно-адресується
DEC A	Декремент акумулятора А
DEC R_n	Декремент регістра
DEC direct	Декремент байта, що прямо-адресується
DEC @Ri	Декремент байта ОЗП, що непрямо-адресується
INC DPTR	Інкремент покажчика даних
MUL AB	Операція множення акумулятора А на регістр В
DIV AB	Операція ділення акумулятора А на регістр В
DA A	Десяткова корекція акумулятора А

Логічні операції та операції зсуву

Таблиця 3 – Основні логічні операції та операції зсуву над даними

Мнемоніка команди	Тлумачення
ANL A,Rn	Логічне І даних акумулятора і регістра
ANL A,direct	Логічне І даних акумулятора і байта, що прямо-адресується
ANL A,@Ri	Логічне І даних акумулятора і байта ОЗУ, що побічно-адресується
ANL A,#data	Логічне І даних акумулятора і константи
ANL direct,A	Логічне І даних байту, що прямо-адресується, і акумулятора
ANL direct,#data	Логічне І даних байту, що прямо-адресується, і константи
ORL A,Rn	Логічне АБО вмісту акумулятора і регістра
ORL A,direct	Логічне АБО вмісту акумулятора і байта, що прямо-адресується
ORL A,@Ri	Логічне АБО вмісту акумулятора і байта ОЗУ, що побічно-адресується
ORL A,#data	Логічне АБО даних акумулятора і константи
ORL direct,A	Логічне АБО вмісту байта, що прямо-адресується, і акумулятора
ORL direct,#data	Логічне АБО вмісту байта, що прямо-адресується, і константи
XRL A,Rn	Виключаюче АБО вмісту акумулятора і регістра
XRL A,direct	Виключаюче АБО вмісту акумулятора і байта, що прямо-адресується

Продовження таблиці 3 – Основні логічні операції та операції зсуву над даними

Мнемоніка команди	Тлумачення
XRL A,@Ri	Виключаюче АБО вмісту акумулятора і байта ОЗУ, що побічно-адресується
XRL A,#data	Виключаюче АБО даних акумулятора і константи
XRL direct,A	Виключаюче АБО вмісту байта, що прямо-адресується, і акумулятора
XRL direct,#data	Виключаюче АБО вмісту байта, що прямо-адресується, і константи
CLR A	Операція обнуління вмісту акумулятора
CPL A	Інверсія акумулятора
RL A	Зсув акумулятора ліворуч, циклічне
RLC A	Зсув акумулятора ліворуч через перенесення
RR A	Зсув акумулятора праворуч, циклічне
RRC A	Зсув акумулятора праворуч через перенесення
SWAP A	Обмін місцями тетрад в акумуляторі

Робоче завдання

1. Написати програму на мові Асемблера за варіантом завдання, що вказується викладачем.
2. Продемонструвати програму викладачу та результати виконання програми, відповідно.
3. Занести до протоколу усі повідомлення, що виводяться програмою-монітором у діалогове вікно при виконанні програми, текст і лістінг програми.

Варіанти завдань для лабораторної роботи

1. Заповнити адреси $0000...,$0015 значеннями #16. Обнулити вміст акумулятора А та регістра R1...R3.

2. Завантажити вміст регістра ознак у осередок $0020.

3. Здійснити обмін даними між стеком та регістром В. А також здійснити передачу вмісту акумулятора А до регістра ознак.

4. Занесіть в акумулятор А дані, що містяться у комірці з адресою $0030 і після цього скопіюйте його у покажчик стека. Завантажте в акумулятор А вміст осередку пам'яті, на яку вказує покажчик стека, а в акумулятор В – вміст комірки пам'яті, адреса якої на одиницю більше. Поміняйте місцями молодшу та старшу тетради акумулятора В.

5. Використовуючи режим регістрової адресації завантажте в регістр R5 вміст осередка пам'яті з адресою $0033.

6. Повторити п.5 тільки занесіть дані з осередку, адреса якого на $13 менша ніж $0034. Вміст осередку скопіюйте також до регістра R5.

7. Занесіть вміст комірки ПД $0031, $0032 до регістра R0. А після цього вміст цього регістра скопіюйте до стеку.

8. Занесіть в акумулятор А вміст комірки пам'яті з номером $0022. Поміняйте місцями молодшу та старшу тетради акумулятора А.

9. Скопіювати вміст регістра ознак у комірку пам'яті з адресою $0014. Занесіть у регістр ознак вміст осередку, на який вказує покажчик стеку.

10. Занесіть в акумулятор В вміст комірки пам'яті з номером $003E. Поміняйте місцями молодшу та старшу тетради акумулятора В.

Контрольні запитання

1. Які групи команд складають систему команд мікроконтролера C8051F120?

2. Які режими адресації підтримує мікроконтролер C8051F120?

3. Яким чином організовано режим відносної адресації в МК C8051F120 по відношенню до операндів, регістрів та констант ?

4. Яким чином визначається режим абсолютної адресації в системі команд МК C8051F120 по відношенню до операндів, регістрів та констант ?

5. Як реалізовано режим широкої адресації в системі команд МК ?

6. Які особливості має режим індексної адресації в системі команд МК ?

7. Наведіть приклади команд, де використовується безпосередня адресація.

8. Яким чином "знаходиться" адреса операнда у випадку використання регістрова адресація? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

9. Яким чином "знаходиться" адреса операнда у випадку використання безпосередньо-регістрової адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

10. Яким чином "знаходиться" адреса операнда у випадку використання прямої бітової адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

11. Яким чином "знаходиться" адреса операнда у випадку використання прямої байтової адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

12. В яких випадках в програмах використовується режим неявної адресації? Наведіть особливості застосування цього режиму адресації.

13. Які види операндів та які операції з ними можна використовувати у випадку застосування команд передачі даних? Наведіть приклади кожного з видів.

14. Що відбувається з програмним лічильником програми при застосуванні режиму відносної адресації ? Наведіть приклади команд.

ЛАБОРАТОРНА РОБОТА №4

Ознайомлення з особливостями застосування на мові Асемблера системи команд керування програмою та процесором, вивчення команд умовного розгалуження.

Мета роботи - Вивчити основні команди керування програмою та процесором, отримати навички та вміння щодо застосування команд умовного розгалуження.

Домашнє завдання

1) Ознайомитися оглядово із основними командами умовного розгалуження.

2)Вивчити основні команди керування програмою, що реалізуються на мові Асемблера.

3) Визначити, що відбувається з вмістом програмного лічильника у випадку використання команд керування програмою.

4)Вивчити основні команди керування процесором мікроконтролера C8051F120. Визначити, до яких з цих команд має доступ користувач.

Теоретичні відомості

Основні команди керування програмою та процесором

Команди умовного переходу, які використовуються в тілі програми, дають можливість користувачу здійснити перехід до підпрограми, яка реалізована на мові Асемблера та вихід з неї відповідно. Крім цього, у

структурі цього блоку команд є такі, що допомагають здійснити виконання програмного переривання та реалізують вихід з цього переривання без втрати корисної інформації. Окрема частина команд призначена для переходу процесору мікроконтролера у різні режими функціонування.Як було зазначено вище, в командах розгалуження використовується пряма та регістрова адресація. Якщо виконується умова розгалуження, то вміст восьмирозрядного знакового байта, який слідує за кодом команди (зсув), завантажується до вмісту програмного лічильника для формування адреси, на яку і буде здійснений перехід; у протилежному випадку виконання триває з команди, що слідує за командою розгалуження. Код команди розгалуження складається як правило з двох або трьох байт.

Таблиця 4 – Основні команди умовного розгалуження та керування програмою та процесором МК

Мнемоніка команди	Тлумачення
JC rel	Перехід, якщо прапорець перенесення встановлено
JNC rel	Перехід, якщо прапорець перенесення не встановлено
JB bit,rel	Перехід, якщо біт рівний одиниці
JNB bit,rel	Перехід, якщо біт рівний нулю
JBC bit,rel	Перехід, якщо біт встановлений, і відбувається подальше скидання біту
ACALL addrl	Абсолютний виклик підпрограми в межах сторінки в 2 кбайти
LCALL addrl6	Довгий виклик підпрограми
RET	Повернення з підпрограми
RETI	Повернення з підпрограми обробки переривання

Продовження таблиці 4 – Основні команди умовного розгалуження та керування програмою та процесором МК

Мнемоніка команди	Тлумачення
AJMP addrl	Абсолютний перехід усередині сторінки в 2 кбайти
LJMP addrl6	Довгий перехід в повному об'ємі пам'яті програм
SJMP rel	Короткий відносний перехід усередині сторінки в 256 байт
JMP @A+DPTR	Непрямий відносний перехід
JZ rel	Перехід, якщо акумулятор рівний нулю
JNZ rel	Перехід, якщо акумулятор не рівний нулю
CJNE A,direct,rel	Порівняння акумулятора з байтом, що прямо-адресується, і перехід, якщо не рівно
CJNE A,#data,rel	Порівняння акумулятора з константою і перехід, якщо не рівно
CJNE Rn,#data,rel	Порівняння регістра з константою і перехід, якщо не рівно
CJNE @Ri,#data,rel	Порівняння байта ОЗУ, що побічно-адресується, з константою і перехід, якщо не рівно
DJNZ R_n,rel	Декремент регістра і перехід, якщо не нуль
DJNZ direct,rel	Декремент байта, що прямо-адресується, і перехід, якщо не нуль
NOP	Пуста команда

Робоче завдання

1. Написати програму на мові Асемблера за варіантом завдання, що вказується викладачем.

2. Продемонструвати програму викладачу (лістінг) та результати виконання програми, відповідно.

3. Занести до протоколу усі повідомлення, що виводяться програмою-монітором у діалогове вікно при виконанні програми, текст і лістинг програми.

Варіанти для виконання лабораторної роботи

1. Напишіть програму сортування комірок пам'яті $0090-$00BF за зростанням.

2. Напишіть на мові асемблера Basic – програму та реалізувати її засобами команд МК

```
10 LETA=5
20 FOR B=1 TO 9
30 IF B<4 THEN GO SUB 60
40 NEXT B
50 GO TO 80
60 LET A=A+B
70 RETURN
80 STOP
```

3. Напишіть програму підрахунку кількості комірок пам'яті зі значеннями, що відрізняються від #28.

4. Напишіть програму підрахунку кількості від'ємних та додатних чисел, вважаючи при цьому що нуль є нейтральним числом.

5. Нехай у комірках пам'яті $0050, $0051 містяться відповідно змінні A, B. Якщо вони упорядковані за зростанням, то виконати заповнення комірок

пам'яті $0090-$00B0 значеннями #15, у противному випадку заповнити значеннями #36.

6. Нехай в комірці $0080 знаходиться деяке число. Перевірити, чи можна це число без остачі поділити на #03.

7. Нехай в комірці $0080 знаходиться деяке число. Перевірити, чи можна це число без остачі поділити на #0A.

8. Сформувати програму сортування значень комірок пам'яті визначеної області за зменшенням.

Контрольні запитання

1. Які ви знаєте команди умовного розгалуження, що використовуються в програмному режимі мікроконтролера C8051F120 ?

2. Наведіть з поясненням основні команди керування процесором МК серії C8051F120.

3. Які види адресацій використовуються в командах розгалуження ?

4. Яким чином визначається адреса програмного лічильника програми при використання команд розгалуження ?

5. Які основні функції покладаються на команди умовного переходу при написанні програм з циклами ?

ЛАБОРАТОРНА РОБОТА №5

Ознайомлення з основними можливостями мікроконтролерів серії 8051 на прикладі використання MCU C8051A120 Silicon Laboratories.

Мета роботи – Розглянути основні можливості мікроконтролера на базі системного контролера CIP-51 з використанням мнемокодів (інструкцій) мікроконтролера серії C8051F12x; основні способи організації переривань і можливість їх використання; можливості регістрів загального призначення (GPR) і особливості роботи з ними; організація пам'яті системного контролера CIP-51 (CIP-51 System Controller); режими керування енергоспоживанням.

Теоретичні відомості

До складу мікроконтролера серії C8051F12x можуть входити до 24 периферійних пристроїв, включаючи і зовнішні пристрої, проте в межах цього прикладу будуть коротко розглянуті лише деякі з них.

Організація пам'яті системного контролера CIP-51 (CIP-51 System Controller) схожа до організації пам'яті у стандарті 8051. Уся пам'ять мікроконтролера розподіляється на пам'ять програм і пам'ять даних.

Пам'ять програм і пам'ять даних використовують один і той самий адресний простір, проте доступ до них здійснюється через різні типи мнемокодів. Для MCU C8051F12x виділено 256 Байт внутрішньої пам'яті даних і 128 кБайт внутрішньої пам'яті програм. Організація пам'яті системного контролера CIP-51 (CIP-51 System Controller) зображена на рисунку 3.

Пам'ять даних

В системному контролері CIP-51 доступно 256 Байт внутрішньої RAM пам'яті, що займає адресний простір 0x00-0xFF. Нижні 128 Байт пам'яті даних використовуються для регістрів загального призначення (РЗП) і власне пам'яті. Для доступу до нижніх 128 Байт пам'яті даних може застосовуватися як пряма так і непряма адресація. Адресний простір 0x00-0x1F використовується для доступу до 4 банків (сховищ) РЗП, причому кожний банк містить 8 байтових регістрів. Наступні 16 байт (0x20-0x2F) можуть бути використані як для байтової адресації так і для бітової адресації (усього 128 біт) через пряму адресацію.

Верхні 128 Байт пам'яті даних доступні лише через непряму адресацію. Ця ділянка займає такий самий адресний простір, що й регістри спеціального призначення (РСП), але фізично вони розділені між собою.

Режим адресації для доступу до адреси вище 0x7F є визначальним показником, чи то є звернення до пам'яті даних, чи до РСП: якщо використовується непряма адресація – звернення до верхніх 128 Байт пам'яті даних; якщо використовується пряма адресація – звернення до РСП.

Регістри загального призначення

Нижні 32 Байти пам'яті даних (0x00-0x1F) можуть бути використані як 4 банки (сховища) РЗП, що містять 8 байтових регістрів R0-R7. Лише один з 4-ох банків може бути активний одночасно. Два біти у PSW: RS0 (PSW.3) і RS1 (PSW.4) керують вибором активного банку. Наявність банків дозволяє прискорити перехід за вектором переривання до ISR

(Interrupt Service Routine). РЗП R0 і R1 використовуються як індексні регістри в режимі непрямої адресації.

Бітова адресація та адресний простір

В додаток до прямої адресації (в пам'яті даних), що визначається як байтова адресація, адресний простір 0x20-0x2F доступний для бітової адресації (усього 128 Біт). Кожний біт має адресу від 0x00 до 0x7F. Біт 0 байта з адресою 0x20 має адресу 0x00, тоді як біт 7 байта з адресою 0x20 має адресу 0x07. Біт 7 байта з адресою 0x2F має адресу 0x7F. Бітовий доступ відрізняється від байтового типом інструкцій, що використовуються.

Завдання

Обчислити суму чисел A_1, A_2, A_3, …, A_n кратних 3, що знаходяться у пам'яті даних мікроконтролера і результат занести до РЗП. За допомогою периферійного пристрою Multiply and Accumulate (MAC0) обчислити алгебраїчний вираз Result=BC+DR_n. Організувати доступ до портів вводу/виводу інформації і за допомогою РСП (SFR) дозволити передачу і передати необхідну інформацію, яка б сигналізувала про те, що Result<0, на виводи порту P1. Результатом перевірки має бути світіння зеленого світлодіода, якщо результат від'ємний.

A_1, A_2, A_3, …, A_n – довільні числа (нехай n=6);

Result – значення, що заноситься до регістру MAC0;

B, C, D – довільні числа (нехай B=22d, C=-4d, D=2);

R_n – значення суми A_1+ A_2+ A_3+ …+ A_n, що зберігається у довільному РЗП.

Алгоритм і порядок виконання завдання

$include (c8051f120.inc)

Ця команда необхідна для того, щоб присвоїти РСП статусу РЗП. Лише після цього стане можливим використання інструкцій CIP-51 по відношенню до РСП. У файлі c8051f120.inc міститься перелік РСП, що підлягають присвоєнню.

green equ P1.6

За допомогою псевдокоманди equ здійснюється присвоєння мітці green значення активного рівня 6-го піну порту P1.

mov WDTCN, #0deh
mov WDTCN, #0adh

За допомогою цих двох команд здійснюється запис до РСП WDTCN, що керує роботою WDT (Watchdog Timer). Мікроконтролер містить програмований WDT, що дозволяє вимкнути системний лічильник (system clock). Переповнення WDT змусить мікроконтролер перейти в статус RESET. Щоб запобігти цьому WDT необхідно перезапускати до його переповнення. У противному випадку, якщо програмно WDT не буде перезапущено, він змусить мікроконтролер перейти в статус RESET.

WDT виконує функцію захисного таймера у разі некоректного перебігу роботи MCU. WDT містить 21 біт таймера, що керується системним лічильником. Таймер вимірює тривалість між двома

послідовними записами до його контрольного регістру. Якщо ця тривалість перевищить програмно-заданий поріг WDT активує RESET.

Для того щоб цього не сталося слід дезактивувати WDT за допомогою РСП WDTCN записом у нього відповідних значень. Слід зазначити, що запис 0xDE і 0xAD має відбуватися з періодом не більшим за 4 машинні цикли; у противному випадку операція дезактивації ігнорується.

```
mov 40h, #0ch
mov 41h, #09h
mov 42h, #04h
mov 43h, #12h
mov 44h, #0ah
mov 45h, #03h
```

Область пам'яті даних обрана 0x40-0x45.

```
mov R0, #40h
mov R3, #00h
mov R2, #00h
```

Ці команди необхідні для реалізації подальшої непрямої адресації на чарунки пам'яті, в яких містяться лічильні дані. Команда mov R2, #00h обнулює регістр R2.

```
loop1:      mov B, #03h
        mov A, @R0
        div AB
        mov A, B
        inc R0
        inc R3
```



```
        jnz loop1
;**************************************
        mov A, R2
        dec R0
        dec R3
        add A, @R0
        mov R2, A      ;the result is here
        inc R0
        inc R3
        cjne R3, #06h, loop1
;**************************************
```

Вище наведений фрагмент програми дозволяє обчислити значення суми усіх чисел, що кратні 3 (трьом). Сума усіх чисел не повинна перевищувати одного байту.

```
loop4:      MOV SFRPAGE, #03h
```

Пам'ять даних з адресним простором 0x80-0xFF містить РСП. За допомогою РСП проводиться контроль і обмін даними з ресурсами і периферійними пристроями CIP-51. Доступ до РСП здійснюється через пряму адресацію. РСП, чиї адреси закінчуються на 0x0 або 0x8 (наприклад P0, TCON, P1, SCON, IE і тд.) є як біт- так і байт адресованими. Усі інші РСП є лише байт адресованими. Незайняті адреси у адресному просторі РСП зарезервовані для майбутнього використання, і доступ до них не є рекомендованим, оскільки це може спричинити невизначені наслідки.

CIP-51 підтримує SFR Paging, тим самим дозволяючи пристроям звертатися до багатьох РСП в одному й тому ж адресному просторі (0x80-

0xFF). Область пам'яті даних містить 256 SFR Pages. Тому у кожному адресному просторі (0x80-0xFF) може бути отримано доступ до 256 РСП. У C8051F12x зарезервовані такі SFR Pages: 0, 1, 2, 3, F. SFR Pages обираються за допомогою РСП SFRPAGE.

Далі програма прикладу напрямлена на роботу з MAC0. За MAC0 закріплена 3-тя SFR Page. Тому вищезазначена команда має такий вигляд.

```
    MOV MAC0CF, #00h
    MOV MAC0AH, #00h
    MOV MAC0AL, #16h
    MOV MAC0BH, #0ffh
    MOV MAC0BL, #0fch
    MOV MAC0AH, #00h
    MOV MAC0AL, #02h
    MOV MAC0BH, #00h
    MOV MAC0BL, R2
;*************************
    NOP
    NOP
    NOP
;*************************
```

Периферійний пристрій MAC0

До складу мікроконтролера C8051F120 входить пристрій MAC0 (Multiply and Accumulate), за допомогою якого можуть бути значно пришвидшені багато математичних операцій. MAC0 містить 16x16 бітний перемножувач і 40 бітний суматор, за допомогою яких можливо реалізувати ціло- та дробочисленні операції (множення і додавання або

окремо множення) над знаковими даними протягом двох SYSCLK циклів. Пристрій округлення забезпечує 16 бітну дробовочисленну операцію після додаткового (третього) SYSCLK цикла. MAC0 також містить 1-но бітний арифметичний пристрій, що дозволяє зсувати право-ліворуч вміст 40 бітного акумулятора/суматора протягом одного SYSCLK циклу. Рисунок 8 ілюструє структурну схему MAC0 і його РСП.

Для коректної роботи MAC0 передбачено 13 РСП. Два з них пов'язані з режимами роботи і операціями MAC0, одинадцять інших використовуються для зберігання багатобайтових вхідних і вихідних даних для MAC0. Регістр конфігурації MAC0CF використовується для конфігурації і контролю MAC0. Статусний регістр MAC0STA містить прапорці (признаки), що сигналізують про переповнення або наявність нульових/від'ємних результатів. 16 бітні регістри MAC0 (MAC0AH: MAC0AL) і MAC0B (MAC0BH: MAC0BL) використовуються як входи даних для перемножувача. 40 бітний акумулятор/суматор складається з 5 РСП: MAC0OVR, MAC0ACC3, MAC0ACC2, MAC0ACC1 і MAC0ACC0. Початкові результати роботи MAC0 зберігаються у акумуляторі/суматорі. Якщо це необхідно, округлені дані зберігаються у 16 бітному MAC0RND (MAC0RNDH: MAC0RNDL). Рисунки 9 і 10 ілюструють, як ціло- та дробовочисленні значення подаються і зберігаються у РСП.

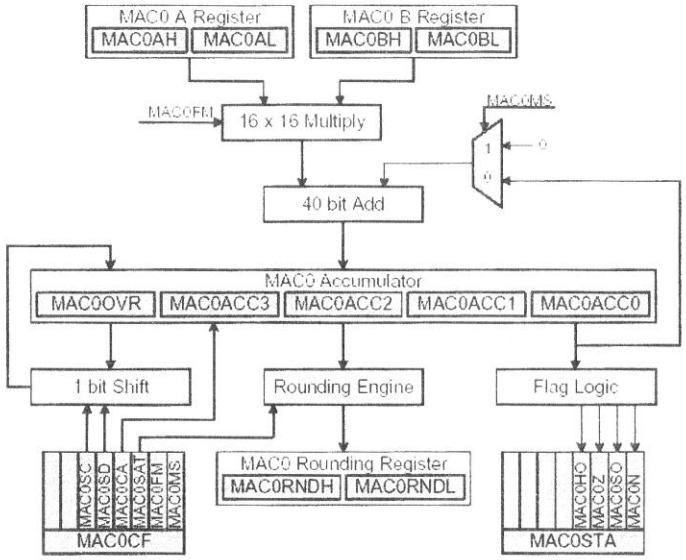

Рисунок 8 – Структурна схема MAC0

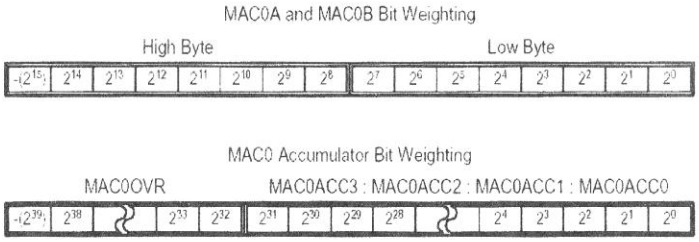

Рисунок 9 – Цілочисленне представлення

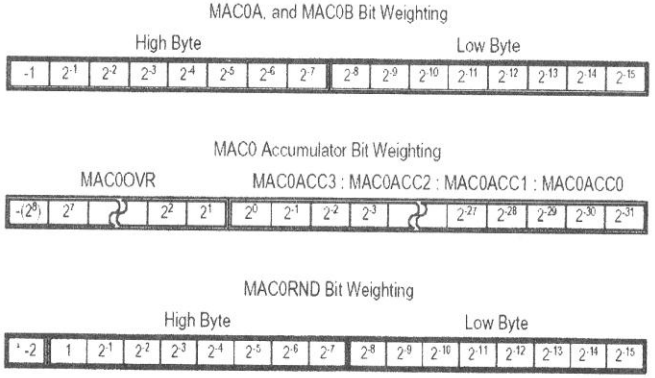

Рисунок 10 – Дробовочисленне представлення

Для нашого прикладу у РСП заноситься значення 00h, тим самим обирається режим роботи MAC0 як MAC (Multiply and Accumulate). У 16 бітні регістри MAC0 (MAC0AH: MAC0AL) і MAC0B (MAC0BH: MAC0BL) вносяться початкові дані: першого разу – для першого доданку, другого разу – для другого доданку. У регістр MAC0BL заносяться дані з РЗП R2, що є результатом підсумовування усіх чисел кратних трьом.

Три мнемокоди NOP у кінці необхідні для створення часових затримок тривалістю один цикл кожна, протягом яких відбувається перемноження, додавання, зберігання і округлення-корекція відповідно. Рисунок 11 ілюструє що необхідність.

```
mov R0, MAC0STA
cjne R0, #00h, cplclick
cpl green
```

Ці команди реалізують занесення до РЗП вмісту регістру MAC0STA. Якщо це значення не нульове, то інвертується біт P1.6, внаслідок чого відбудеться загорання зеленого світлодіоду, що сигналізуватиме про наявність від'ємного результату у акумуляторі MAC0.

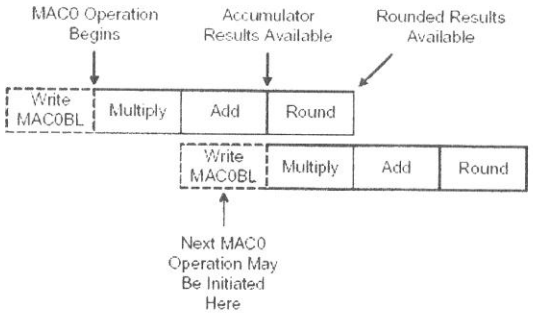

Рисунок 11 – Трубопровідна структура MAC0

```
cplclick:    mov SFRPAGE, #CONFIG_PAGE

             mov XBR2, #40h

             mov P1MDOUT, #40h

;**********************************
```

Наступним кроком є активація CROSSBAR для того, щоб надати порту P1 статусу General-Purpose I/O. Це досягається вибором SFR Page=0Fh=#CONFIG_PAGE і занесенням до РСП XBR2 відповідного

значення, що активує CROSSBAR. Ще одним кроком, аби світлодіод засвітився, є вибір конфігурації підключення виводу P1.6 як Push-Pull, а не Open-Drain (занесення до регістру P1MDOUT відповідного значення).

```
             mov PCON, #01h
             mov PCON, PCON
```

Передостанні дві команди переводять мікроконтролер в режим IDLE. Двобайтна команда <mov PCON, PCON> необхідна для того, щоб після переривання, мікроконтролер вийшов з режиму IDLE. У випадку відсутності двобайтної команди або наявності однобайтної після переривання, мікроконтролер може не вийти з режиму IDLE.

```
end
```

Необов'язкова команда, що вказує на завершення програми. Усі команди, що знаходяться після цієї команди ігноруються.

Лістінг програми

```
$include (c8051f120.inc)

             green equ P1.6

             mov WDTCN, #0deh
             mov WDTCN, #0adh

             mov 40h, #0ch
```

```
        mov 41h, #09h
        mov 42h, #04h
        mov 43h, #12h
        mov 44h, #0ah
        mov 45h, #03h   ;necessary must be divided in 3

;********************
        mov R0,      #40h
        mov R3, #00h
        mov R2, #00h

loop1:      mov B,      #03h
        mov A, @R0
        div AB
        mov A, B
        inc R0
        inc R3
        jnz loop1
;***************************************
        mov A, R2
        dec R0
        dec R3
        add A, @R0
        mov R2, A     ;the result is here

        inc R0
        inc R3
        cjne R3, #06h, loop1
```



```
;***********************************

loop4:      MOV SFRPAGE, #03h
        MOV MAC0CF, #00h

        MOV MAC0AH, #00h
        MOV MAC0AL, #16h
        MOV MAC0BH, #0ffh
        MOV MAC0BL, #0fch

        MOV MAC0AH, #00h
        MOV MAC0AL, #02h
        MOV MAC0BH, #00h
        MOV MAC0BL, R2
;*************************
        NOP
        NOP
        NOP
;*************************
        mov R0, MAC0STA
        cjne R0, #00h,cplclick
        cpl green

cplclick:   mov SFRPAGE, #CONFIG_PAGE

        mov XBR2, #40h

        mov P1MDOUT, #40h
```

```
;*********************************
        mov PCON, #01h
        mov PCON, PCON

end
```

Цей приклад використання мікроконтролера C8051F120 не є особливо показовим, проте він демонструє особливі можливості мікроконтролера, які можуть бути використані у подальшому для реалізації більш складних завдань.

СПИСОК ЛІТЕРАТУРИ

1. Алгоритмы и процессоры цифровой обработки сигналов/ А.И. Солонина, Д.А.Улахович, Л.А.Яковлев. – СПб.: БХВ-Петербург, 2001. – 464 с.
2. Марков С. Цифровые сигнальные процессоры. – М.: МИКРОАРТ, 1996. – 144 с.
3. Калабеков Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов. – М.: Радио и связь, 1988. – 368 с.
4. Соломенчук В.Г. Аппаратные средства персональных компьютеров. – СПб.:БХВ-Петербург, 2003. – 512 с.
5. Одноплатные микроконтроллеры. Проектирование и применение/ В.А.Швец, В.В.Шестакова, Н.В. Бурцева, Т.В.Мелешко. К.:МК-Пресс, 2005. – 304 с.
6. Корнеев В.В., Киселев А.В. Современные микропроцессоры. – М.: Нолидж, 2000. – 320 с.
7. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola. – М.: Радио и связь, 1998. – 560 с.
8. Проектирование цифровых устройств на однокристальных микроконтроллерах/ В.В.Стапин, А.В.Урусов, О.Ф.Мологонцева. – М.: Энергоатомиздат, 1990. – 224 с.
9. Цифровые устройства и микропроцессорные системы. Задачи и упражнения/ Л.М.Гольденберг, В.А. Малеев, Г.Б.Малько. – М.: Радио и связь, 1992. – 256 с.
10. Архитектура однокристальных ЭВМ. Организация, программирование, практика/ В.И.Жабин и др. – М.: ВЕК, 1997. – 128 с.
11. Абель П. Язык Ассемблера для IBM PC и программирования. – М.: Высш.шк., 1992. – 447 с.

12. Сергиенко А.Б. Цифровая обработка сигналов. – СПб: Питер, 2003. – 604 с.

13.Схемотехніка електронних систем: У 3 кн. Кн..3. Мікропроцесори та мікро- контролери/ В.І.Бойко, Ф.М.Гуржій та ін. – К.: Вища шк., 2004. – 399 с.

14. Николайчук О.И. x51-совместимые микроконтроллеры фирмы Silicon Laboratories (CYGNAL). – М.:ООО"ИД СКИМЕН", 2004. – 628 с.

Додаткова література:

1. Гаврилов П.Ф., Дедов А.Я. Ремонт цифровых телевизоров. Принципы работы, типичные неисправности. – М.: Радиотон, 1999. – 288 с.

2. Гук Ю., Юров В. Процессоры Pentium III, Athlon и другие. – СПб.: Питер, 2000. – 480 с.

3. Трапезон К.О., Швайченко В.Б. Методичні вказівки до лабораторного практикуму з вичення мікроконтролера фірми Motorola типу 68HC11. – К.: НТУУ-КПІ, 2005. – 64 с.

4. Аналоговые и цифровые интегральные микросхемы. Справочное пособие/ С.В.Якубовский и др. - М.: Радио и связь, 1984. - 432с.

5. Завадский В.А. Компьютерная электроника. – К.: ВЕК, 1996. – 368 с.

6. Микроконтроллеры. Выпуск 1. – М.:ДОДЭКА, 1998. – 384 с.

7. WWW.AMD.COM

8. WWW.intel.ru

9. WWW.MOTOROLA.com

10.WWW.TI.COM