Вісник Нац. ун-ту "Львівська політехніка" Інформаційні системи та мережі. – 2004. – № 519. – С.275–284. 11. Плешаков В.А. Киберсоциализация человека в информационном пространстве // Информация и образование: границы коммуникаций INFO'2009: Сборник научных трудов. – Горно-Алтайск: РИО ГАГУ, 2009. 12. Федушко С. Аналіз архітектури та сучасних тенденцій розвитку віртуальних спільнот / С. Федушко // Вісник Нац. ун-ту "Львівська політехніка": "Інформаційні системи та мережі". – 2011. – № 699. – 362–375. 13. Miller K. Communication Theories: Perspectives,processes, and contexts / K. Miller. – New York: McGraw-Hill, 2005. 14. Rheingold H. The Virtual Community. [Електронний ресурс]. – Доступний з: http://www.well.com/user/hlr/vcbook/index.html. 15. Rheingold H. The Virtual Community: Homesteading onthe Electronic Frontier / H. Rheingold // The MIT Press. – 2000. – P.360. 16. Wire N. Led by facebook, twitter, global time spent on social media sites up 82 % year over year, 2010. [Електронний ресурс] – Режим доступу – http://blog.nielsen.com/nielsenwire/global/led-byfacebook-twitter-global-time-spent-on-social-media-sites-up-82-year-over-year/. 17. Zhang Y. Web Communities: Analysis and Construction / Y. Zhang, J. Xu Yu, J. Hou. – 2005. – 187 p.

УДК 004.421.4

В.М. Заяць, В.Р. Колосов
Національний університет "Львівська політехніка",
кафедра загальної екології та екоінформаційних систем

АЛГОРИТМИ І ПРОГРАМНІ ЗАСОБИ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ НАД ЧИСЛАМИ У ФОРМАТІ ЧАСУ ДЛЯ МІКРОКОНТРОЛЕРА СІМЕЙСТВА AVR

Запропоновано способи виконання арифметичних операцій додавання та віднімання чисел, поданих у форматі часу. Розроблено алгоритми та програмні засоби алгоритмічною мовою С для мікроконтролера сімейства AVR.

Ключові слова: арифметичні операції, формат часу, режим реального часу, алгоритм, програма.

The method of implementation of operations of addition and subtraction of numbers, presented in the time-format is considered. The algorithm and the program are developed by the C-language for microcontrollers of AVR family.

Key words: arithmetic operations, time format, real-time algorithm, program.

Вступ

Цифрові пристрої зазвичай виконують операції із перетворення даних у двійковій системі числення. Але часто виникає необхідність обробляти дані в форматі часу (наприклад, зчитаних з годинника реального часу). Найпоширенішими операціями є операції додавання і віднімання чисел.

Автори подібних операції або пропонують використання існуючих додатків (наприклад, Microsoft Excel [1]), або застосування змінних типу string [2], або використання як проміжного BCD-формату [3].

Цілі роботи

Метою роботи є розроблення алгоритмів та реалізація програмних засобів для виконання арифметичних операцій додавання і віднімання чисел, поданих у форматі часу (gg:hh:ss) та підтвердження доцільності його застосування до мікроконтролера сімейства AVR [4–7].

Огляд літературних джерел

Мікроконтролери AVR є розробкою і продуктом фірми Atmel. Це сімейство універсальних 8-бітних мікроконтролерів RISC-архітектури (програма і дані знаходяться в різних адресних просторах) з різними вбудованими периферійними пристроями (АЦП, приймач-передавач, модуль TWI, лічильники, SPI і низка інших пристроїв залежно від типу моделі). Виробляються за технологією 0,35 мкм, працюють з тактовою частотою від 16 МГц. Фірма ATMEL випускає сімейства 8-бітних мікроконтролерів двох типів: tiny і mega. Мікроконтролери tiny мають Флеш-ПЗУ по 1 і 2 кбайти в корпусі на 8–20 виводів, а мікроконтролери mega відповідно: Флеш-ПЗУ 8–128 кбайтів в корпусі на 28–64 виводи, можуть працювати за напруг живлення 2–6 вольт. Можливе переведення їх програмним шляхом в режими економного споживання енергії.

У кінці 1996 року було випущено перший дослідний мікроконтролер AT90S1200, а в другій половині 1997-го корпорація Atmel розпочала серійне виробництво нового сімейства мікроконтролерів, здійснюючи при цьому їх рекламну та технічну підтримку.

Основні версії контролерів такі:

- AT (mega / tiny) xxx – базова версія.
- ATxxxL – версії контролерів, що працюють за зниженої (Low) напруги живлення (2,7 В).
- ATxxxV – версії контролерів, що працюють на низькій напрузі живлення (1,8 В).
- ATxxxP – малоспоживні версії (до 100 нА в режимі Power-down), застосована технологія picoPower (анонсовані в липні 2007) [4], за кількістю виводів і функційними можливостями сумісні з попередніми версіями.

У [5] викладено принципи функціонування, особливості архітектури і прийоми програмування мікроконтролерів Atmel AVR. Наведено готові рецепти для програмування основних функцій сучасної мікроелектронної апаратури: від реакції на натискання кнопки або побудови динамічної індикації до складних протоколів запису даних до зовнішньої пам'яті або особливостей підключення годинника реального часу. Особливу увагу звернено на обмін даними мікроелектронних пристроїв з персональним комп'ютером, наведено приклади програм. У [5] враховано особливості сучасних моделей AVR і супутніх мікросхем останніх років випуску.

У [6] показано всі етапи розроблення пристроїв на мікроконтролерах. Особливу увагу звертають на зв'язок пропонованих схемних рішень з програмним забезпеченням розроблюваних пристроїв. У кожній главі пропонуються електричні схеми пристроїв-контролерів на базі мікроконтролерів AVR, а також декілька програм, що визначають функціонування цих контролерів, а основні прикладні застосування мікроконтролерів цього класу.

Поряд з базовими мікроконтролерами AVR існують і успішно розвиваються інші технології побудови цих пристроїв. Так, у роботі [7] дано вичерпний опис базової серії мікроконтролерів сімейства AVR від компанії Atmel, побудованих на базі прогресивної архітектури RISC із застосуванням програмованої флеш-пам'яті EPROM. Крім того, докладно розглядається програмування мікроконтролерів цієї серії мовою асемблер, а також середовище відлагодження AVR-Studio і програмно-апаратний набір STK200.

Основна частина

Виконання операцій над числами, представленими у форматі часу, ускладнюється тим, що вони не відповідають вимогам до класичних систем числення.

Так, для представлення величин часу використовуються цифри 0 … 9, як і в десятковій системі числення, водночас існують обмеження на максимальне число секунд, хвилин і годин. Так, секунди і хвилини змінюються в межах від 0 до 59, а години – від 0 до 23 (у 24-годинному форматі часу). Наприклад, при додаванні 40 і 30 секунд без урахування цих обмежень результат буде 70 секунд, а з урахуванням – 1 хвилина 10 секунд. Один з варіантів вирішення проблеми полягає в десятковій корекції результату виконання операції і почерговій корекції значення секунд, хвилин і годин. Другий – у конвертуванні числа з формату “години–хвилини–секунди” у формат “секунди”, виконанні операції і зворотному перетворенні.

Опишемо кожен з можливих підходів у алгоритмічному поданні.

Перший алгоритм. Для того, щоб не виконувати десяткової корекції як після додавання, так і після віднімання, що значно ускладнює алгоритм (практично вдвічі), конвертуємо один з операндів у доповняльний код, що дозволить замінити операцію віднімання операцією додавання (рис. 1).

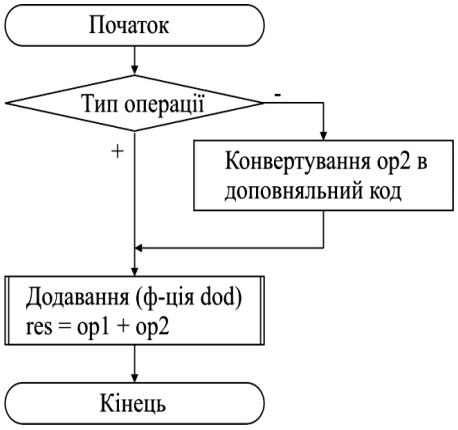

Рис. 1. Схема алгоритму додавання/віднімання чисел, представлених у форматі часу

Процедуру додавання двох чисел з подальшою десятковою корекцією результату наведено на рис. 2.

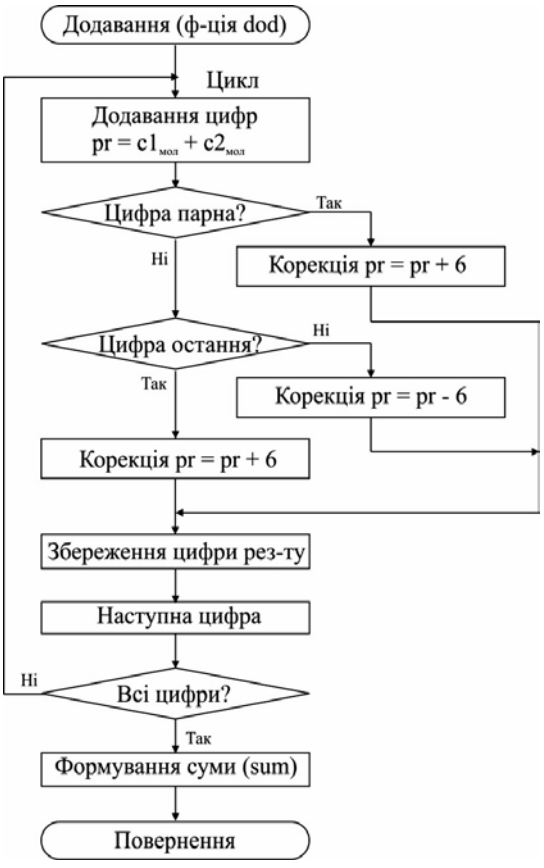

Рис. 2. Схема алгоритму додавання двох чисел з подальшою десятковою корекцією

Від'ємне число представлене в доповняльному коді. Десяткова корекція результату додавання виконується з урахуванням того факту, що цифри одиниць секунд, хвилин і годин (парні цифри числа) змінюються від “0” до “9”, а цифри десятків секунд і хвилин – від “0” до “5”. Кількість десятків годин у цьому алгоритмі не обмежена.

Другий алгоритм. Для конвертування числа в секунди можна використати поширений метод вагових коефіцієнтів, суть якого полягає в додаванні добутків кожної з цифр числа та її ваги. Вагу цифр у секундах для числа, поданого у форматі часу, наведено в таблиці.

Вага	36000	3600	600	60	10	1
Цифра	Десятки годин	Години	Десятки хвилин	Хвилини	Десятки секунд	Секунди

Тоді

$$sek = dg*36000 + g*3600 + dh*600 + h*60 + ds*10 + s, \quad (1)$$

де sek – результат у секундах; dg– кількість десятків годин; g – кількість годин; dh – кількість десятків хвилин; h – кількість хвилин; ds – кількість десятків секунд; s – кількість секунд.

Для зворотного перетворення зручно використати метод послідовного обчислення цифр: десятків годин, годин, десятків хвилин, хвилин, десятків секунд, секунд. Обчислення виконується за таким алгоритмом:

- число в секундах ділиться на вагу цифри десятків годин. Частка дає цифру десятків годин;
- залишок ділиться на вагу цифри годин. Частка дає цифру годин;
- залишок ділиться на вагу цифри десятків хвилин. Частка дає цифру десятків хвилин;
- залишок ділиться на вагу цифри хвилин. Частка дає цифру хвилин;
- залишок ділиться на вагу цифри десятків секунд. Частка дає цифру десятків секунд, залишок дає цифру секунд.

Схему алгоритму наведено на рис. 3.

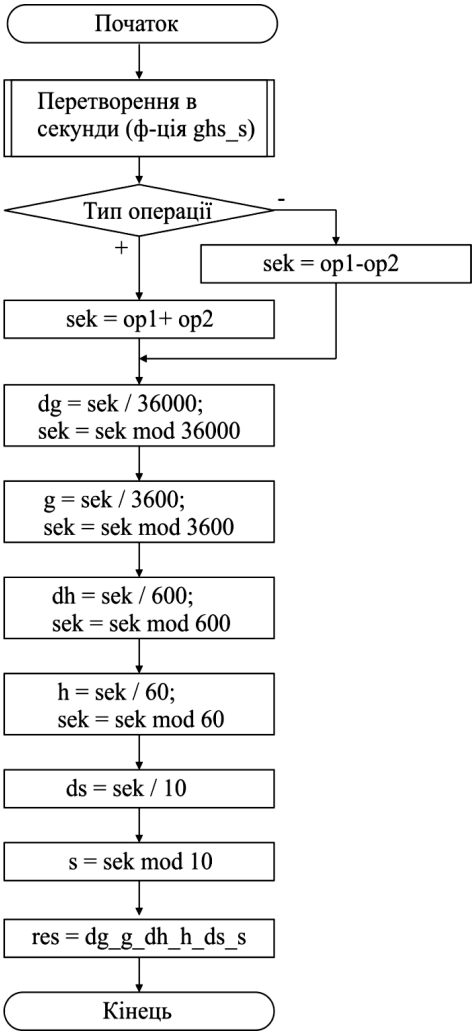

Рис. 3. Схема алгоритму додавання/віднімання чисел, представлених у форматі часу (подання через секунди)

Схему алгоритму перетворення чисел з формату години-хвилини-секунди в секунди наведено на рис. 4.

Рис. 4. Схема алгоритму перетворення чисел з формату години-хвилини-секунди на секунди

Обидві програми зручно представити у вигляді параметризованої процедури, в яку під час виклику передаються тип операції (додавання або віднімання) і значення операндів.

Значення, що повертається, і буде шуканим результатом.

Роздрук програми, реалізованої за алгоритмом 1, показано на Listing 1.

Listing 1. Програма виконання операцій додавання/віднімання над числами, представленими в форматі часу.

```
#include <avr/io.h>

#define u8          unsigned char
#define u32         unsigned long

volatile u32 a,b,c,d;                  //числа у форматі часу (години-хвилини-секунди)
//------------ п/пр переводу годин, хвилин і секунд у секунди ----------------------
u32 ghs_s(u32 ghs)
{
 volatile u32 sek; //секунди

sek=(((ghs>>20)&0x0F)*36000)+(((ghs>>16)&0x0F)*3600)+(((ghs>>12)&0x0F)*600)+(((ghs>>8)&0x0F)*60)+(
((ghs>>4)&0x0F)*10)+(ghs&0x0F);
 return sek;
}

//------------ п/пр арифм. операцій над числами у форматі часу ---------------------
u32 ar_op(u8 op, u32 op1, u32 op2)
{
 volatile u8 s,ds,h,dh,g,dg;                    //цифри вихідного числа
 volatile u32 sek,res;                                  //результат

 if(op==0)          sek=ghs_s(op1)+ghs_s(op2);   //додавання
 else               sek=ghs_s(op1)-ghs_s(op2);   //віднімання

 dg=sek/36000; sek%=36000;                      //розрахунок десятків годин
 g=sek/3600; sek%=3600;                         //розрахунок годин
 dh=sek/600; sek%=600;                          //розрахунок десятків хвилин
 h=sek/60; sek%=60;                             //розрахунок хвилин
 ds=sek/10;                                             //розрахунок десятків секунд
 s=sek%10;                                              //розрахунок секунд

 res=((u32)dg<<20)|((u32)g<<16)|((u32)dh<<12)|((u32)h<<8)|((u32)ds<<4)|s;
 return res;
}

int main()
{
 a=0x132259;
 b=0x023441;
 while(1)
 {
 c=ar_op(0,a,b);   //додавання
 d=ar_op(1,a,b);   //віднімання
 a++;
 b++;
 }
}
```

Аналіз коду програм показав, що в першому випадку програма займає 67 рядків, а в другому – 38 рядків. Перший алгоритм складний для розуміння, другий – доступніший.

Водночас скомпільована перша програма має об'єм 906 байтів, а друга – 1398 байтів.

Для порівняння розглянуто програму, написану мовою CI++, в якій реалізований тип даних string [2]. Роздрук програми наведено на Listing 2. Результати виконання арифметичних операцій на користь алгоритму 1.

Listing 2.

```
. include <stdio.h>

void set_time(char* _time, char form, char oper, int value)
{
    int *tmp_int = new int;
    sscanf(_time, “%d”, tmp_int);
    int time = 0;
    time = time + (*tmp_int%100) + ((*tmp_int%10000)/100)*60 + (*tmp_int/10000)*3600;
    delete tmp_int;
    switch(oper)
    {
    case '+':
        switch(form)
        {
        case 'h': time = time + value*3600; break;
        case 'm': time = time + value*60; break;
        case 's': time = time + value; break;
        default: return;
        }
        break;
    case '-':
        switch(form)
        {
        case 'h': time = time - value*3600; break;
        case 'm': time = time - value*60; break;
        case 's': time = time - value; break;
        default: return;
        }
        break;
    default: return;
    }
    if(time<=0)
    {
    sprintf(_time, “000000”);
    }
    else
    {
        if((time/3600)>=10)
        {
            sprintf(_time, “%d”, (time/3600));
        }
        else
        {
            sprintf(_time, “0 %d”, (time/3600));
        }
        if(((time%3600)/60)>=10)
        {
            sprintf(_time, “%s%d”, _time,  ((time%3600)/60));
        }
        else
        {
            sprintf(_time, “%s0 %d”, _time, ((time%3600)/60));
        }
        if((time%60)>=10)
        {
            sprintf(_time, “%s%d”, _time, time%60);
        }
        else
        {
            sprintf(_time, “%s0 %d”, _time, time%60);
        }
    }
}

int main(){
```

```
    char *time = new char[7];
    sprintf(time, “071355”);
    set_time(time, 'm', '+', 1);
    set_time(time, 's', '+', 13);
    printf("%s\n”, time);
    return 0;
}
```

Аналіз швидкодії запропонованих та розглянутих алгоритмів показав, що для виконання алгоритму 1 вимагається виконання 7746 операцій додавання, тоді як для алгоритму 2 – 1074 операцій, програма зображена на Listing 2, вимагає ще меншого числа операцій. Це пов’язано з необхідністю опрацювання довгих 32-бітних масивів числових даних при 8-бітному мікропроцесорі AVR. При опрацюванні даних у форматі часу на 32-розрядному мікропроцесорі передбачається часовий виграш при застосуванні алгоритму 1 у 1,5 разу порівняно з алгоритмом 2, і часові затрати будуть співмірними із застосуванням програми Listing 2.

Висновки

Розроблено два варіанти алгоритму і програми виконання операцій додавання та віднімання чисел, представлених у форматі часу. Програма написана мовою С для мікроконтролера сімейства AVR. Проаналізовано та досліджено роботу алгоритму виконання операцій за запропонованим методом. Алгоритм дає змогу швидко виконати операції над числами.

Перша програма дає довгий код, але займає менше місця в пам'яті. Друга – навпаки. При виборі ж варіанта слід враховувати також зрозумілість програми.

1. Курбатова К.А. Microsoft Exel 2003. Стислий курс / К.А. Курбатова. – М.: Вільямс, 2004. – 288 с. 2. http://www.cyberforum.ru/cpp-beginners/thread773189.html, Арифметические операции с данными типа “время”. 3. http://controllersystems.com/ books/praktika_programmirovaniya_atmel_avr/ operacii-s-chislami-v-formate-bcd.html. Операции с числами в формате BCD. 4. Евстифеев А.В. Микроконтроллеры AVR семейства Classic фирмы ATMEL / А.В. Евстифеев. – М.: Издательский дом “Додэка-XXI”, 2006. – 288 с. 5. Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. – СПб.: БХВ Петербург, 2011. – 352с. 6. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов. – М.: Издательский дом “Додэка-XXI”, 2004. – 288 с: 7. Трамперт В. AVR-RISC мікроконтролери / В. Трамперт: Пер. з нім. – К.: МК-Пресс, 2006. – 464 с.