6. *Архипов А. Е.* Применение мотивационно-стоимостных моделей для описания вероятностных соотношений в системе «анализ – защита» / А. Е. Архипов, С. А. Архипова // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – Вип. 1 (16). – К. : 2008. – С. 57–61.
7. *Архипов О. Є.* Оцінювання ефективності системи охорони державної таємниці / О. Є. Архипов, І. Т. Бородавко, В. П. Ворожко. – К. : Наук.-вид. від-діл НА СБ України, 207. – 63 с.

Надійшла 16.02.2009
Після доробки 17.03.2009

Розглянуто деякі особливості застосування методології інформаційних ризиків до дослідження захищеності інформаційних систем, зокрема практичні підходи до оцінювання кількісних показників, що використовуються для обчислення інформаційних ризиків, а також показників захищеності інформаційних систем та ефективності систем захисту інформації.

In this paper some features of information risk methodology application for information systems security research, in particular practical estimation approaches of the quantity indicators used for calculation information risk, and also parameters of information systems security and efficiency of information protection systems are considred.

УДК 004.43

В. М. Крищук, О. Ю. Малий, О. Ю. Воропай

УНІВЕРСАЛЬНА АЛГОРИТМІЧНА МОВА ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ

Досліджено існуючі методи створення алгоритмічних мов програмування. Розроблено універсальну мову програмування мікроконтролерів, що дозволяє переводити програми з одного мікроконтролера в інший, а також задавати загальний алгоритм програми при роботі з любим видом мікроконтролерів.

ВСТУП

Достатня кількість різних сімейств мікроконтролерів, а також відмінності між мікроконтролерами усередині одного сімейства вимагає розробки методів створення універсального засобу, що могло б дозволити робити налагодження пристроїв на основі мікроконтролерів будь-якого типу. Першим етапом на шляху створення такого засобу є розробка універсальної алгоритмічної мови програмування мікроконтролерів, що дозволить врахувати всі особливості кожного з обраних для роботи мікроконтролерів.

Для створення такої мови необхідно вивчити формальну граматику мов програмування, а також визначити метод універсального опису мікроконтролерів, що дозволив би не тільки описувати існуючі на сьогоднішній день мікроконтролери з усіма виконуваними ними функціями, але і давав можливість розширення достатку виконуваних мікроконтролерами функцій, для опису подальших апаратних розробок структур мікроконтролерів. Для цього пропонується врахувати особливості виконання однієї і тієї ж задачі на різних типах мікроконтролерів, включаючи й апаратну реалізацію задачі, що дозволить провести аналіз і знайти загальні риси роботи. Для чого необхідно проаналізувати особливості перетворення програми, написаної для одного типу мікроконтролера, в програму для іншого типу чи підтипу усередині одного сімейства [1].

АНАЛІЗ КОНСТРУКЦІЙ МОВ ПРОГРАМУВАННЯ

Реакція комп'ютера на данні, що вводяться, однозначна – першою справою занести їх до пам'яті, забезпечивши повну схоронність і чекати подальших команд.

Формально мова програмування – це відкрита безліч текстів, написаних за допомогою деякого набору символів – алфавіту мови. За основним своїм призначенням мова програмування – це засіб спілкування між користувачем і комп'ютерною системою.

Синтаксис мов програмування – сукупність вимог, яким повинна задовольняти будь-яка осмислена програма.

Для завдання синтаксичних правил найбільшою популярністю користується апарат форм Бакуса – Наура. Їхнє основне призначення визначити, які саме послідовності символів вважаються програмами в даній мові програмування. Це досягається вказівкою з яких складових частин і яким способом можуть бути побудовані програми [2].

Семантика мови програмування – це правила, що визначають, які операції і у якій послідовності повинна виконати машина, що працює по довільній заданій їй програмі.

Семантика мови програмування в цілому задається вказівкою:

1. Використовуваних у мові типів (тобто множин) простих значень, наприклад цілих і нецілих чисел.

2. Способів побудови складених значень, наприклад комплексних чисел, векторів і чи матриць їхніх аналогів з декількох простих.

3. Операцій над простими, а іноді і складеними значеннями й іншими діями, що відрізняються від операцій, які не виробляють нових числових чи інших значень (наприклад дія, що називається присвоюванням, полягає в тім, що за деяким обчисленим чи обраним значенням закріплюється деяке позначення).

4. Способів завдання складних дій, які складаються з ряду простих, що виконуються у визначеній послідовності.

Прагматика мови програмування – це методологія програмування, тобто опис принципів, методів і прийомів, що дозволяють виходячи з постановки задачі скласти програму її рішення.

У загальному випадку під реалізацією мови програмування на деякій комп'ютерній системі мається на увазі програма, що перекладає текст програм з однієї мови на машинну мову даної комп'ютерної системи [3].

Оператори у відмінності від виразів це конструкції призначені для організації дій більш складних ніж обчислення нових значень. Ці дії зводяться до декількох основних категорій: створити нове ім'я для подальшого використання в програмі (оголошення), обчислити значення деякого виразу і дати йому одне з передбачених у програмі імен, зберігши це значення для використання в майбутньому (присвоєння), установити послідовність, у якій повинні виконуватися інші дії (керування). До операторів керування відносяться: порожній оператор, послідовна композиція, умовна композиція, циклічна композиція, оператори переходу.

ЗАПРОПОНОВАНА РЕАЛІЗАЦІЯ ОПИСУ АПАРАТНИХ І ПРОГРАМНИХ ЗАСОБІВ В УНІВЕРСАЛЬНІЙ АЛГОРИТМІЧНІЙ МОВІ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ

Як універсальну мову програмування мікроконтролерів пропонується застосувати мову, що складається з оператора оголошення, оператора присвоєння («=») і двох операторів керування («if», «goto»), це задовольняє умові апарата форм Бакуса – Наура та включає всі необхідні типи операторів.

Для опису апаратних засобів, що входять до складу мікроконтролерів, варто застосувати об'єктний підхід, при якому всі їхні особливості будуть описані як властивості об'єкта з визначеними параметрами унікальними для кожного окремого мікроконтролера, при тому, що об'єкти будуть створені універсальними, тобто придатними до будь-якого мікроконтролера. У виді об'єктів необхідно описувати як програмні так і апаратні частини мікроконтролера [4]. Синтаксис опису властивостей модулів у виді об'єктів застосуємо стандартний, котрий використовується у всіх макромовах програмування, тобто <об'єкт>. <дочірній об'єкт 1-го порядку> ... <властивість> = = значення властивості.

Властивості об'єкта описуються як змінні чи як масиви, при цьому одна і та сама властивість може бути просто змінної й одномірним масивом, одномірним і двомірним масивом і т. д. у тому випадку, коли описується його частина, наприклад один біт цілого байта. (PORT[1]=45h у той час як PORT[1,0]=1, тут одиниця вказує на загальну властивість тобто номер порту, 0 у другому випадку вказує номер біту (номер виводу), при цьому 45h=01000101b, отже 0-й біт=1), це відповідає аналогічному використанню спеціальних регістрів у мікроконтролерах (MOVWF PORTA, однак BSF PORTA, 2).

Розглянемо на прикладі мікроконтролера PIC16F84A опис деяких програмних і апаратних модулів.

Опишемо деякі з них у вигляді об'єктів із властивостями.

1. Об'єкт стек адресів функцій, які викликають процедури(8 level stack) має наступні властивості:

– розрядність 13 біт (Stack.bits=13);

– кількість рівнів (Stack.levels=8);

– робота над стеком виконується тільки командами виклику процедури і повернення з процедури.

Над ним можна робити наступні операції:

– додати дані (командою виклику процедури), при цьому всі попередні дані зміщуються на один рівень униз, а саме нижнє значення віддаляється;

– прочитати дані (командою повернення з процедури), при цьому об'єкт видає верхнє значення і потім стирає його, а інші значення зміщуються на один рівень нагору.

Приклади застосування:

.....

015A: CALL 02F1h; виклик процедури яка знаходиться за адресою 02F1h

(Stack.level[1]=PC=015Ah=00000001 01011010b;

For i=8 downto 2 do Stack.level[i]=Stack.level[i–1])

.....

02F1: RETURN; повернення з процедури

(PC=Stack.level[1]; for i=1 to 7 do Stack.level[i] = =Stack.level[i+1])

.....

2. Об'єкт оперативна пам'ять (RAM File registers) має наступні властивості:

– кількість осередків (RAM.bytes=68);

– розрядність (RAM.bits=8);

– діапазон адрес (RAM.beginaddr=0Ch, RAM.endaddr=4Fh);

– має 2 банки пам'яті с різницею адресації 80h (RAM.banks=2, RAM.incbanks=80h).

Над ним можна робити наступні операції:

– записати в обраний осередок дані;

Рисунок 1 – Структурна схема процесу перекладу програми з одного мікроконтролера в інший

– вважати з обраного осередку дані.

Приклади застосування:

MOVLW CFh (w=CFh)

MOVWF 12h (RAM.byte[12]=w)

.....

MOVF 4Fh,w (w=RAM.byte[4F])

При цьому будь-яка виконувана команда змінює які-небудь властивості одного чи декількох об'єктів (змінює вміст масиву пам'яті, значення вектора виводів порту, виводів компаратора і т. д.). Усі команди мікроконтролера зводяться до команд присвоєння («=») і команд перевірки («if») з використанням переходів («goto») універсальної алгоритмічної мови програмування.

ВИЗНАЧЕННЯ НЕОБХІДНОЇ КІЛЬКОСТІ ОБ'ЄКТІВ ТА ЇХ ВЛАСТИВОСТЕЙ ШЛЯХОМ ПЕРЕКЛАДУ ПРОГРАМИ З ОДНОГО МІКРОКОНТРОЛЕРА В ІНШИЙ

Для визначення необхідної кількості об'єктів того чи іншого мікроконтролеру та особливостей опису необхідно визначити множини загальних та унікальних для кожного окремого мікроконтроллеру об'єктів. Визначення множини загальних об'єктів пропонується зробити шляхом переводу програм одного мікроконтролера в інший, аналіз цих перетворень дозволить визначити загальні властивості мікроконтролерів та їх відмінності.

Для трансляції програм необхідно створити універсальну (загальну для всіх мікроконтролерів) середню частину перекладу тобто запропоновану універсальну мову, що буде враховувати особливості кожного з мікроконтролерів і приводити їх у деякі загальні структури поруч з приватними властивостями, що описують унікальність (рис. 1). При цьому на МК2 особливості МК1 реалізовувати емуляцією за допомогою наявних структур і їхніх властивостей [4].

ОСОБЛИВОСТІ ОПИСУ ДЕЯКИХ ЗАГАЛЬНИХ ОБ'ЄКТІВ МІКРОКОНТРОЛЕРІВ УНІВЕРСАЛЬНОЮ АЛГОРІТМІЧНОЮ МОВОЮ

Проведений аналіз мов програмування мікроконтролерів дав можливість зробити наступні висновки щодо особливостей опису об'єктів:

1. Робота з виводами мікроконтролера повинна зводиться не до кількості портів визначеної розрядності, а описуватися кількістю необхідних висновків з визначеними властивостями.

2. Робота з будь-яким видом пам'яті мікроконтролера повинна вказувати необхідний обсяг використовуваної пам'яті, а не максимальний обсяг пам'яті мікроконтролера.

3. Настроювання мультіпліцированих пристроїв повинні вироблятися автоматично – якщо у визначений момент часу використовуються один пристрій, всі інші (мультіпліцированні з даним) використовуватися не можуть.

4. Використання спеціальних регістрів і регістрів загального призначення не повинне прив'язуватися до визначених адрес у пам'яті, а необхідно тільки вказувати функції регістра.

ПРИКЛАД ПЕРЕТВОРЕННЯ ПРОГРАМИ ОДНОГО МІКРОКОНТРОЛЕРА В ПРОГРАМУ ІНШОГО

Як приклад, за допомогою якого буде надалі зроблений висновок про методи створення універсальної алгоритмічної мови програмування, приводиться перетворення програми мікроконтролера PIC16F84A у програму мікроконтролера PIC12F629. Даний приклад поки не враховує різницю у вбудованих апаратних пристроях, а працює тільки з пристроями, що входять до складу як одного, так і іншого мікроконтролера з метою показати на даному етапі особливості програмного перетворення без програмної

Рисунок 2 – Карта пам'яті електронного ключа

й апаратної емуляції не вхідних до складу заданих мікроконтролерів пристроїв.

Задана задача наступного змісту:

Розробляється електронний ключ з обмеженою кількістю разів доступу, а також з різними рівнями доступу.

Мікроконтролер у пам'яті EEPROM містить деяку інформацію обсягом 64 байта служить для авторитизації. Серед її 32 байта утримуючих серійний номер ключа (унікальний для кожного), 2 байти рівень доступу і 30 байт лічильник кількості разів доступу (рис. 2). При цьому пристрій, що зчитує, може виконувати тільки 3 команди: читання біта, запис біта (скидання з одиниці в нуль) і встановлення початку області авторитизації. Дані команди виконуються шляхом сполучення двох керуючих сигналів RST і CLK. Читанню біта відповідає високий CLK при низькому RST, запису біта відповідає послідовність високого RST і високого CLK, а встановленню початку області авторитизації – високий CLK при високому RST (рис. 3).

Як видно з особливості задачі лічильник кількості разів доступу побітовий, тобто кількість встановлених біт у лічильнику відповідає дозволеній кількості разів доступу.

Як видно з особливості задачі, лічильник кількості разів доступу побітовий, тобто кількість встановлених біт у лічильнику відповідає дозволеній кількості разів доступу.

Спочатку вирішимо дану задачу на PIC16F84A.

Переходячи до безпосередньої реалізації цієї задачі, необхідно врахувати ту особливість, що порти в заданих для рішення мікроконтролерах здатні видавати в порт інформацію тільки цілим байтом (на кожен вивід по біту). Ця особливість зважується шляхом бітового зрушення вліво чи вправо байта, при цьому в біті carry спеціального регістра status залишається 1 біт, саме він і буде подаватися в порт. Для сприйняття команд пристрою, що зчитує, використовуємо 2 виводи порту A, один із яких візьмемо тактовий (T1CK0 для підрахунку кількості CLK за допомогою регістра TMR0), другий беремо довільно з тих виводів із входом. Також необхідно взяти один вивід для даних які будуть виходити з ключа в пристрій, що зчитує, (вибираємо будь-який двонаправлений). Після цього пишуться процедури обробки подій, що забезпечують правильну роботу ключа по заданих умовах.

Якщо переходити до перекладу цієї програми в мікроконтролер PIC12F629 варто помітити наступні розходження в характеристиках:

– у 12F629 тільки один шостирозрядний порт вводу/виводу, для даної задачі це не істотно, але може значно вплинути на рішення інших задач, у яких необхідна більша кількість виводів;

– 12F629 має в 2 рази більше пам'яті EEPROM, що в даній задачі вносить необхідність контролю за виходом з першої половини адресації EEPROM пам'яті;

– у 12F629 виводи порту мультіпліцированні з внутрішніми пристроями і тому навіть не дивлячись на те, що дані пристрої в задачі використовуватися не будуть, необхідне їх відповідне налаштування;

– 12F629 має відмінну від 16F84A карту пам'яті, у якій спеціальні регістри використовувані в програмі знаходяться по інших адресах, що особливо необхідно врахувати при перекладі програми. Регістри загального призначення також знаходяться в іншому адресному діапазоні, це потрібно враховувати з метою запобігання використання спеціальних регістрів і комірок пам'яті що не можна використовувати.

Далі приведений фрагмент коду програми на мікроконтролерах 16F84A, 12F629 і проміжний код на універсальній алгоритмічній мові.

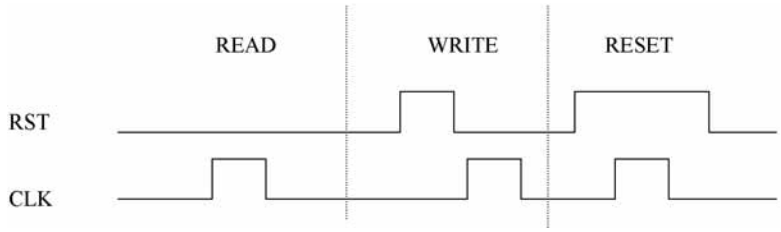

Рисунок 3 – Амплітудно-тимчасова діаграма команд пристрою читання електронного ключа

PIC 16F84A

```
0089: BTFSC PORTB , 00

008A: GOTO 0089
008B: BSF PORTA , 00
008C: MOVWF INTCON
008D: CLRF TMR0
008E: NOP
008F: NOP
0090: GOTO 0094
0091: BTFSC PORTA , 04

0092: GOTO 0091
0093: MOVWF PORTA
0094: BTFSS TMR0 , 00

0095: GOTO 0094
0096: RLF EEDATA , f

0097: MOVF STATUS , W
0098: BTFSC PORTA , 04

0099: GOTO 0098
```

Універсальна алгоритмічна мова

```
if PORT[2,0]=0 then PC=PC+2
else PC=PC;

PORT[1,0]=1;PC=PC+1;
INTCON=w;PC=PC+1;
TMR0=0;PC=PC+1;
PC=PC+1;
PC=PC+1;
PC=PC+4;
if PORT[1,4]=0 then PC=PC+2
else PC=PC;

PORT[1]=w;PC=PC+1;
if TMR0[0]=1 then PC=PC+2
else PC=PC;

EEDATA=EEDATA*2+EEDATA[7];
PC=PC+1;
w=STATUS;PC=PC+1;
if PORT[1,4]=0 then PC=PC+2
else PC=PC;
```

Універсальна алгоритмічна мова

```
if PORT[2,0]=0 then PC=PC+2
else PC=PC;

PORT[1,0]=1;PC=PC+1;
INTCON=w;PC=PC+1;
TMR0=0;PC=PC+1;
PC=PC+1;
PC=PC+1;
PC=PC+4;
if PORT[1,4]=0 then PC=PC+2
else PC=PC;

PORT[1]=w;PC=PC+1;
if TMR0[0]=1 then PC=PC+2
else PC=PC;

EEDATA=EEDATA*2+EEDATA[7];
PC=PC+1;
w=STATUS;PC=PC+1;
if PORT[1,4]=0 then PC=PC+2
else PC=PC;
```

PIC 12F629

```
007F: BTFSC GPIO , 01

0080: GOTO 007F
0081: BSF GPIO , 00
0082: MOVWF INTCON
0083: CLRF TMR0
0084: NOP
0085: NOP
0086: GOTO 008A
0087: BTFSC GPIO , 02

0088: GOTO 0087
0089: MOVWF GPIO
008A: BTFSS TMR0 , 00

008B: GOTO 008A
008C: RLF EEDATA , f

008D: MOVF STATUS , W
008E: BTFSC GPIO , 02

008F: GOTO 008E
```

На прикладі видно, що використовуються різні виводи портів, різні спеціальні регістри опису портів, а також не збігаються адреси фрагментів коду, що не заважає програмі ідентично працювати (робота програми була перевірена практично, шляхом апаратної реалізації).

ВИСНОВКИ

Методика опису програмних і апаратних модулів з допомогою об'єктів дозволить полегшити розуміння їхньої роботи за рахунок наочності взаємодії блоків системи в цілому.

Використання універсальної алгоритмічної мови написаного по приведених методах дозволить розкласти роботу мікроконтролера на елементарні взаємодії об'єктів, що дасть можливість здійснювати емуляцію роботи деяких апаратних модулів програмно і використовуючи наявні в наявності апаратні модулі.

Універсальна алгоритмічна мова була випробувана в програмному комплексі MC-CAD v0.1, у якому реалізоване дизассемблірування програм чи переклад ассемблерної програми будь-якого типу мікроконтролерів в універсальну алгоритмічну мову з метою подальшої компіляції в ассемблер чи безпосередньо у файл мікроконтролера, що виконується, іншого довільного типу. Комплекс містить у собі конструктор мікроконтролерів, за допомогою якого можна описати кожний з існуючих мікроконтролерів.

ПЕРЕЛІК ПОСИЛАНЬ

1. *Негода Д. В.* Автоматизация проектирования симуляторов микропроцессоров и микроконтроллеров : дис. ... канд. техн. наук : 05.13.12 / Негода Дмитрий Викторович. – Ульяновск : РГБ, 2005. – 160 с.
2. *Джаныбаев К.* Язык спецификаций для верификации набора операций языков // Теория языков и автоматизация программирования : сб. науч. тр. / АН УССР, науч. совет по пробл. «Кибернетика», Ин-т Кибернетики им. В. М. Глушкова; [Ред-кол.: Е. Л. Ющенко]. – К. : ИК, 1986. – С. 19–24.
3. *Стеблянко В. Г.* Разработка входных языков кроссовых систем подготовки программ реального времени / Стеблянко В. Г., Марченко А. И., Подобед Л. Е. // Теория языков и автоматизация программирования : сб. науч. тр. / АН УССР, науч. совет по пробл. «Кибернетика», Ин-т Кибернетики им. В. М. Глушкова; [Ред-кол.: Е. Л. Ющенко]. – К. : ИК, 1986. – С. 41–48.
4. *Белых А. А.* Унификация архитектур однокристальных микроконтроллеров и ее применение для разработки программного обеспечения встраиваемых систем : дис. ... канд. техн. наук : 05.13.15 / Белых Андрей Александрович. – М. : РГБ, 2006. – 179 с.

Надійшла 09.02.2009

Исследованы существующие методы создания алгоритмических языков программирования. Разработан универсальный алгоритмический язык программирования микроконтроллеров, который позволяет переводить программы с одного микроконтроллера в другой, а также задавать общий алгоритм программы при работе с любым видом микроконтроллеров.

The existing methods of creation of algorithmic languages of programming are investigated. The universal programming language of microcontrollers that allows to translate the program from one microcontroller to another and also to set common algorithm of the program at work with any kind of microcontrollers is developed.

УДК 004.853+004.832

В. В. Литвин

МУЛЬТИАГЕНТНІ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, ЩО БАЗУЮТЬСЯ НА ПРЕЦЕДЕНТАХ ТА ВИКОРИСТОВУЮТЬ АДАПТИВНІ ОНТОЛОГІЇ

Досліджено методи побудови та функціонування мультиагентних систем підтримки прийняття рішень, що базуються на прецедентах та використовують адаптивні онтології, які входять у склад інтелектуальних агентів. Розроблено метрику для визначення відстані між прецедентом та поточною ситуацією на основі адаптивних онтологій.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПОСТАНОВКА ПРОБЛЕМИ В ЗАГАЛЬНОМУ ВИДІ

Згідно до сучасного рівня розвитку інформаційних технологій, а саме систем підтримки прийняття рішень, розрізняють два напрями розвитку систем логічного виведення базованих на знаннях [1]:

– системи логічного виведення, заснованого на правилах;

– системи логічного виведення, заснованого на прецедентах.

Практично всі ранні експертні системи моделювали хід ухвалення рішення експертом як дедуктивний процес з використанням логічного виведення, заснованого на правилах. Це означало, що в систему закладалася сукупність правил вигляду «якщо... то...», згідно до яких на підставі вхідних даних генерувався

© Литвин В. В., 2009