нанаоструктурних ГМО, якими керує проф. Edmund Burte, мають значну перспективу при створенні біосенсорів. Питання фінансування спільного проекту цим інститутом та нашою організацією в даний час знаходяться на розгляді в міністерствах урядів Німеччини та України.

На рис. 1.16 наведено блок-схему одного з діагностичних пристроїв та як цей пристрій може бути реалізованим на елементній базі компанії Texas Instruments [3].

І останнє питання, яке безпосередньо стосується проблеми створення біосенсорів – схемотехніка та електронні засоби обробки інформації. Серед задач сенсорної техніки, вирішення яких має пріоритетність, і здебільшого вирішується на схемотехнічному рівні, є підвищення точності вимірювань, забезпечення багатофункціональності, сумісність з комп'ютерною технікою, зниження енергоспоживання та собівартості сенсорних пристроїв. Принципово новою вимогою до сучасних сенсорних пристроїв є можливість функціонування з низьковольтними однополярними джерелами живлення. Основою вирішення цих проблем є спеціалізовані твердотільні ІС. Базуючись на спеціальних алгоритмах функціонування, передових технологіях та засобах автоматизованого проектування, формується новий науково-технічний напрям інтегральної електроніки – створення спеціалізованого класу твердотільних аналогових ІС для сенсорної техніки. Вершиною цього напрямку є інтелектуальні сенсорні пристрої, які характеризуються такими функціями, як самодіагностика, самокорекція, автоматичний перехід з робочого стану в стан очікування, що суттєво зменшує енергоспоживання та довговічність, можливістю передавати дані по уніфікованих протоколах обміну тощо.

Висновки

1. Дана науково-дослідна робота представляє собою лише першу пошукову частину комплексної програми по створенню серії сенсорів біомедичного застосування, для виконання якої в повній мірі необхідне залучення спеціалістів та організацій з хімічної, біологічної та медичної галузей.

2. Основні роботи будуть зосередженні на проблемі інтелектуалізації процесу вимірювання. Новизною виконання даної роботи є вперше продемонструвати можливість розробки спеціалізованого контролера для забезпечення складних алгоритмів вимірювання в магнітних біосенсорах з допомогою мікроелектронної елементної бази нового покоління, і зокрема – мікроконверторів (MicroConverter®) фірми Analog Devices.

Література

1. www.liquidsresearch.com
2. http://www.ttz.uni-magdeburg.de
3. www.ti.com/medical

Надійшла 18.3.2009 р.

УДК 004.052.42: 044.056.52

К.Л. ГОРЯЩЕНКО
Хмельницький національний університет

АСПЕКТИ ЗАХИСТУ ПРОГРАМНОГО КОДУ У ВІДКРИТОМУ АПАРАТНОМУ СЕРЕДОВИЩІ

В статті розглянуті деякі аспекти побудови захищеного програмного середовища за умов відкритого апаратного середовища взаємодії між процесором та зовнішньою пам'яттю. Розглянуто методи підвищення стійкості програмного коду у мікроконтролерних системах з фон Нейманівською архітектурою до несанкціонованого втручання.

Several sides of protected software environment with open hardware architecture of interaction between processor and external memory were discussed in article. Here were considered several different methods of security improvement against unauthorized intrudes in software code for microcontrollers with von Neumann architecture.

Постановка задачі

Якщо розглядати існуючі апаратні рішення сучасної мікропроцесорної та мікроконтролерної РЕА, то можна встановити виготовлення РЕА за вже відомими принципами – це так звані архітектура фон Неймана та Гарвардська архітектура. За цими принципами продовжується розвиток сучасної мікропроцесорної техніки.

З появою апаратно-програмних засобів виникли не тільки задачі використання цих засобів з користю. На жаль, з'явилися бажаючі використати ці апаратно-програмні засоби з іншою метою – побудови програм-вірусів, а також спеціалісти з технічного шпіонажу, мета яких полягає у відновленні первісного коду програмних засобів.

Тобто разом з розробкою чистого програмного продукту виникає необхідність у розробці

спеціалізованого типу програмного забезпечення з метою захисту від можливих зовнішніх втручань.

Аналіз досліджень та публікацій

Розглянемо існуючі архітектури з метою встановлення їх особливостей.

Базовий принцип Гарвардської архітектури полягає у розділенні адресного простору між пам'яттю програм та пам'яттю даних. Найбільше поширення така структура має у мікропроцесорних системах. Так, за принципами Гарвардської структури побудована переважна більшість сучасних 8- та 16-бітних мікроконтролерів (Atmel, Microchip, Texas Instruments, Analog Device [1, 11]) та деяких мікропроцесорів.

Принципово інша ідеологія структури закладена у фон Нейманівську (англ. Von Neumann architecture) структуру системи. Пам'ять програм та пам'ять даних об'єднані у єдиному адресному просторі. Історично саме фон Нейманівська архітектура є першою. Практично до 1970-х років концепція цієї архітектури була єдиною домінуючою для комп'ютерних систем.

Виділення невирішених частин

До суттєвої переваги Гарвардської архітектури необхідно віднести саме апаратне розділення програмного коду та простору даних. Цілком зрозуміло, оскільки в мікроконтролері наявне апаратне розділення пам'яті, то жодний код, що буде розміщено у пам'яті, даних ніколи не буде виконаний як програмний код. Перевагою такої моделі архітектури є можливість отримання різних варіантів розташування пам'яті даних та програми. З них найбільш поширеними є розміщення пам'яті даних та програм в процесорному ядрі. Або розміщення тільки пам'яті програм в процесорному ядрі. Як приклад, можна взяти i8051 (MCS-51).

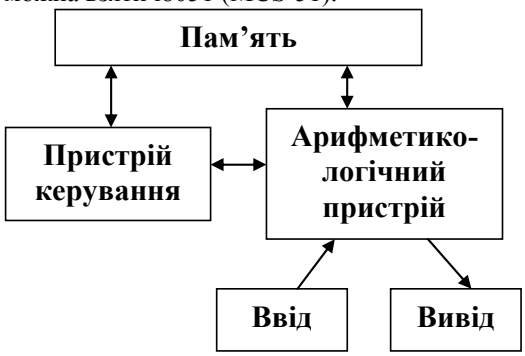

Рис. 1. Архітектура за фон Нейманівським принципом

Тому, якщо виробник програмного забезпечення використав наявні технічні засоби захисту пам'яті від зовнішнього втручання, що наявні в сучасному мікроконтролері з Гарвардською архітектурою, то це дає можливість захистити з високою надійністю програмний продукт від доступу та несанкціонованого дублювання або модифікації.

Зовсім інша ситуація наявна у системах з фон Нейманівської структурою. Найбільш поширеними представниками систем за Гарвардською архітектурою є сучасні мікропроцесорні системи, наприклад, на базі процесорів Intel Pentium, Celeron, AMD Athlon, на базі процесорного ядра ARM7, ARM9, ARM11 та ряду інших.

Метою статті є визначення можливих методів боротьби з несанкціонованим доступом до програмного забезпечення у системах за фон Нейманівською архітектурою та визначення основних пріоритетних шляхів для забезпечення унеможливлення або обмеження такого доступу.

Виклад основного матеріалу. За останніх три десятиріччя, радіоелектронна галузь зробила суттєвий крок у своєму розвитку. Як один з прикладів, можна показати розвиток мікропроцесорної техніки. Починаючи від найпростішого i4040 до сучасних процесорів типу Pentium 4.

Одночасно з технічним суперництвом, що виявляється у розробці альтернативних технічних рішень, існує інший тип суперництва – інформаційний. Скорочення часу виробництва з метою виготовлення альтернативної продукції може бути досягнуто декількома шляхами:

- створення власної науково-технічної бази, що вимагає великих капіталовкладень та характеризується суттєвою затримкою у виробництві в порівнянні з конкурентом-виробником;
- створення спільного підприємства з конкуруючою фірмою;
- розвиток технічного шпіонажу з метою відтворення вже розроблених технічних рішень, що вже реалізовані або плануються до реалізації; цей шлях розвитку дає можливість отримати результат з мінімальною затримкою.

Як видно, фірма-виробник має захищати свої власні інтереси. Такий захист має бути реалізований не тільки на теренах фірми, філій та інших установ. До "власної території", де захищаються інтереси фірми-виробника з повним правом, слід віднести готовий пристрій, що вже наданий користувачу для використання.

Цілком зрозуміло, що захистити внутрішню будову РЕА практично неможливо від стороннього ока [2]. Однак, цілком можливо захистити внутрішній вміст елементів пам'яті пристрою.

Під особливим контролем в сучасній електроніці знаходиться програмний код, що керує роботою пристрою – так званий firmware. До тих пір, доки код є закритим для інших користувачів, лише фірма-виробник, її авторизовані філії або інші довірені установи мають можливість виконувати ремонтні роботи. Це забезпечує монопольне становище не тільки в сегменті продажу, але й обслуговування.

Отримання програмного коду керування приладом відкриває як широкі можливості для користувачів, так і несе загрози виробнику, а саме [5-7]:

- **дублювання** програмного коду – дозволяє виготовляти прилади-клони, це, в свою чергу, підриває монопольне становище фірми-розробника (виробника), а у випадку проблем з цими клонами ще й підриває авторитет фірми-виробника;

- **відновлення** оригінального програмного тексту дозволяє розробляти сторонні "латки", мета яких може полягати у в усуненні недоліків оригінального програмного коду, можливості адаптації до потреб окремої групи користувачів, наданні нових властивостей, так і зміні поведінки приладу при настанні деяких подій-станів;
- **створення** шкідливих "латок" та/або програм, що використовують відомості з відтвореного програмного коду (reverse-engineering).

Як приклад такого стану речей, можна навести програмно-апаратний комплекс PC3000 Російської фірми "ACELab" [9], що направлений на ремонт та відновлення жорстких дисків таких відомих фірм, як "Westerd Digital", "Seagate" (Maxtor/Quantum), "IBM" від найбільш простих до сучасних моделей. Робота даного продукту базується на результатах відновлення тексту програмного коду програм керування жорстких дисків зі скомпільованого байт-коду.

Ще більший розвиток роботи з чужим апаратно-програмним середовищем можна побачити, проаналізувавши програми для роботи з телефонами фірми "BENQ" ("Benq-Siemens"). На рис. 2 показано інтерфейс одної з таких широкодоступних програм – PopuaUtils. Розвиток програмного забезпечення сторонніх виробників надає потужний інструментарій в руки користувача. Але цей інструментарій в руках недосвідченого користувача веде до виходу з ладу телефону, в руках злодія – дозволяє знищити сліди від попереднього власника, відключити телефонні коди.

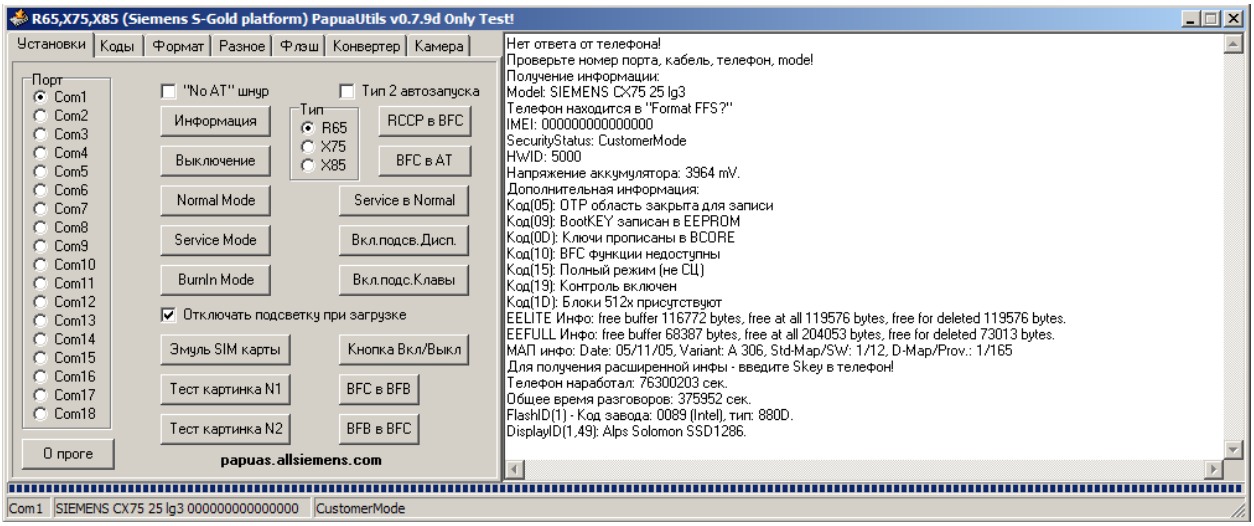

Рис. 2. Інтерфейс PopuaUtils v0.7.9d.
Показана розширена інформація про стан телефону Benq-Siemens CX75

Розглянемо найбільш типові способи захисту програмного продукту в системах із відкритою архітектурою на прикладі систем, побудованих на процесорах сімейства x86 [10].

1. Використання прив'язки до особливостей носіїв пам'яті. Найбільш розповсюджений варіант цього захисту – це використання особливостей форматування носіїв на гнучких дисках. Цей спосіб є досить простим в реалізації, але одночасно породжує масу незручностей для легального користувача програмного забезпечення.

Для мікроконтролерних систем широко застосовувався варіант цього методу – особливості поведінки мікросхем EEPROM при програмуванні за недостатнього рівня напруги програмування. В результаті отримувались так звані "плаваючі" комірки, результат зчитування (0 або 1) з яких є змінною величиною в часі. На жаль, сучасні елементи EEPROM/FlashROM мають власний перетворювач напруги живлення у напругу програмування, а тому цей метод втратив своє застосування через недоступність до безпосереднього маніпулювання.

2. Використання серійного номеру для роботи програмного забезпечення. Робота програми частково або повністю залежить від серійного номеру. Для кінцевого споживача цей спосіб зручний, оскільки дозволяє встановлювати програму безліч разів, копіювати на будь-який носій. Для розробника такий спосіб є незручним, оскільки не дозволяє запобігти розповсюдженню програми з робочим серійним номером.

Для мікроконтролерних систем перевагою такого способу є можливість запису ключа або серійного номеру програми у внутрішній сегмент EEPROM/FlashROM. Це дає можливість кожному пристрою вводити свій унікальний код. Недоліком такого підходу є те, що програма може бути модифікована таким чином, щоб або обійти запити на перевірку вірності кодової послідовності, або навіть зчитати її з оригінального пристрою для дублювання у клоні.

Наявність різноманітних технічних засобів типу "електронного ключа" також не є виходом з даної ситуації. На рис. 3 показано структуру системи та взаємодію із зовнішніми елементами системи.

Оскільки очевидно, що шина даних та адресу є відкритою, то існує повна можливість підключення до цих шин та отримання доступу до інформації.

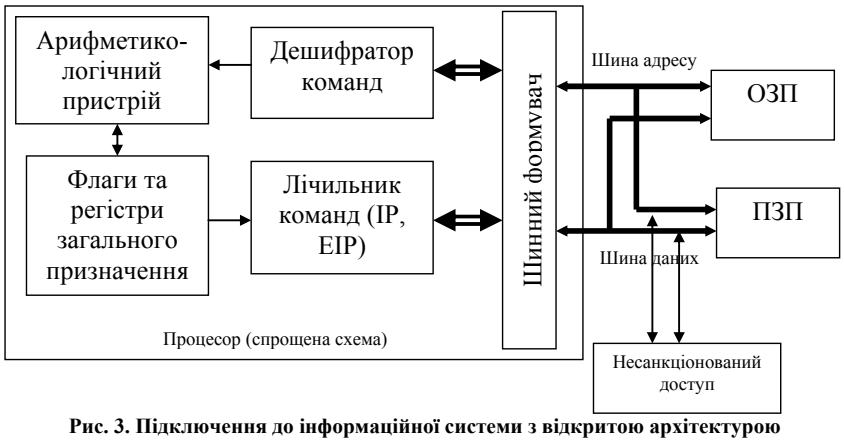

Рис. 3. Підключення до інформаційної системи з відкритою архітектурою

Причина явної простоти несанкціонованого доступу полягає в тому, що програмний код у ОЗП та ПЗП представлений у відкритому вигляді. Отже, дублювання програмного коду та даних виконується без участі процесора взагалі або із мінімальним втручанням з метою встановлення поточних значень робочих регістрів процесора.

Відомі програмні методи, що дозволяють підвищити надійність захисту програмного коду. Наприклад, для систем на базі x86 та ОС Windows широке застосування отримав програмний комплекс ASprotect. Принциповою особливістю комплексу є блочне шифрування програмного коду за допомогою криптостійкого алгоритму. В результаті, в кожний момент часу лише окремий сегмент програмного коду є розшифрованим та доступним, в той час як всі інші є скритими. Тому, змінити та не пошкодити сегмент даних просто неможливо, якщо у сторонньої системи нема ключа на шифрування/дешифрування даних. Недоліком такої системи є її вузькоспеціалізованість – лише системи на базі ОС Windows.

На рис. 4 показаний варіант такого захисту, вільного від прив'язки до операційної системи та прозорого для системи. Основою такого підходу є криптографічний блок з власним блоком ППЗП для розміщення службової інформації – серійного номеру, ключів шифрування та ін. В тому числі задачею цього блоку є створення перехрещень в адресації окремих комірок або блоків даних у зовнішньому ОЗП та ПЗП з метою ускладнення процесу отримання карти перекодування між фізичною та віртуальною адресами. Це ускладнить процес зчитування даних з цих елементів. Крім того кріптографічний блок може виконувати не тільки найпростішу задачу прозорого шифрування, а й більш складну – попередню ініціалізацію ОЗП відповідним змістом з ПЗП. Причому кожне включення блоку буде відповідати генерації нового порядку фізичних адрес ОЗП.

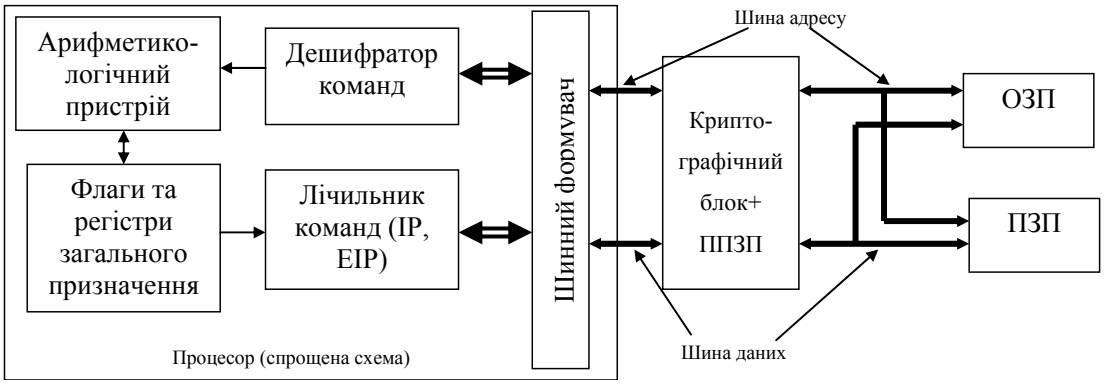

Рис. 4. Кросплатформена система захисту програмного коду від несанкціонованого доступу

Висновки. Перевагою такого підходу є можливість захисту довільного програмного коду, блоків даних та іншої інформації, оскільки криптографічний блок з ППЗП не потребує жодної модифікації програмного коду.

Найбільш можливою основою для реалізації такої системи можуть слугувати системи SOC на базі процесорів ARM7 від Atmel або SOC на базі ПЛІС від Altera, Xilinx.

Література

1. ARM7TDMI Data Sheet / Advanced RISC Machines // ARM DDI 0029E, 1995. – 268 p.

2. Касперски Крис. Техника и философия хакерских атак. – М.: Солон-Пресс, 2004. – 272 с.

3. Майерс Гленфорд Дж. Искусство тестирования программ. – М.: Финансы и статистика, 1982. – 312 с.

4. Расторгуев С.П., Дмитриевский Н.Н. Искусство защиты и раздевания программ. – М.: Совмаркет,

1991. – 94 с.

5. Семьянов П.В., Зегжда Д.П. Анализ средств противодействия исследованию программного обеспечения и методы их преодоления // КомпьютерПресс. – 1993. – № 11. – С. 34-36.

6. Середа С.А. Анализ средств преодоления систем защиты программного обеспечения // ИНФОРМОСТ: Радиоэлектроника и Телекоммуникации. – 2002. – № 4 (22). – С. 11-16.

7. Середа С.А. Оценка эффективности систем защиты программного обеспечения // КомпьЛог. – 2000. – № 2. – С. 56-60.

8. Safe and Secure firmware upgrade for AT91SAM microcontrollers // Atmel Corporation, 2006 / Document № 6253A-ATARM-07-sep-06.

9. Комплекс "РС-3000" // http://www.datarec.ru/company/devices.php

10. Аппаратная защита программного обеспечения // http://karman.com.ua/articles/_apparatnaja_zashchita_0_0_0_823_1.html

11. Мікропроцесорна техніка: Підручник / Ю.І. Якименко, Т.О. Терещенко, Є.І. Сокол та ін. – 2-е вид., переробл. та доповн. – К.: ІВЦ "Видавництво "Політехніка"; "Кондор", 2004. – 440 с.

Надійшла 4.2.2009 р.

УДК 004.78: 004.042

В.Я. ЛЯШКЕВИЧ, А.В. СИДОРУК
Чернівецький національний університет імені Юрія Федьковича

ПРОГРАМНА РЕАЛІЗАЦІЯ ВІРТУАЛЬНОГО ПРОЦЕСОРА ДЛЯ НАПИСАННЯ ПРОГРАМ З МОЖЛИВІСТЮ МОДИФІКАЦІЇ ЇХ КОДУ В ПРОЦЕСІ ВИКОНАННЯ

Розглянуто проблеми розширення гнучкості програмного коду за рахунок модифікації його у виконуваному файлі під час виконання. Розроблено мову програмування та віртуальний процесор, що дозволяють проводити зміни в області даних, коду і команд програми.

Вступ. Сучасні інструментальні засоби мов програмування надають можливість повноцінно використовувати можливості апаратного забезпечення комп'ютерних систем користувачам прикладного програмного забезпечення. Така взаємодія користувача із комп'ютерною системою можлива згідно з наперед визначеними і реалізованими алгоритмами у вигляді завантажувальних *.com чи *.exe файлів [1, 2] Це зумовлено тим, що не завжди є можливість вийти за рамки „безумовних" реакцій на вхідні дані, тобто обмеження полягає в тому, що неможливо в процесі виконання змінити виконавчий файл без перекомпіляції.

Постановка задачі. Системи обробки інформації, можуть змінювати свою структуру в процесі функціонування, але для цього необхідно виконати компіляцію, тривалість якої пропорційно залежить від величини досліджуваної системи. Також, ефективність використання інформаційних систем та можливість їх швидкої адаптації до користувача пов'язані з апаратною та програмною сумісністю, яку можна організувати за допомогою віртуального процесора тому, що фізична архітектура комп'ютерної системи накладає певні обмеження на програми й вимагає їх виконання за певних умов. Основне обмеження – програма являє собою кінцевий файл з незмінним кодом та набором команд, з можливістю проводити зміни тільки в області даних [2].

Для усунення зазначених недоліків пропонується забезпечити:

- можливість якісної динамічної зміни коду, з метою формування алгоритмів, які будуть володіти максимальною гнучкістю. На базі отриманих алгоритмів створювати принципово нові алгоритми, що зможуть змінювати свою структуру в процесі виконання. Зміна повинна відбуватись на рівні початкового алгоритму;
- зв'язок з іншими мовами програмування для використання модулів написаних на інших мовах програмування, наприклад, для забезпечення відповідної швидкодії певних модулів, взаємодії з операційною системою чи іншим прикладним програмним забезпеченням.

Загальні підходи вирішення проблеми гнучкості коду. На сьогодні, вирішенням даної задачі – розроблений ряд віртуальних процесорів, а саме: віртуальна машина Java; Microsoft JScript; Microsoft Visual Basic Scripting; мова сценаріїв ASP; мова VCC та ін. мови (REXX, PERL, Python, LangMF). В їх основі використовуються наведені нижче підходи.

1. Зовнішнє програмування (рис. 1). У випадку невідомих вхідних даних програма робить запит користувачу для подальшого виконання й запису відповідної реакції.